

Redes

Añadir

Modelo osi seba, transporte, stack tcp ip

Unidad 3: Capa de enlace de datos.

<https://www.youtube.com/watch?v=IXIIWBATCG8>

<https://www.youtube.com/watch?v=ITvKVGMWtWo>

CUESTIONES DE DISEÑO DE LA CAPA DE ENLACE DE DATOS

En esta unidad se estudian los **principios de diseño de la capa 2 o capa de enlace de datos**, que permite una **comunicación confiable y eficiente entre máquinas adyacentes** conectadas por un canal (cable o conexión inalámbrica). Aunque parezca simple, factores como **errores de transmisión, tasa de datos limitada y retardos** complican la transferencia. Por eso, se necesitan **protocolos** que aborden estos problemas.

Se analizarán:

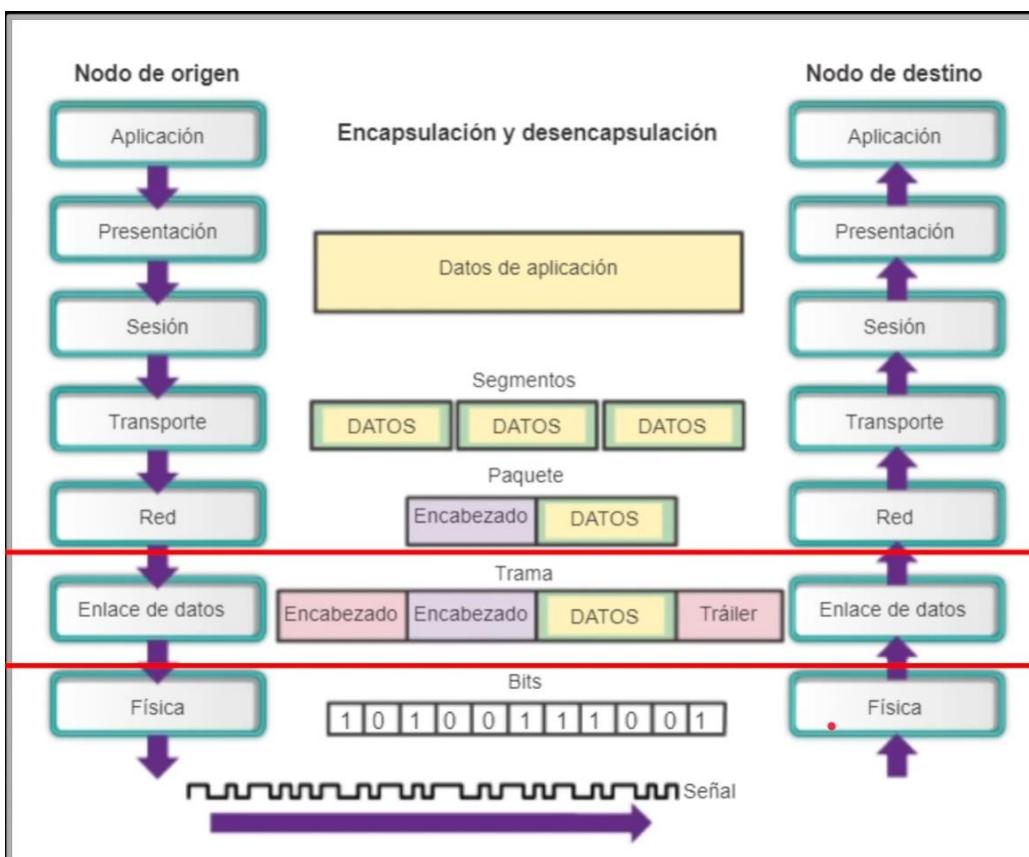
- **Errores:** causas, detección y corrección
- **Protocolos** de complejidad creciente
- **Modelado y verificación** de protocolos
- **Ejemplos concretos**

Cosas a tener en cuenta: Una **trama** tiene varias partes:

- **Encabezado (header):** información de control (como dirección, tipo, longitud, etc.)
- **Payload:** los datos que queremos enviar (texto, archivo, comandos, etc.)
- **Finalizador (footer):** información extra, como un código de verificación (CRC)

Movimiento de los datos en las distintas capas.

