- Enlace de datos
  - Servicios proporcionados a la capa de red
  - Entramado
  - Control de errores
  - Problemas en la capa de enlace de datos:
  - Soluciones propuestas:
    - 1. Retroalimentación con confirmaciones:
    - 2. Temporizadores:
    - 3. Números de secuencia:
  - Objetivo final:
  - Deteccion y correcion de errores
    - Codigo de correcion de errores
      - Codigo de Hamming
        - Calculo del codigo
          - Paso 1: Determinar la cantiad de bits de paridad (r):
          - Paso2 : Posicionar los bits de paridad:
    - Codigo de deteccion de errores
    - 1. Método de Paridad
      - Funcionamiento:
      - Ejemplo:
    - 2. Método de Suma de Verificación (Checksum)
      - Funcionamiento:
      - Ejemplo:
    - 3. Método CRC (Cyclic Redundancy Check)
      - Funcionamiento:
      - Ejemplo:
  - Protocolos Elementales
    - Protocolo simplex utopico
    - Protocolo simplex de parada y espera para un canal libre de errores
    - Protocolo simplex de parada y espera para un canal ruidoso
  - Superposición (Piggybacking)
  - Ventana Desplazantes
    - Funcionamiento Básico
      - Protocolo de ventana deslizante de 1 bit
    - Protocolo que utiliza retroceso N
    - Protocolo Punto a Punto (PPP)
- Subcapa de control de acceso al medio

# Enlace de datos

La capa de enlace de datos utiliza los servicios de la capa física para enviar y recibir bits a través de los canales de comunicación. Tiene varias funciones específicas, entre las que se incluyen:

1. Proporcionar a la capa de red una interfaz de servicio bien definida
2. Manejar los errores de transmisión.
3. Regular el flujo de datos para que los emisores rápidos no saturen a los receptores lentos.

Para cumplir con estas metas, la capa de enlace de datos toma los paquetes que obtiene de la capa de red y los encapsula en tramas para transmitirlos. Cada trama contiene un encabezado, un campo de carga útil (payload) para almacenar el paquete y un terminador. El manejo de las tramas es la tarea más importante de la capa de enlace de datos.

## Servicios proporcionados a la capa de red

La función de la capa de enlace de datos es proveer servicios a la capa de red. El servicio principal es la transferencia de datos de la capa de red en la máquina de origen, a la capa de red en la máqui□na de destino. En la capa de red de la máquina de origen está una entidad, llamada proceso, que entrega algunos bits a la capa de enlace de datos para que los transmita al destino. La tarea de la capa de enlace de

datos es transmitir los bits a la máquina de destino, de modo que se puedan entregar a la capa de red de esa máquina.

La capa de enlace de datos puede diseñarse para ofrecer varios servicios. Los servicios reales ofrecidos varían de un protocolo a otro. Tres posibilidades razonables que normalmente se proporcionan son:

1. **Servicio sin conexión ni confirmación de recepción** : El servicio sin conexión ni confirmación de recepción consiste en hacer que la máquina de origen envíe tramas independientes a la máquina de destino sin que ésta confirme la recepción. Ethernet es un buen ejemplo de una capa de enlace de datos que provee esta clase de servicio. No se establece una conexión lógica de antemano ni se libera después. Si se pierde una trama debido a ruido en la línea, en la capa de datos no se realiza ningún intento por detectar la pérdida o recuperarse de ella. Esta clase de servicio es apropiada cuando la tasa de error es muy baja, de modo que la recuperación se deja a las capas superiores. También es apropiada para el tráfico en tiempo real, como el de voz, en donde es peor tener retraso en los datos que errores en ellos
2. **Servicio sin conexión con confirmación de recepción**: El siguiente paso en términos de confiabilidad es el servicio sin conexión con confirmación de recepción. Cuando se ofrece este servicio tampoco se utilizan conexiones lógicas, pero se confirma de manera individual la recepción de cada trama enviada. De esta manera, el emisor sabe si la trama llegó bien o se perdió. Si no ha llegado en un intervalo especificado, se puede enviar de nuevo. Este servicio es útil en canales no confiables, como los de los sistemas inalámbricos. 802.11 (WiFi) es un buen ejemplo de esta clase de servicio
3. **Servicio orientado a conexión con confirmación de recepción**: el servicio más sofisticado que puede proveer la capa de enlace de datos a la capa de red es el servicio orientado a conexión. Con este servicio, las máquinas de origen y de destino establecen una conexión antes de transferir datos. Cada trama enviada a través de la conexión está numerada, y la capa de enlace de datos garantiza que cada trama enviada llegará a su destino. Es más, garantiza que cada trama se recibirá exactamente una vez y que todas las tramas se recibirán en el orden correcto. Así, el servicio orientado a conexión ofrece a los procesos de la capa de red el equivalente a un flujo de bits confiable. Es apropiado usarlo en enlaces largos y no confiables, como un canal de satélite o un circuito telefónico de larga distancia. Si se utilizara el servicio no orientado a conexión con confirmación de recepción, es posible que las confirmaciones de recepción

perdidas ocasionaran que una trama se enviara y recibiera varias veces, desperdiciando ancho de banda.

# Entramado

Para proveer servicio a la capa de red, la capa de enlace de datos debe usar el servicio que la capa física le proporciona. Lo que hace la capa física es aceptar un flujo de bits puros y tratar de entregarlo al destino. Si el canal es ruidoso, como en la mayoría de los enlaces inalámbricos y en algunos alámbricos, la capa física agregará cierta redundancia a sus señales para reducir la tasa de error de bits a un nivel tolerable. Sin embargo, no se garantiza que el flujo de bits recibido por la capa de enlace de datos esté libre de errores. Algunos bits pueden tener distintos valores y la cantidad de bits recibidos puede ser menor, igual o mayor que la cantidad de bits transmitidos. Es responsabilidad de la capa de enlace de datos detectar y, de ser necesario, corregir los errores. El método común es que la capa de enlace de datos divida el flujo de bits en tramas discretas, calcule un token corto conocido como suma de verificación para cada trama, e incluya esa suma de verificación en la trama al momento de transmitirla. . Cuando una trama llega al destino, se recalcula la suma de verificación. Si la nueva suma de verificación calculada es distinta de la contenida en la trama, la capa de enlace de datos sabe que ha ocurrido un error y toma las medidas necesarias para manejarlo (por ejemplo, desecha la trama errónea y es posible que también devuelva un informe de error).

Es más difícil dividir el flujo de bits en tramas de lo que parece a simple vista. Un buen diseño debe facilitar a un receptor el proceso de encontrar el inicio de las nuevas tramas al tiempo que utiliza una pequeña parte del ancho de banda del canal. cuatro métodos de entramado son:

1. **Conteo de bytes** : se vale de un campo en el encabezado para especificar el número de bytes en la trama. Cuando la capa de enlace de datos del destino ve el conteo de bytes, sabe cuántos bytes siguen y, por lo tanto, dónde concluye la trama. El problema con este algoritmo es que el conteo se puede alterar debido a un error de transmisión. , el destino perderá la sincronía y entonces será incapaz de localizar el inicio correcto de la siguiente trama. Incluso si el destino sabe que la trama está mal puesto que la suma de verificación es incorrecta, no tiene forma de saber dónde comienza la siguiente trama. Tampoco es útil enviar una trama de vuelta a la fuente para solicitar una retransmisión, ya que el destino no sabe

cuántos bytes tiene que saltar para llegar al inicio de la retransmisión. Por esta razón raras veces se utiliza el método de conteo de bytes por sí solo