Capa OSI	Unidad de Datos	Ejemplo común
7. Aplicación	Datos	Texto de un mensaje, un archivo
6. Presentación	Datos	Datos codificados (por ejemplo, JPEG)
5. Sesión	Datos	Control de sesión entre apps
4. Transporte	Segmento (TCP) / Datagrama (UDP)	Puerto origen/destino, control de errores
3. Red	Paquete	IP origen/destino
2. Enlace de datos	Trama	MAC origen/destino, CRC
1. Física	Bits	1s y 0s transmitidos por cable o aire



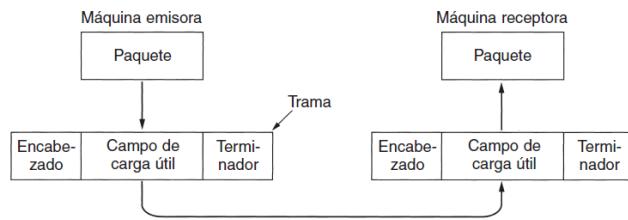
Funciones clave:

1. **Interfaz de servicio bien definida con la capa de red** (ofrece servicios orientados o no a conexión.)

En cuanto al entramado: Construye tramas según el formato de protocolo que use para:

2. **Manejo de errores de transmisión**
3. **Control de flujo** entre emisor y receptor para evitar saturación.

Para ello, **encapsula paquetes en tramas**, compuestas por **encabezado**, **carga útil** y **final**. Osea, delimita el inicio y fin.



Aunque funciones como **control de errores y de flujo** también se ven en capas superiores (como la de transporte), en la capa de enlace se presentan en su forma **más simple y clara**, siendo ideal para su estudio.

Servicios brindados a la capa de red.

La **función principal** de la capa de enlace de datos es:

Transferir datos de la **capa de red** de la máquina de origen a la **capa de red** de la máquina de destino.

Aunque la transmisión real involucra el medio físico, conceptualmente se considera que **dos procesos de la capa de enlace** se comunican entre sí usando un **protocolo de enlace de datos**.

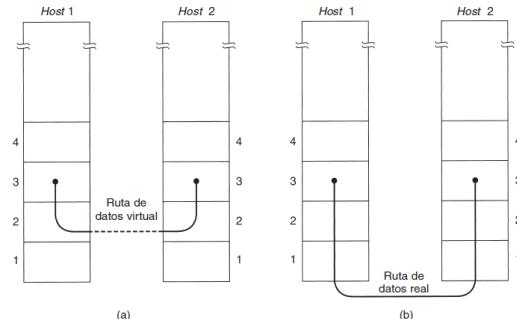


Figura 3-2. (a) Comunicación virtual. (b) Comunicación real.

Tipos de servicios que puede ofrecer

La capa de enlace puede diseñarse para ofrecer **diferentes niveles de confiabilidad**. Los tres principales son:

1. Servicio no orientado a la conexión sin confirmación

- **No hay confirmación de recepción.**
- **No se establece ni se cierra conexión.**
- **Simple y rápido**, pero poco confiable.
- Si se pierde una trama, **no se recupera**.
- Útil cuando:
 - o La tasa de errores es baja.
 - o En tráfico en **tiempo real** (como voz), donde el **retraso es peor que los errores**.
- Muy común en **LANs**.
- **Ejemplo:** Sin confirmación LLC (Logical Link Control) y PPP (Point-to-Point)

2. Servicio no orientado a la conexión con confirmación

- **Sin conexión lógica**. Una conexión lógica es un acuerdo temporal entre dos dispositivos para establecer una relación de comunicación estructurada. Intercambian mensajes de control (como "holo, estoy listo"). Se numeran los mensajes o tramas. Se sigue el orden. Se mantiene un "estado" (quién ha recibido qué, cuántas tramas, etc.).
- Cada trama tiene **confirmación de recepción** individualmente.
- Si no hay confirmación, se **retransmite** tras un tiempo.
- Útil en **canales inestables**, como los **inalámbricos**.

- **Optimización**, no requisito obligatorio: La capa de red podría encargarse, pero sería más ineficiente.

Uso: en redes con taza de error bajas. Hoy en día las más comunes. UDP

3. Servicio orientado a la conexión con confirmación

Garantiza la entrega confiable y ordenada de tramas:

- **Establece una conexión** antes de transmitir datos.
- Cada trama es recibida **exactamente una vez**.
- Se controla el flujo
- **Tramas numeradas** y recibidas en orden.
- Se entregan a la capa superior ordenadas y libres de errores
- Detecta errores en la transmisión y se descartan tramas erróneas
- Las tramas perdidas o erróneas se retransmiten

Para ello utiliza:

- Técnicas de detección de errores
- Numeración de tramas
- Confirmación de recepción
- Retransmisión de las no confirmadas.

Uso: TCP

¿Qué quiere decir "confirmación de recepción"?

Cuando una computadora (o un router, por ejemplo) **envía datos**, puede recibir una especie de "ok, lo recibí bien" desde el destino. A eso se lo llama **confirmación de recepción (ACK)**.

Si no llega esa confirmación en cierto tiempo, el emisor vuelve a enviar los datos

Paquetes y tramas en Perdidas

La capa de enlace maneja **paquetes**, en ocasiones suelen ser grandes por lo que se dividen en **tramas** que el hardware puede manejar. Si se pierde **una sola trama**, el paquete completo **no se puede reconstruir**. Por otro lado, si se confirma **cada trama** por separado, solo se retransmiten las tramas que se **pierden**, no todo el paquete completo. Esto hace que los datos lleguen **más rápido y con menos trabajo**

Fases del servicio orientado a la conexión:

1. **Establecimiento** de la conexión (inicialización de variables y contadores para seguir la pista de las tramas que han sido recibidas y las que no).
2. **Transmisión** de tramas.
3. **Liberación** de la conexión (libera variables, los búferes y otros recursos).

Uso: en redes con taza de error alta. Hoy en día no se usan tanto.

Ejemplo: Protocolo HDLC (Hight-Level data link control)

¿Cómo funciona un servicio orientado a la conexión?

1. Establecimiento de conexión

El emisor y el receptor se "ponen de acuerdo" para comunicarse. En redes, esto se hace con una **negociación inicial** (como el famoso **handshake** de TCP).

2. Transferencia de datos

Se intercambian datos sabiendo que hay una conexión establecida. Hay **control de errores**, **confirmaciones**, y se asegura que los datos lleguen **en orden y sin duplicados**.

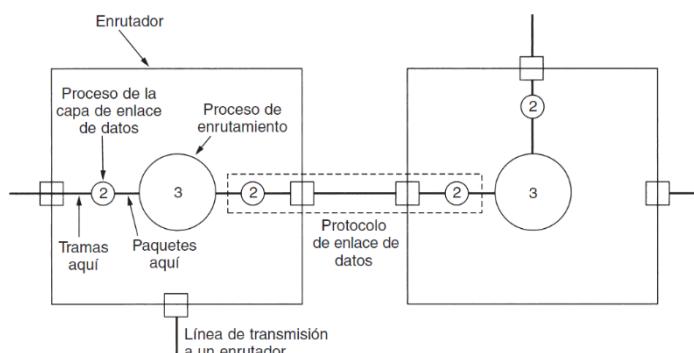
3. Terminación de la conexión

Una vez que se terminó de enviar la información, se **cierre la conexión** de forma ordenada.

Ejemplo típico: Enrutadores en una WAN

- Las tramas llegan a un **enrutador**, el hardware verifica errores.
- Luego, el **software de la capa de enlace** entrega el paquete si todo está bien. Dicho software comprueba si ésta es la trama esperada y, de ser así, entrega el paquete contenido en el campo de carga útil al software de enrutamiento
- La **capa de red** elige la ruta y reenvía el paquete.
- Cada línea punto a punto es manejada como si fuera **confiable**, gracias al protocolo de enlace de datos.