2. **Bytes banderas con relleno de bytes** : evita el problema de volver a sincronizar nuevamente después de un error al hacer que cada trama inicie y termine con bytes especiales. Con frecuencia se utiliza el mismo byte, denominado byte bandera, como delimitador inicial y final. . Dos bytes bandera consecutivos señalan el final de una trama y el inicio de la siguiente. De esta forma, si el receptor pierde alguna vez la sincronización, todo lo que tiene que hacer es buscar dos bytes bandera para encontrar el fin de la trama actual y el inicio de la siguiente. Se puede dar el caso de que el byte bandera aparezca en los datos, en especial cuando se transmiten datos binarios como fotografías o canciones. Esta situación interferiría con el entramado. Una forma de resolver este problema es hacer que la capa de enlace de datos del emisor inserte un byte de escape especial (ESC) justo antes de cada byte bandera “accidental” en los datos. De esta forma es posible diferenciar un byte bandera del entramado de uno en los datos mediante la ausencia o presencia de un byte de escape antes del byte bandera. La capa de enlace de datos del lado receptor quita el byte de escape antes de entregar los datos a la capa de red. Esta técnica se llama relleno de bytes.
3. **Bits bandera con relleno de bits**: resuelve una desventaja del relleno de bytes: que está obligado a usar bytes de 8 bits. También se puede realizar el entramado a nivel de bit, de modo que las tramas puedan contener un número arbitrario de bits compuesto por unidades de cualquier tamaño. Cada trama empieza y termina con un patrón de bits especial, 01111110 o 0x7E en hexadecimal. Este patrón es un byte bandera. Cada vez que la capa de enlace de datos del emisor encuentra cinco bits 1 consecutivos en los datos, inserta automáticamente un 0 como relleno en el flujo de bits de salida. Este relleno de bits es análogo al relleno de bytes, en el cual se inserta un byte de escape en el flujo de caracteres de salida antes de un byte bandera en los datos. Además asegura una densidad mínima de transiciones que ayudan a la capa física a mantener la sincronización. La tecnología USB (Bus Serie Universal, del inglés Universal Serial Bus) usa relleno de bits por esta razón.
4. **Violaciones de codificación de la capa física** : es utilizar un atajo desde la capa física. Esta redundancia significa que algunas señales no ocurrirán en los datos regulares. Por ejemplo, en el código de línea 4B/5B se asignan 4 bits de datos a 5 bits de señal para asegurar suficientes transiciones de bits. Esto significa que no se utilizan 16 de las 32 posibles señales. Podemos usar algunas señales reservadas para indicar el inicio y el fin de las tramas. En efecto, estamos

usando “violaciones de código” para delimitar tramas. La belleza de este esquema es que, como hay señales reservadas, es fácil encontrar el inicio y final de las tramas y no hay necesidad de rellenar los datos.

# Control de errores

Una vez resuelto el problema de marcar el inicio y el fin de cada trama, llegamos al siguiente dilema: cómo asegurar que todas las tramas realmente se entreguen en el orden apropiado a la capa de red del destino. Suponga por un momento que el receptor puede saber si una trama que recibe contiene la información correcta o errónea. Para un servicio sin conexión ni confirmación de recepción sería ideal si el emisor siguiera enviando tramas sin importarle si llegan en forma adecuada. Pero para un servicio confiable orientado a conexión no sería nada bueno

# Problemas en la capa de enlace de datos:

1. **Tramas perdidas o con errores:**
    - Una trama puede llegar con errores o no llegar debido a fallos en el hardware, ruido en el canal o interferencias.
    - Sin retroalimentación, el emisor no sabe si la transmisión fue exitosa.
2. **Retransmisiones:**
    - Si una trama se retransmite debido a pérdida o errores, existe el riesgo de que el receptor acepte la misma trama más de una vez.
3. **Confirmaciones perdidas:**
    - Si una confirmación de recepción (positiva o negativa) no llega al emisor, este no puede determinar cómo proceder.

---

# Soluciones propuestas:

**1. Retroalimentación con confirmaciones:**

- **Tramas de control especiales** :
- Confirmación positiva (ACK): Indica que una trama llegó correctamente.
- Confirmación negativa (NAK): Indica que una trama llegó con errores y debe retransmitirse.

- Este mecanismo garantiza que el emisor sepa si debe reenviar o continuar con la siguiente trama.

**2. Temporizadores:**

- Al enviar una trama, el emisor inicia un **temporizador** .
- Si la confirmación no llega antes de que el temporizador expire, el emisor retransmite la trama.
- Esto evita esperas indefinidas en caso de pérdida de la trama o de la confirmación.

**3. Números de secuencia:**

- Las tramas llevan un **número de secuencia** único.
- El receptor utiliza este número para distinguir entre:
  - Tramas nuevas.
  - Retransmisiones de tramas ya recibidas.
- Esto garantiza que cada trama sea procesada una sola vez por la capa de red.

---

# Objetivo final:

- **Transmisión confiable:** Asegurar que todas las tramas lleguen al destino en el orden correcto, sin duplicados ni omisiones.
- **Detección de errores:** Identificar tramas erróneas o perdidas y corregirlas mediante retransmisiones.
- **Eficiencia:** Evitar esperas innecesarias y minimizar las retransmisiones

Una **NIC** ( **Network Interface Card** , por sus siglas en inglés) o **tarjeta de interfaz de red** es un componente de hardware que conecta un dispositivo (como una computadora, servidor o impresora) a una red. La NIC permite que el dispositivo envíe y reciba datos a través de la red, ya sea mediante medios cableados o inalámbricos. Pertenece a la capa de enlace.

# Deteccion y correcion de errores

Los diseñadores de redes han desarrollado dos estrategias básicas para manejar los errores. Ambas añaden información redundante a los datos que se envían. Una es incluir suficiente información redundante para que el receptor pueda deducir cuáles debieron ser los datos transmitidos. La otra estrategia es incluir sólo suficiente redundancia para permitir que el receptor sepa que ha ocurrido un error (pero no qué error) y entonces solicite una retransmisión. La primera estrategia utiliza códigos de corrección de errores; la segunda usa códigos de detección de errores

# Codigo de correcion de errores

1. Códigos de Hamming
2. Códigos convolucionales binarios : En un código convolucional, un codificador procesa una se□cuencia de bits de entrada y genera una secuencia de bits de salida. No hay un tamaño de mensaje natural, o límite de codificación, como en un código de bloque. La salida depende de los bits de entrada actual y previa. Es decir, el codificador tiene memoria. El número de bits previos de los que depende la salida se denomina longitud de restricción del código. Los códigos convolucionales se especifican en términos de su tasa de transmisión y su longitud de restricción.
3. **Códigos de Reed-Solomon**: Al igual que los códigos de Hamming, los códigos de Reed-Solomon son códigos de bloques lineales y con frecuencia también son sistemáticos. A diferencia de los códigos de Hamming, que operan sobre bits individuales, los códigos de Reed-Solomon operan sobre símbolos de m bits.
4. **Códigos de verificación de paridad de baja densidad** : En un código LDPC, cada bit de salida se forma sólo a partir de una fracción de los bits de entrada. Esto conduce a una representación matricial del código con una densidad baja de 1s, razón por la cual tiene ese nombre. Las palabras codificadas recibidas se decodifican con un algoritmo de aproximación que mejora de manera reiterativa con base en el mejor ajuste de los datos recibidos con una palabra codificada válida. Esto corrige los errores. álida. Esto corrige los errores. Los códigos LDPC son prácticos para tamaños grandes de bloques y tienen excelentes habilidades de corrección de errores que superan a las de muchos otros códigos (incluyendo los que vimos antes) en la práctica. Por esta razón se están incluyendo rápidamente en los nuevos protocolos. Forman parte del estándar para la difusión de video digital, la Ethernet de 10 Gbps, las redes de líneas eléctricas y la versión más reciente de 802.11

Todos estos códigos agregan redundancia a la información que se envía. Una trama consiste en m bits de datos (mensaje) y r bits redundantes (verificación). En un código de bloque, los r bits de verificación se calculan únicamente en función de los m bits de datos con los que se asocian, como si los m bits se buscaran en una gran tabla para encontrar sus correspondientes r bits de verificación

## Codigo de Hamming

Se agrega información redundante (bits de paridad) a los datos originales para que el receptor pueda detectar y corregir errores. Los bits de paridad se calculan usando relaciones lógicas (XOR) entre bits específicos de los datos. En el receptor, los bits de paridad se recalculan y comparan con los recibidos. Si hay una discrepancia, se identifica la posición del bit erróneo y se corrige.

#### Calculo del codigo

##### Paso 1: Determinar la cantiad de bits de paridad (r):

Se necesitan rr**r** bits de paridad para proteger mm**m** bits de datos, donde:

$$2^r \geq m + r + 1$$

Ejemplo para proteger 4 bits de datos (m=4) se necesitan r=3 ya que : $8 \geq 4 + 3 + 1$

##### Paso2 : Posicionar los bits de paridad:

Los bits de paridad se colocan en posiciones que son potencias de 2 (1,2,4,8,…)

Ejemplo:

- Datos : D1,D2,D3,D4
- Estructura final : P1,P2, D1, P4 ,D2, D3, D4

### Paso 3: Calcular los bits de paridad:

- Cada bit de paridad cubre un grupo específico de bits según su posición.
- Los grupos se determinan por las posiciones cuyos números binarios tienen un 11**1** en la posición correspondiente.

### Ejemplo para P1:

- P1 verifica las posiciones 1, 3, 5, 7.

- $P1 = D1 \oplus D2 \oplus D4$

# Codigo de deteccion de errores

Los códigos de corrección de errores se utilizan de manera amplia en los enlaces inalámbricos, que son notoriamente más ruidosos y propensos a errores si se les compara con la fibra óptica. Sin los códigos de corrección de errores sería difícil hacer pasar cualquier cosa. Sin embargo, a través de la fibra óptica o del cable de cobre de alta calidad, la tasa de error es mucho más baja, por lo que la detección de errores y la retransmisión por lo general son más eficientes para manejar un error ocasional.

Tres códigos de detección de errores distintos. Todos son códigos de bloques sistemáticos lineales:

1. Paridad
2. Sumas de verificación
3. Pruebas de Redundancia Cíclica (CRC)

# 1. Método de Paridad

El método de **paridad** es uno de los más simples para la **detección de errores** .

**Funcionamiento:**

- **Objetivo:** Verificar si el número total de bits **1** en los datos recibidos es par o impar, dependiendo del tipo de paridad utilizado.
- **Proceso:**
    1. **En el transmisor:**
        - Se calcula el número de bits **1** en el mensaje.
        - Si se usa **paridad par** , se agrega un bit adicional para que el número total de **1** sea par.
        - Si se usa **paridad impar** , se agrega un bit adicional para que el número total de **1** sea impar.
    2. **En el receptor:**
        - Se recalcula la paridad del mensaje recibido.
        - Si la paridad no coincide con lo esperado, se detecta un error.
- **Limitación:** Solo detecta errores simples (un número impar de bits alterados). Si hay un número par de bits alterados, no se detecta.

**Ejemplo:**

- Datos originales: **110101**
- **Paridad par:**
  - Número de **1**: **4** (par).
  - Bit de paridad: **0** (para mantenerlo par).
  - Datos enviados: **1101010**.
- **Paridad impar:**
  - Número de **1**: **4** (par).
  - Bit de paridad: **1** (para hacerlo impar).
  - Datos enviados: **1101011**.

---

# 2. Método de Suma de Verificación (Checksum)

**Funcionamiento:**

- **Objetivo:** Detectar errores mediante la suma de segmentos de datos y el envío del complemento de la suma como verificación.
- **Proceso:**
  1. **En el transmisor:**
     - Los datos se dividen en bloques de longitud fija.
     - Se calculan las sumas de estos bloques.
     - Si la suma excede la capacidad del bloque (overflow), se envuelve el exceso (carry-around sum).
     - Se toma el complemento de la suma y se envía como el **checksum** .
  2. **En el receptor:**
     - Se realizan los mismos pasos para calcular la suma de los datos recibidos, incluido el checksum.
     - Si la suma resultante es 00**0**, los datos son correctos. De lo contrario, hay un error.
- **Limitación:** Es menos robusto que otros métodos como CRC para detectar errores más complejos.

**Ejemplo:**

- Bloques de datos (en binario): **1101**,**1010**,**0111**

- Suma binaria: 1111
- Complemento: **0000** (checksum enviado).
- En el receptor, la suma de los bloques y el checksum debe dar **0000**.

---

# 3. Método CRC (Cyclic Redundancy Check)

El CRC es un método más avanzado que utiliza divisiones polinomiales para detectar errores. Es ampliamente usado en redes y almacenamiento.

**Funcionamiento:**

1. **Representación de datos:**
   - Los datos a transmitir se representan como un polinomio en binario.
   - Por ejemplo, el mensaje **1101** se representa como x^**3**+x^**2**+1
2. **División polinómica:**
   - Se elige un polinomio generador **G**(**x**), que es conocido por el emisor y receptor.
   - Los datos se expanden agregando ceros al final (según el grado de **G**(**x**)).
   - Se realiza una división binaria (XOR) entre los datos expandidos y **G**(**x**).
   - El residuo de la división (R) se agrega al mensaje original.
3. **Envío de datos:**
   - El mensaje transmitido es: **M**+**R** (datos originales más el residuo).
4. **En el receptor:**
   - Se realiza la misma división con el polinomio generador **G**(**x**).
   - Si el residuo resultante es **0**, los datos están correctos. De lo contrario, hay un error.

**Ejemplo:**

- Datos originales: **1101**.
- Generador **G**(**x**)): **1011** (polinomio de grado 3).
- Datos expandidos: **1101000** (se agregan 3 ceros).
- División:
  - **1101000÷1011=1001** (cociente), residuo: **111**.
- Datos transmitidos: **1101111** (datos originales + residuo **111**).
- En el receptor:
  - **1101111÷1011**. Si el residuo es **0**, no hay errores.

# Protocolos Elementales

## Protocolo simplex utopico

Para este algoritmo asumiremos que:

1. El envio y la recepcion son procesos independientes
2. La comunicacion es unidireccional
3. Los equipos y medios son confiable

Sin control de flujo ni corrección de errores

```
void sender(void){
    frame s;
    packet buffer;
    while (true){
        from_network_layer(&buffer);
        s.info = buffer;
        to_physical_layer(&s);
    }
}

void receiver(void){
    frame r;
    event_type event;
    while (true){
        wait_for_event(&event);
        from_physical_layer(&r);
        to_network_layer(&r.info)
    }
}
```

## Protocolo simplex de parada y espera para un canal libre de errores

Ahora debemos lidiar con el problema principal de evitar que el emisor sature al receptor enviando tramas a una mayor velocidad de la que este último puede procesarlas

Control de flujo sin corrección de errores

```
void sender(void){
    frame s;
    packet buffer;
    event_type event;
    while (true){
        from_network_layer(&buffer);
        s.info = buffer;
        to_physical_layer(&s);
        wait_for_event(&event);
    }
}

void receiver(void){
    frame r, s;
    event_type event;
    while (true){
        wait_for_event(&event);
        from_physical_layer(&r);
        to_network_layer(&r.info)
        to_physical_layer(&s);
    }
}
```

# Protocolo simplex de parada y espera para un canal ruidoso

A pesar de que se puede advertir ya la deteccion de tramas dannadas o perdidas o detectadas correctamente mediante un frame que devuelve al emisor . Sin embargo es necesario advertir un temporizador para retransmitir la trama nuevamente y ademas evitar tramas enviadas duplicadas en caso de perderse por el camino el frame de confirmacion. Para ello el receptor debe manejar por el numero de secuencia que la trama que llega para ver si es una trama nueva o un duplicado que puede descartarse.