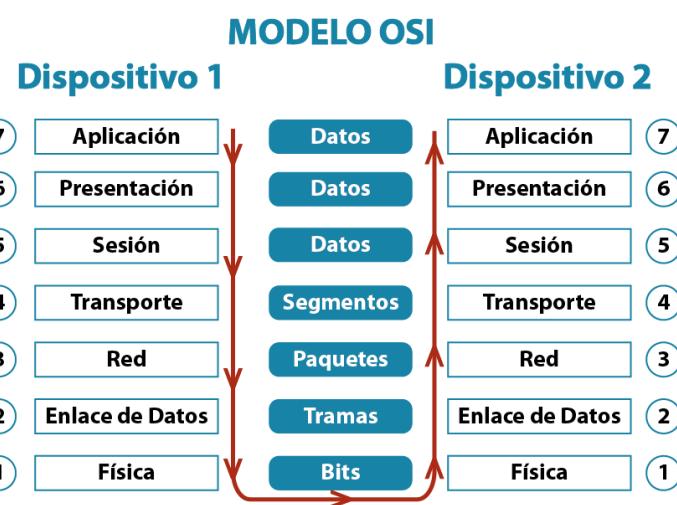
Una única copia del software maneja **todas las líneas**, con estructuras separadas por cada una.



Entramados (Framing) y tipos

<https://www.youtube.com/watch?v=1LG6yUO-loc>

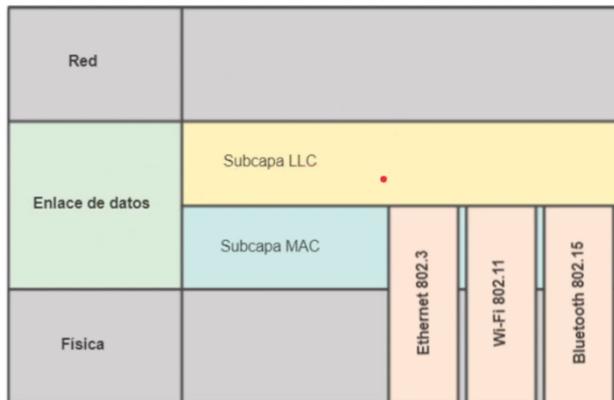
- Acepta datos de la capa 3 y los empaqueta en unidades llamadas tramas.
- Controla el acceso al medio y detecta errores
- La **capa de enlace de datos** transforma el flujo de bits de la **capa física** en **tramas** y **detecta/correge errores** usando **sumas de verificación**. Para separar tramas se usan métodos que **no dependan del tiempo**.



Subcapas:

- **Capa de enlace lógico (LLC):** Se comunica con la capa de red. Transiciona de la parte física a lógica, para poder trabajar con los protocolos de capa tres, como ser IPV6 E IPV4, dándole la información correcta.
- **Capa de acceso al medio (MAC):** más cercano a la capa física, define los procesos de acceso al medio (entramado representado en hardware). Y como se van a delimitar las tramas que usa cada uno de los estándares.