Los protocolos en los que el emisor espera una confirmacion de recepcion positiva antes de avanzar al siguiente elemento de datos se llaman **ARQ** (Solicitud Automatica de Repeticion) o **PAR** (Confirmacion de Recepcion Positiva con Retransmision)

```
void sender(void) {
    seq_nr next_frame_to_send;  // Número de secuencia del próximo marco a enviar.
    frame s;                    // Marco a enviar.
    packet buffer;              // Paquete desde la capa de red.
    event_type event;           // Tipo de evento (como llegada de un ACK).

    next_frame_to_send = 0;     // Inicializa el número de secuencia.
    from_network_layer(&buffer); // Obtiene el primer paquete de la capa de red.
```

```
    while (true) {
        s.info = buffer;           // Coloca el paquete en el marco.
        s.seq = next_frame_to_send; // Asigna el número de secuencia al marco.
        to_physical_layer(&s);     // Envía el marco por el canal físico.
        start_timer(s.seq);        // Inicia el temporizador para la retransmisión.
        wait_for_event(&event);    // Espera un evento, como un ACK o un
temporizador vencido.
        if (event == frame_arrival) {  // Si llega un marco del receptor...
            from_physical_layer(&s);  // Extrae el marco del canal físico.
            if (s.ack == next_frame_to_send) { // Si el ACK coincide con el marco
enviado...
                stop_timer(s.ack);           // Detiene el temporizador.
                from_network_layer(&buffer); // Obtiene el siguiente paquete de la
capa de red.
                next_frame_to_send++;        // Incrementa el número de secuencia.
            }
        }
    }
}
```

```
void receiver(void) {
    seq_nr frame_expected;   // Número de secuencia esperado.
    frame r, s;              // `r`: Marco recibido, `s`: ACK a enviar.
    event_type event;        // Tipo de evento (como llegada de un marco).

    frame_expected = 0;      // Inicializa el número de secuencia esperado.
    while (true) {
        wait_for_event(&event);        // Espera un evento, como la llegada de un
marco.
        if (event == frame_arrival) { // Si llega un marco...
            from_physical_layer(&r);   // Extrae el marco del canal físico.
            if (r.seq == frame_expected) { // Si el número de secuencia coincide
con el esperado...
                to_network_layer(&r.info); // Pasa los datos del marco a la capa de
red.
                frame_expected++;         // Incrementa el número de secuencia
esperado.
            }
            s.ack = 1 - frame_expected;  // Prepara un ACK para el marco procesado.
            to_physical_layer(&s);      // Envía el ACK al emisor.
        }
    }
}
```

# Superposición (Piggybacking)

1. **Transmisión Full-Dúplex:**

- En lugar de usar enlaces separados para enviar datos en ambas direcciones, se utiliza un único enlace para transmitir datos y confirmaciones (ACK).
- Las tramas de datos y las confirmaciones se diferencian mediante el campo `kind` en el encabezado.

2. **Superposición (Piggybacking):**
   - Permite incluir la confirmación (ACK) en las tramas de datos de salida, evitando tramas de control independientes.
   - **Ventajas:**
     - Ahorra ancho de banda.
     - Reduce la sobrecarga de procesamiento.
   - **Desafío:** Determinar cuánto tiempo esperar un nuevo paquete antes de enviar un ACK independiente. Se puede usar un tiempo fijo para decidir.

# Ventana Desplazantes

- **Número de Secuencia:**
  - Cada trama enviada se identifica con un número de secuencia único.
  - Estos números se asignan cíclicamente entre `0` y un valor máximo (por lo general `2^n - 1`, donde `n` es el número de bits usados para representar el número de secuencia).

- **Ventanas:**
  - **Ventana del emisor:** Es el conjunto de tramas que el emisor puede enviar pero que aún no han sido confirmadas.
  - **Ventana del receptor:** Es el conjunto de tramas que el receptor puede aceptar.

  Estas ventanas son dinámicas y se "deslizan" conforme se envían y reciben tramas.

- **Tamaño de Ventana:**
  - Define cuántas tramas pueden estar "en vuelo" sin confirmación.
  - Ejemplo:
    - Si el tamaño de la ventana del emisor es 3, puede enviar hasta 3 tramas sin esperar confirmaciones.

- Si el tamaño de la ventana del receptor es 1, solo aceptará la siguiente trama esperada en orden.

# Funcionamiento Básico

1. **Ventana del Emisor:**
   - Contiene tramas que han sido enviadas pero no confirmadas (ACK no recibido).
   - Cada vez que se recibe una confirmación (ACK) de una trama, el extremo inferior de la ventana avanza.
   - Si llega un paquete nuevo desde la capa de red, se asigna el siguiente número de secuencia y la ventana se desliza.

   Ejemplo: Si la ventana del emisor es `[3, 4, 5]` y se confirma la trama `3`, la ventana pasa a `[4, 5, 6]`.

2. **Ventana del Receptor:**
   - Contiene los números de secuencia de las tramas que puede aceptar.
   - Si recibe una trama dentro de la ventana, la guarda y genera una confirmación.
   - Si recibe una trama fuera de la ventana (por ejemplo, ya confirmada o fuera de orden), la descarta.

   Ejemplo: Si la ventana del receptor es `[3, 4, 5]` y recibe la trama `3`, la ventana se desliza a `[4, 5, 6]`.

### Protocolo de ventana deslizante de 1 bit

```
void protocol4 (void)
{

seq_nr next_frame_to_send; /* sólo 0 o 1 */

seq_nr frame_expected; /* sólo 0 o 1 */

frame r, s; /* variables de trabajo */

packet buffer; /* paquete actual que se envía */

event_type event;
```

```
next_frame_to_send = 0; /* siguiente trama del flujo de salida */

frame_expected = 0; /* próxima trama esperada */

from_network_layer(&buffer); /* obtiene un paquete de la capa de red */

s.info = buffer; /* se prepara para enviar la trama inicial */

s.seq = next_frame_to_send; /* inserta el número de secuencia en la trama */

s.ack = 1 - frame_expected; /* confirmación de recepción superpuesta */

to_physical_layer(&s); /* transmite la trama */

start_timer(s.seq); /* inicia el temporizador */

while (true){
        wait_for_event(&event); /*
frame_arrival, cksum_err o timeout */
        if (event == frame_arrival){ /* ha llegado una trama sin daño. */
            from_physical_layer(&r); /* la obtiene */
            if(r.seq == frame_expected) { /* maneja flujo de tramas de entrada. */
                to_network_layer(&r.info); /* pasa el paquete a la capa de red */
                inc(frame_expected); /* invierte el siguiente número de secuencia
esperado */
            }

            if(r.ack == next_frame_to_send){ /* maneja flujo de tramas de salida.
*/
                 stop_timer(r.ack); /* desactiva el temporizador */
                 from_network_layer(&buffer); /* obtiene un nuevo paquete de la
capa de red */
                 inc(next_frame_to_send); /* invierte el número de secuencia del
emisor */

            }

         }
         s.info = buffer; /* construye trama de salida */
         s.seq = next_frame_to_send; /* le inserta el número de secuencia */
         s.ack = 1 - frame_expected; /* número de secuencia de la última trama
recibida */
         to_physical_layer(&s); /* transmite una trama */
         start_timer(s.seq); /* inicia el temporizador */
}
}
```

El problema con este protocolo es cuando la comunicacion inicia simultaneamente, se cruzan sus tramas.

# Protocolo que utiliza retroceso N

En vez de mantener y bloquear el receptor por trama, se envian varaias tramas por el canal de forma continua para luego bloquearlo. Esta estrategia mejora significativamente el tiempo de espera y se aprovecha el ancho de banda. Esta tecnica de mantener varias tramas en movimineot es un ejemplo de canalizacion (pipeling).

La canalización de tramas a través de un canal de comunicación no confiable presenta problemas serios. Primero, ¿qué ocurre si una trama a la mitad de un flujo extenso se daña o pierde? Llegarán grandes cantidades de tramas sucesivas al receptor antes de que el emisor se entere de que algo anda mal. Cuando llega una trama dañada al receptor es obvio que debe descartarse pero, ¿qué debe hacer el receptor con todas las tramas correctas que le siguen? Recuerde que la capa de enlace de datos del receptor está obligada a entregar paquetes a la capa de red en secuencia.

Para ellos hay dos metodos , ambos tienen como objetivo garantizar que las tramas se reciban correctamente y, en caso de error, volver a enviarlas de manera eficiente.

- Retroceso n:
- Repeticion selectiva:

El protocolo **Go-Back-N** se utiliza para manejar errores en la transmisión de tramas cuando la ventana de recepción del receptor es de tamaño 1. Es decir, el receptor sólo puede aceptar la trama esperada, y todas las tramas siguientes se descartan si una trama intermedia se pierde o se recibe con error.

**Funcionamiento:**

1. El emisor puede enviar múltiples tramas (hasta el tamaño de su ventana), pero el receptor sólo puede aceptar una trama a la vez. Cuando recibe una trama correcta, la pasa a la capa superior y envía una confirmación de recepción (ACK) para esa trama.
2. Si el receptor detecta una trama dañada o perdida (por ejemplo, debido a un error de verificación), no envía ningún ACK para esa trama. Todas las tramas siguientes que haya recibido también se descartan, ya que deben llegar en el orden correcto.
3. El emisor, al no recibir un ACK para una trama, la reenvía junto con las tramas siguientes, incluso si algunas de ellas se recibieron correctamente.

La **repetición selectiva** es más eficiente que el retroceso N, ya que permite al receptor aceptar y almacenar tramas correctas que recibe, incluso si alguna de las tramas anteriores está dañada o perdida.

**Funcionamiento:**

1. El emisor también puede enviar múltiples tramas, pero a diferencia de **Go-Back-N** , el receptor no descarta las tramas correctas que llegan después de una trama perdida. El receptor puede almacenar estas tramas correctamente recibidas en un búfer.
2. Si una trama se pierde o se recibe con error, el receptor no envía un ACK para esa trama. Sin embargo, cuando el emisor recibe un ACK o una confirmación de recepción negativa (NAK) para la trama perdida, sólo retransmite esa trama.
3. El receptor puede entonces entregar las tramas en orden cuando la trama perdida se haya retransmitido y recibido correctamente, permitiendo que las tramas almacenadas en el búfer se entreguen a la capa superior en el orden correcto.

# Protocolo Punto a Punto (PPP)

es un estándar ampliamente utilizado para la transmisión de datos a través de enlaces de comunicación punto a punto. Es comúnmente implementado en redes de área amplia (WAN) para conectar dispositivos como enrutadores, computadoras y módems a través de distintos tipos de enlaces, tales como **fibra óptica SONET** y **ADSL.** Constituye una mejora del protocolo mas simple conocido como SLIP (Protocolo de Linea Serial de Internet).

Se utiliza para manejar la configuración de detección de errores en los enlaces, soporta múltiples protocolos, permite la autentificación y tiene muchas otras funciones. Con un amplio conjunto de opciones, PPP provee tres características principales:

1. Un método de entramado que delinea sin ambigüedades el final de una trama y el inicio de la siguiente. El formato de trama también maneja la detección de errores.
2. Un protocolo de control de enlace para activar líneas, probarlas, negociar opciones y desactivarlas en forma ordenada cuando ya no son necesarias. Este protocolo se llama LCP (Protocolo de Control de Enlace, del inglés Link Control Protocol).
3. Un mecanismo para negociar opciones de capa de red con independencia del protocolo de red que se vaya a utilizar. El método elegido debe tener un NCP (Protocolo de Control de Red, del inglés Network Control Protocol) distinto para cada capa de red soportada.

Esta orientado a bytes, de fomra que el relleo de bytes y las tramas tienen numero entero de bytes.

**ADSL** es una tecnología que se utiliza en redes de acceso de última milla, permitiendo conexiones de banda ancha a Internet mediante la infraestructura de la red telefónica. ADSL proporciona un canal de transmisión asimétrico, donde la velocidad de descarga (desde Internet hacia el usuario) es mucho mayor que la velocidad de subida (desde el usuario hacia Internet).

**PPP** (protocolo de la capa de enlace de datos)

Su funcionalidad incluye:

- **Encapsulación de paquetes de red:** PPP puede encapsular diversos protocolos de red, lo que lo hace adecuado para ser utilizado en diferentes tipos de enlaces de comunicación (como SONET o ADSL).
- **Control de errores y detección de fallos:** PPP incluye mecanismos para detectar y corregir errores en la transmisión de datos, lo que asegura la fiabilidad de la comunicación.
- **Autenticación:** PPP soporta autenticación, lo que permite verificar la identidad de los dispositivos que se comunican a través de los enlaces.
- **Compresión:** El protocolo puede usar compresión de datos para mejorar la eficiencia en el uso del ancho de banda.

# Subcapa de control de acceso al medio

Los protocolos que se utilizan para determinar quién sigue en un canal multiacceso pertenecen a una subcapa de la capa de enlace de datos llamada subcapa MAC (Control de Acceso al Medio, del inglés Medium Access Control). La subcapa MAC tiene especial importancia en las LAN, en especial las inalámbricas puesto que el canal inalámbrico es de difusión por naturaleza. En contraste, las WAN usan enlaces punto a punto, excepto en las redes satelitales. Es la parte inferior de la capa de enlace de datos.

# Problema de asignacion del canal

El problema de asignacion de canales consiste en que solo existe un medio compartido por n usuarios y el uso compartido por parte de todos ellos de forma

inadecuada interfiere con el resto de los usuarios.

# Asignacion estatica del canal

Se tienen en cuenta dos metodos tradicionales de multiplexacion:

- **FDM (Multiplexión por División de Frecuencia): En FDM, el ancho de banda disponible de un canal se divide en múltiples subcanales, y cada usuario recibe un subcanal para transmitir o recibir datos. Este enfoque es adecuado cuando el número de usuarios es pequeño y constante, o cuando el tráfico es estable y predecible. Sin embargo, presenta varios problemas cuando, el numero de usuarios es gradne y variable ; trafico de rafagas**
- **TDM (Multiplexión por División de Tiempo): En TDM, el canal se divide en segmentos de tiempo, y a cada usuario se le asigna una ranura de tiempo. Si un usuario no utiliza su ranura asignada, el tiempo se desperdicia, lo que también resulta en una asignación ineficiente cuando el tráfico es variable. Si se utilizara un conjunto de redes de 10 Mbps en lugar de una de 100 Mbps y se asignara una red a cada usuario, el retardo promedio se incrementaría.**

La asignación estática (ya sea por FDM o TDM) es ineficiente en redes donde el tráfico es **variable o en ráfagas** . Cuando los usuarios no usan su canal de forma constante, ya sea por inactividad o por picos de tráfico, el ancho de banda se desperdicia. Este enfoque no se adapta bien a las necesidades dinámicas de la mayoría de los sistemas de cómputo modernos, donde las cargas de tráfico varían enormemente.

# Supuestos para la asignacion dinamica de canales

Se establece supuestos claves que subyacen en los protoclos de accesos mutliples para la asignacion de canales de comunicacion. Estos protocolos permiten que multiples estaciones de comunicacion compartan un unico canal, gestionando el acceso al medio para evitar colisiones y maximizar la eficiencia de transmision:

- Trafico independiente
- Canal Unico

- Colisiones observables
- Tiempo continuo o ranurado
- Deteccion de portadora o sin deteccion de portadora

# Protocolos de Acceso Multiple

## Aloha puro

El **sistema ALOHA** es un protocolo simple para el acceso múltiple a un canal compartido. Permite que los usuarios transmitan cuando tengan datos, pero debido a la naturaleza de acceso no coordinado, las transmisiones pueden superponerse y causar **colisiones** , lo que lleva a la pérdida de las tramas involucradas.

El **sistema ALOHA** es un protocolo simple para el acceso múltiple a un canal compartido. Permite que los usuarios transmitan cuando tengan datos, pero debido a la naturaleza de acceso no coordinado, las transmisiones pueden superponerse y causar **colisiones** , lo que lleva a la pérdida de las tramas involucradas. Aquí están los puntos clave de este protocolo:

**1. Proceso de transmisión:**

- Las estaciones transmiten tramas a la computadora central.
- Si una estación recibe una confirmación de que su trama llegó correctamente, continúa. Si la trama sufrió una colisión, la estación debe retransmitirla.
- El tiempo de retransmisión es **aleatorio** para evitar que las tramas se colisionen repetidamente en sincronía.

**2. Sistema de Contención:**

- ALOHA es un sistema de **contención** , donde varios usuarios comparten un canal común y pueden transmitir al mismo tiempo, lo que puede generar colisiones.

**3. Colisiones:**

- Cuando dos tramas se transmiten simultáneamente, se colisionan. Esto ocurre si el primer bit de una trama se traslapa con el último bit de otra.

- Ambas tramas son destruidas, y las estaciones involucradas deben retransmitir sus tramas.

**4. Eficiencia del Canal ALOHA:**

- La **eficiencia** de ALOHA se refiere a qué fracción de las tramas transmitidas escapan a las colisiones.
- El sistema tiene **limitada eficiencia** debido a la naturaleza aleatoria de las transmisiones y las colisiones.

**5. Modelo de Tráfico:**

- El tráfico generado por los usuarios sigue una **distribución de Poisson** con una media de **N tramas por tiempo de trama** .
- Si N > 1, el canal está sobrecargado, y la mayoría de las tramas colisionarán. Si N < 1, hay menos colisiones y el rendimiento mejora.

**6. Retransmisiones:**

- Además de las nuevas tramas, las estaciones retransmiten tramas previamente colisionadas. Las tramas nuevas y las retransmisiones combinadas siguen una distribución de Poisson con media **G tramas por tiempo de trama** .

**7. Probabilidad de Éxito ($P_0$):**

- La probabilidad de que una trama se transmita sin colisiones, es decir, la probabilidad de éxito, se calcula como **$P_0$ = e^(-2G)** , donde **G** es la carga de tráfico (la tasa de tramas generadas).
- La **velocidad real de transmisión (S)** se calcula como **S = G * $P_0$ = G * e^(-2G)** .

**8. Máxima Eficiencia:**

- La **máxima eficiencia** se alcanza cuando **G = 0.5** , lo que da una tasa de **S ≈ 0.184** . Esto significa que solo aproximadamente **18%** de las tramas transmitidas tienen éxito, ya que el canal está sobrecargado debido a las colisiones.

# Aloha ranurado

El **ALOHA ranurado** es una variante del protocolo ALOHA que mejora la eficiencia de acceso al canal compartido al dividir el tiempo en intervalos discretos o **ranuras** . Cada ranura corresponde a una trama, y las estaciones deben esperar al inicio de una ranura para transmitir, en lugar de hacerlo de manera continua como en el ALOHA puro. Esto introduce una sincronización en las transmisiones y reduce el **periodo vulnerable** (el tiempo durante el cual las tramas pueden colisionar). A continuación, se detallan los aspectos clave del ALOHA ranurado:

**1. Principio de Funcionamiento:**

- **División del tiempo en ranuras:** En lugar de transmitir en cualquier momento, las estaciones esperan al inicio de una ranura para enviar su trama.
- **Reducción del periodo vulnerable:** Al esperar a que comience una nueva ranura, las colisiones ocurren solo dentro de esa ranura, lo que reduce el tiempo durante el cual pueden ocurrir colisiones. Esto hace que la probabilidad de colisión se reduzca comparada con ALOHA puro.

**2. Eficiencia del Canal:**

- **Probabilidad de éxito:** La probabilidad de que una transmisión no choque con otra durante una ranura es **e^(-G)** , donde **G** es la carga del canal, es decir, el número de tramas generadas por las estaciones por ranura de tiempo.
- La **velocidad real de transmisión (S)** en ALOHA ranurado es **S = G * e^(-G)** . Esto alcanza su máxima eficiencia cuando **G = 1** , con una velocidad de transmisión de **S ≈ 1/e ≈ 0.368** , que es **el doble** de la eficiencia del ALOHA puro, que tenía un rendimiento máximo de **0.184** .

**3. Distribución de Tráfico:**

- **Distribución de Poisson:** Las tramas generadas siguen una distribución de Poisson, con una media de **G** tramas por ranura de tiempo.
- **Probabilidad de ranura vacía y colisiones:**
  - Cuando el sistema opera a **G = 1** , alrededor del **37%** de las ranuras estarán vacías, **37%** tendrá éxito, y **26%** sufrirá colisiones.
  - Si **G** aumenta, el número de **ranuras vacías** disminuye, pero las **colisiones aumentan exponencialmente** .

**4. Colisiones y Retransmisiones:**

- **Probabilidad de colisión:** La probabilidad de colisión en una ranura es **1 − e^(-2G)** . Si una transmisión choca, se requiere un nuevo intento.
- El número esperado de transmisiones para una línea (considerando colisiones y retransmisiones) está relacionado con **G** . A medida que **G** aumenta, el rendimiento se degrada rápidamente debido a un mayor número de colisiones.

**5. Aplicaciones Modernas:**

- El ALOHA ranurado fue utilizado inicialmente en los primeros sistemas experimentales en la década de 1970, pero fue en gran parte olvidado hasta que problemas modernos de acceso compartido en redes, como el **Internet a través de cable** y la **comunicación entre etiquetas RFID y lectores** , lo revivieron.
- Su capacidad para resolver problemas de asignación de canales compartidos ha sido redescubierta en aplicaciones actuales, demostrando que protocolos aparentemente obsoletos pueden encontrar nuevas utilidades.

**6. Impacto de Carga Alta (G alto):**

- A medida que **G** aumenta, se reduce la eficiencia del sistema debido a un **aumento exponencial de las colisiones** , lo que implica que un mayor tráfico en el canal provoca un peor rendimiento.

# Protocolos de acceso múltiple con detección de portadora

Con el ALOHA ranurado, el mejor aprovechamiento de canal que se puede lograr es 1/e. Este resultado tan bajo no es muy sorprendente pues, con estaciones que transmiten a voluntad propia, sin prestar atención a lo que están haciendo las demás estaciones, es inevitable que haya muchas colisiones. Sin embargo, en las redes LAN es posible que las estaciones detecten lo que están haciendo las demás estaciones y adapten su comportamiento con base en ello. Estas redes pueden lograr una utilización mucho mejor que 1/e.

Los protocolos en los que las estaciones escuchan una portadora (es decir, una transmisión) y actúan de manera acorde se llaman protocolos de detección de portadora.

**1-persistent CSMA**

- **Escuchar el canal:**
  - Cuando una estación quiere transmitir, **escucha** primero el canal para ver si está ocupado o libre.
- **Si el canal está libre:**
  - Si el canal está **inactivo** (sin transmisiones en curso), la estación **transmite** sus datos de inmediato. Esto es lo que se llama "persistente-1", porque la estación actúa con **probabilidad 1** de transmitir si el canal está libre.
- **Si el canal está ocupado:**
  - Si el canal está **ocupado** (otra estación está transmitiendo), la estación **espera** a que se libere el canal. Una vez libre, la estación transmite.
- **Colisiones:**
  - Si dos estaciones intentan transmitir al mismo tiempo, se **produce una colisión** .
  - Después de la colisión, cada estación espera un tiempo aleatorio antes de volver a intentar transmitir, lo que se conoce como **backoff aleatorio** .

**CSMA no persistente**

El protocolo **CSMA no persistente** es una variación del protocolo **CSMA persistente-1** , y se introduce para reducir la probabilidad de colisiones al hacer que las estaciones sean menos "impulsivas" y más "pacientes". Aquí están los detalles clave:

**Funcionamiento del CSMA no persistente:**

1. **Escuchar el canal:**
   - Al igual que en **CSMA persistente-1** , cuando una estación tiene datos para transmitir, **escucha primero el canal** para ver si está libre.
2. **Si el canal está libre:**
   - Si el canal está **inactivo** (sin transmisión en curso), la estación **transmite** los datos de inmediato, como en CSMA persistente.
3. **Si el canal está ocupado:**
   - Si el canal está **ocupado** , la estación no intentará transmitir de inmediato (como en CSMA persistente-1), sino que **espera un tiempo aleatorio** antes de volver a intentar escuchar el canal.
4. **Reintentos:**
   - Después de esperar, la estación vuelve a escuchar el canal y repite el proceso. Si el canal sigue ocupado, espera nuevamente y repite el algoritmo, siempre eligiendo un tiempo de espera aleatorio cada vez.

#### Ventajas sobre CSMA persistente-1:

- **Menos colisiones:**
  - El **CSMA no persistente** reduce la probabilidad de colisiones simultáneas. Como las estaciones no transmiten inmediatamente cuando el canal se libera, hay menos posibilidades de que varias estaciones comiencen a transmitir al mismo tiempo.
- **Mejor uso del canal:**
  - Al esperar aleatoriamente, se **distribuyen** mejor las transmisiones a lo largo del tiempo, lo que **optimiza** el uso del canal y evita que las estaciones se "pongan en fila" para transmitir inmediatamente.

#### Desventajas:

- **Mayor retardo:**
  - Como las estaciones esperan un tiempo aleatorio para retransmitir cuando el canal está ocupado, este protocolo introduce **mayores retardos** en comparación con el **CSMA persistente-1** , donde las estaciones transmiten inmediatamente si el canal está libre.
- **Eficiencia reducida en cargas altas:**
  - Aunque el CSMA no persistente ayuda a evitar las colisiones simultáneas, los **retrasos adicionales** pueden hacer que el canal se use de forma menos eficiente en comparación con otros protocolos como el **ALOHA ranurado** o incluso el **CSMA persistente** en cargas bajas.

### CSMA persistente-p

#### Funcionamiento del CSMA persistente-p:

1. **Escuchar el canal:**
   - Cuando una estación está lista para transmitir, primero **escucha el canal** para ver si está libre.
2. **Si el canal está libre:**
   - Si el canal está **inactivo** , la estación no transmite inmediatamente, sino que lo hace con una **probabilidad p** . Con una probabilidad **q = 1 - p** , la estación decide **posponer la transmisión** hasta la siguiente ranura.
3. **Si el canal está ocupado:**
   - Si el canal está ocupado, la estación **espera hasta la siguiente ranura** y luego aplica el mismo proceso de probabilidades **p** (transmitir) y **q**

(posponer).

4. **Reintentos:**
    - Este proceso de decidir si transmitir o posponer se repite en cada ranura, hasta que la estación consiga transmitir exitosamente, o hasta que otra estación comience a transmitir. En caso de que se detecte que otra estación ha comenzado a transmitir, se **actúa como si hubiera ocurrido una colisión** : la estación espera un tiempo aleatorio y comienza el proceso nuevamente.
5. **Refinamiento en IEEE 802.11:**
    - El estándar **IEEE 802.11** para redes Wi-Fi emplea una versión refinada de este protocolo para gestionar el acceso al canal compartido, con reglas adicionales para optimizar el rendimiento en redes de alta densidad.

**Ventajas del CSMA persistente-p:**

- **Flexibilidad en la transmisión:**
    - Al permitir que la estación **decida probabilísticamente** si transmitir o esperar, este protocolo introduce un equilibrio entre transmisión inmediata y espera, reduciendo la probabilidad de colisiones en comparación con el **CSMA persistente-1** y **CSMA no persistente** .
- **Menos congestión en canales con tráfico pesado:**
    - Al usar una probabilidad para decidir si transmitir o esperar, se evita que múltiples estaciones intenten transmitir al mismo tiempo, lo cual es más probable en redes congestionadas.

**Desventajas:**

- **Mayor complejidad:**
    - El comportamiento probabilístico introduce **complejidad adicional** en la implementación y el análisis del protocolo, en comparación con el **CSMA persistente-1** más simple.
- **Rendimiento subóptimo a altas cargas:**
    - Si el tráfico en la red es muy alto, el rendimiento puede **decaer** , ya que la probabilidad de que las estaciones sigan esperando a transmitir aumenta, lo que resulta en **bajos niveles de utilización del canal** . Sin embargo, es más eficiente que el **ALOHA puro** o **ALOHA ranurado** en muchos casos.

**Relación con el rendimiento:**

- El **rendimiento del protocolo CSMA persistente-p** es mejor que el ALOHA en situaciones de tráfico moderado. A medida que la carga de tráfico aumenta, el rendimiento mejora, pero también comienza a deteriorarse por las decisiones de esperar o transmitir.

**CSMA/CD (Acceso Múltiple con Detección de Colisiones)**

es una mejora significativa sobre los protocolos CSMA persistente y no persistente, ya que permite que las estaciones **detecten y aborten rápidamente las colisiones** , lo que optimiza el uso del canal y ahorra tiempo y ancho de banda. Este protocolo es especialmente utilizado en **Ethernet** (en redes cableadas) y tiene un funcionamiento clave para mantener la eficiencia en redes compartidas.

**Funcionamiento de CSMA/CD:**

1. **Escucha el canal** : Al igual que en los protocolos anteriores, cuando una estación tiene datos para transmitir, primero escucha el canal para verificar si está libre. Si el canal está libre, comienza a transmitir inmediatamente.
2. **Colisiones y detección** :

- Si dos estaciones transmiten al mismo tiempo, sus señales se **colisionan** . Sin embargo, con el CSMA/CD, las estaciones pueden detectar estas colisiones **de inmediato** al comparar la señal transmitida con la señal recibida en el canal.
- Si una estación detecta que su transmisión ha colisionado, **detiene su transmisión** de inmediato para no seguir ocupando el canal con datos corruptos.

1. **Espera aleatoria** :

- Después de abortar la transmisión, la estación espera un tiempo **aleatorio** (según el algoritmo de retroceso exponencial), y luego intenta transmitir de nuevo. Este mecanismo reduce las probabilidades de que las estaciones vuelvan a colisionar en el siguiente intento.

**Detalles del proceso de contención:**

El protocolo CSMA/CD usa un modelo de **contención** en el que las estaciones compiten por el canal. Este modelo tiene **periodos de contención, transmisión e inactividad** .

- **Periodo de contención** : Cuando una estación quiere transmitir, primero verifica si el canal está libre. Si está ocupado, espera.
- **Periodo de transmisión** : Cuando el canal está libre, la estación transmite. Si ocurre una colisión, la estación aborta la transmisión y pasa al siguiente **periodo de contención** .
- **Periodo de inactividad** : Si no hay datos para enviar, las estaciones permanecen inactivas.

**Detección de colisiones:**

El tiempo que tarda una estación en **detectar una colisión** depende del **tiempo de propagación** de la señal entre las estaciones.

- En un **escenario de colisión** :
    - Supongamos que dos estaciones comienzan a transmitir al mismo tiempo.
    - La estación que está más cerca de la colisión detecta la interferencia casi de inmediato.
    - Sin embargo, la estación más distante no detecta la colisión hasta que la **señal de la colisión** regresa a ella, lo que puede tomar hasta **2τ** (donde τ es el tiempo de propagación).

  Este **tiempo de propagación** influye en la longitud mínima del **periodo de contención** y es crucial para calcular el **ancho de las ranuras de contención** .

**Comparación con ALOHA ranurado:**

El **CSMA/CD** se comporta de manera similar al **ALOHA ranurado** , pero con una diferencia significativa: en el CSMA/CD, las estaciones **detectan y abortan las colisiones** lo que mejora la eficiencia en comparación con el ALOHA ranurado, que solo permite esperar o retransmitir después de una colisión sin un mecanismo inmediato de detección.

**Ventajas :**

1. **Reducción de tiempo perdido** : La capacidad de detectar colisiones en tiempo real reduce el tiempo desperdiciado por transmisiones corruptas.
2. **Mayor eficiencia que ALOHA** : Al permitir que las estaciones abandonen las transmisiones de manera inmediata tras una colisión, se mejora el uso del canal

comparado con ALOHA, donde se sigue transmitiendo hasta completar la trama y luego se detecta la colisión.
3. **Prácticamente en tiempo real** : A medida que la estación transmite y escucha el canal, las colisiones se detectan rápidamente, lo que permite reintentos rápidos y más eficientes.

**Desventajas :**

1. **No aplicable a redes inalámbricas** : En redes inalámbricas, la **detección de colisiones** es mucho más difícil debido a la diferencia de potencia entre la señal transmitida y la señal recibida. Las estaciones no siempre pueden escuchar lo suficientemente bien el canal mientras están transmitiendo.
2. **Interferencia y colisiones aumentadas en redes grandes** : En redes grandes con muchos dispositivos, las colisiones y los intentos de retransmisión pueden aumentar, lo que lleva a una menor eficiencia si no se controla adecuadamente el **tiempo de contención** .

# Protocolos libres de colisiones

**Protocolo de mapa de bit**

- **Definición inicial:**
  - Hay **N estaciones** conectadas al canal.
  - Cada estación tiene un número único del 0 al **N−1**.
- **Ranuras de contención:**
  - El tiempo se divide en **N ranuras de contención** , una para cada estación.
  - Durante la ranura **j**, **sólo la estación j** puede transmitir un bit.
    - Si la estación tiene algo que enviar, transmite un **1** .
    - Si no tiene nada que enviar, transmite un **0** .
  - **Resultado:** Después de estas **N** ranuras, todas las estaciones saben cuáles quieren transmitir.
- **Transmisión ordenada:**
  - Las estaciones que quieren transmitir lo hacen **en orden numérico** (0, 1, 2, ..., N-1).
  - No hay colisiones porque ya se reservó el canal durante el mapa de bits.
- **Reinicio:**
  - Una vez que todas las estaciones que tenían algo para enviar han terminado, se repite el ciclo: comienza otro periodo de contención de **N**

ranuras.

**Paso de token**

El **protocolo de paso de token** es una técnica utilizada en redes para organizar el acceso al canal de comunicación de forma ordenada y sin colisiones. Funciona mediante el uso de un mensaje especial llamado **token** , que se pasa de una estación a otra en un orden predefinido. El token representa el permiso para enviar. Si una estación tiene una trama puesta en cola para transmitirla cuando recibe el token, puede enviar esa trama antes de pasar el token a la siguiente estación. Si no tiene una trama puesta en cola, simplemente pasa el token.

**Conteo Ascendente binario**

El **conteo descendente binario** es un protocolo utilizado para resolver la contención por el canal en redes con múltiples estaciones. Su objetivo principal es seleccionar una estación ganadora de manera eficiente, utilizando direcciones binarias y un canal que combina las transmisiones de las estaciones (OR booleano). Para ello se establece direcciones binarias para las estaciones y el ganador en cada ronda es el de mayor direccion. La eficiencia de canal de este metodo es $\frac{d}{d+log_2 N}$.

# Contencion limitada

Usa lo mejor de los protocolos de contencion como el CSMA y los protoclos libres de colision.

**Protocolo de recorrido de arbol adaptable**

El **protocolo de recorrido de árbol adaptable** mejora el acceso al canal en redes de difusión al dividir las estaciones en subgrupos organizados en un **árbol binario** . Cada nodo del árbol corresponde a una **ranura de contención** , y las estaciones compiten por el canal según su ubicación en el árbol. Si hay una **colisión** , se examinan los nodos inferiores recursivamente para identificar qué estaciones están listas para transmitir. La búsqueda comienza más abajo en el árbol cuando hay una alta carga en el sistema, optimizando la contención.

**Mejoras clave** :

1. **Búsqueda adaptativa** : Empieza en el nivel adecuado del árbol según el número de estaciones.
2. **Optimización** : Se omiten nodos inactivos para reducir la sobrecarga.

El protocolo reduce las colisiones y mejora la eficiencia, especialmente en redes grandes.

# Protocolos de LAN inalambrica

El protocolo **MACA (Multiple Access with Collision Avoidance)** se usa para evitar colisiones en redes inalámbricas (como Wi-Fi) donde las estaciones (como computadoras o dispositivos) comparten el mismo canal de comunicación.

**Problemas que resuelve :**

1. **Terminal Oculta** : Si una estación A está enviando datos a una estación B, y otra estación C está fuera del alcance de A pero cerca de B, C podría empezar a transmitir sin saber que A ya está enviando datos a B. Esto causaría una colisión.
2. **Terminal Expuesta** : Si una estación C está cerca de B pero no de A, podría escuchar que B está transmitiendo y decidir no enviar datos a D, aunque no haya ninguna interferencia entre C y D.

**Cómo funciona MACA :**

1. **RTS (Request to Send)** : A envía una pequeña señal a B (RTS) para decirle que quiere empezar a enviar datos.
2. **CTS (Clear to Send)** : B responde con un mensaje (CTS), diciéndole a A que puede enviar los datos. Este mensaje también informa a otras estaciones cercanas que deben esperar.
3. **Estaciones cercanas** :

- Las estaciones que escuchan la **RTS** de A y la **CTS** de B saben que no deben transmitir mientras A está enviando datos a B.
- Las estaciones que solo escuchan la **RTS** de A pueden transmitir sin problema, siempre que no interfieran con la señal de B.

**Colisiones :**

Si dos estaciones intentan enviar la **RTS** al mismo tiempo, habrá una colisión. Después de eso, ambas estaciones esperan un tiempo aleatorio antes de intentarlo de nuevo.

**En resumen :**

- **MACA** es un protocolo para coordinar las transmisiones entre estaciones en una red inalámbrica, asegurando que no interfieran entre sí y evitando colisiones.
- Usa los mensajes RTS y CTS para avisar a las estaciones cercanas de que una estación va a transmitir y que deben esperar.

# Dispositivos

- **Repetidores** (capa fisica): Amplificar y reevniar las sennales fisicas.
- **Hub** (capa fisica): Conecta multiples dispositivos en una red de are local(LAN) . Opera en la capa física, y simplemente retransmite los datos a todos los dispositivos conectados sin tomar decisiones inteligentes.
- **Cables (Ethernet, fibra, medios inalambricos de capa fisica)**
- **Modems :** Convierte las sennales digitales a analogicas y viceversa. Esto es crucial en conexiones a Internet mediante líneas telefónicas o cable coaxial (por ejemplo, ADSL).
- **Antenas WIFI :** Los sistemas de acceso permiten la transmisión de señales de radio entre los dispositivos y la infraestructura de red. Las antenas amplían el rango de cobertura de una red inalámbrica.
- **Switches** (capa de enlace): conecta varios dispositivos dentro de una red local (LAN). Opera en la capa de enlace de datos. A diferencia de un hub, un switch puede leer las direcciones MAC (Media Access Control) de los dispositivos y enviar los datos solo al dispositivo correspondiente
- **Bridge (puente)** :**Función** : El puente conecta dos segmentos de una red y opera en la capa de enlace de datos. Puede filtrar el tráfico entre los segmentos según las direcciones MAC y segmentar la red para reducir el tráfico.. Se usa para dividir una red en segmentos más pequeños y mejorar la eficiencia.
- **NIC (Tarjeta de interfaz de red):** La tarjeta de red es un dispositivo que conecta un dispositivo final (como una computadora) a la red. Opera en la capa de enlace de datos y se encarga de la comunicación de la computadora con la red. Se instala en las computadoras y otros dispositivos finales.

- **Router:** (capa de red) pero tmb interactua con la capa de enlace de datos. Se encarga de dirigir los paquetes de datos entre diferentes redes. Un router examina las direcciones IP y elige la mejor ruta para que los datos lleguen a su destino. Se usa para conectar redes diferentes, como una LAN a Internet o una LAN a otra LAN.