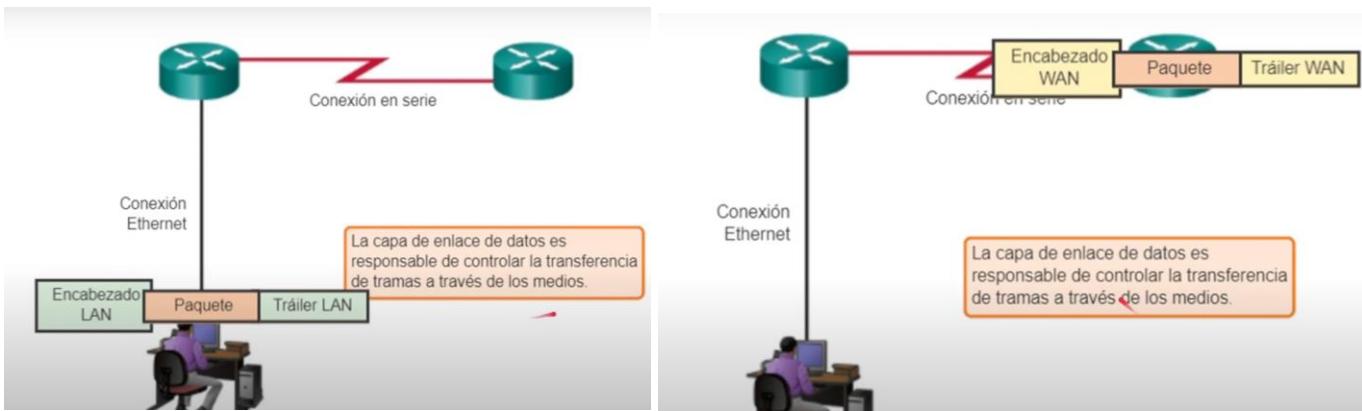
Subcapas de enlace de datos



Estructura de la dirección MAC de Ethernet



La dirección Mac es de 48 bits o 12 dígitos en hexadecimal. Los mas significativos representan a la organización que distribuye o genera los dispositivos. Asignado por el proveedor es el NIC, para representar la dirección específica de ese dispositivo.



Cada dispositivo cambia el encabezado.

Campos de las tramas:



Dada una trama, tenemos tres partes:

- Encabezado:
 - o Inicio de trama: secuencia de bits que determina el inicio de la trama
 - o Direccionamiento: la dirección MAC. Identifica los nodos origen y destino
 - o Tipo: el LLC utiliza este campo para identificar el protocolo de la capa 3.
 - o Control: identifica servicios especiales para el control de flujo.
- Paquete de datos: incluye el contenido de la trama, es decir, el encabezado del paquete, el encabezado del segmento, y los datos.
- Tráiler:
 - o Cod. De detección de errores
 - o Detención o fin de trama

7 bytes	1 byte	6 bytes	6 bytes	2 bytes	42 to 1500 bytes	4 bytes	12 bytes
Preamble	Start of Frame Delimiter	Destination MAC Address	Source MAC Address	Type	Data (payload)	CRC	Inter-frame gap

For TCP/IP communications,
the payload for a frame is a
packet

WiFi (802.11) Frame Format								
2 bytes	2 bytes	6 bytes	6 bytes	6 bytes	2 bytes	6 bytes	0 to 2312 bytes	4 bytes
Frame Control	Duration	MAC Address 1 (Destination)	MAC Address 2 (Source)	MAC Address 3 (Router)	Seq Control	MAC Address 4 (AP)	Data (payload)	CRC

En el caso de el WIFI, se necesita mas información. Tiene como particularidad que tiene la MAC, destino, origen, del router y ACP.

Métodos de entramado:

<https://www.youtube.com/watch?v=1LG6yUO-loc>

1. **Conteo de caracteres:** El primer byte indica cuantos debo contar para delimitar la trama. Debo incluirlo en el conteo.

Error común: si la cuenta se corrompe (causados por ruido o interferencias en el canal de comunicación), se pierde sincronía. Para delimitar tramas es poco confiable, ya que un error en la cuenta puede desincronizar la comunicación. Aunque se detecte el error, no es posible saber dónde comienza la siguiente trama, lo que impide una correcta retransmisión. Por esta razón, este método ha caído en desuso

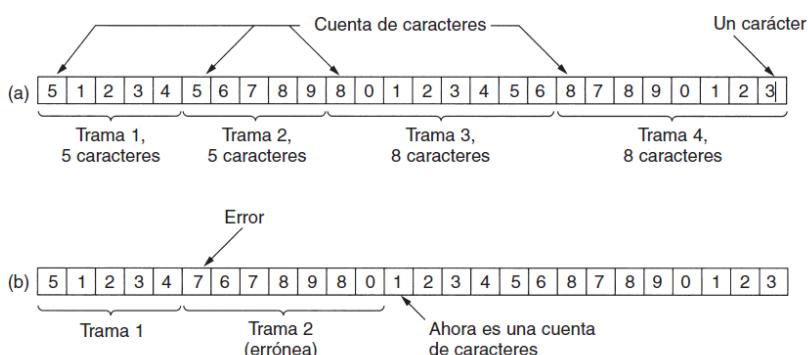


Figura 3-4. Un flujo de caracteres. (a) Sin errores. (b) Con un error.

- El número 5 en binario es 0101.
- El número 7 en binario es 0111.

2. **Rellenos de bytes o Banderas + relleno de caracteres:** tenemos bytes bandera de inicio y fin conocidos como FLAGS, son una secuencia de bits conocidos. Además, rellenamos de bits en el caso que se requiera.

Dos bytes banderas consecutivos señalan el final de la trama y el inicio del siguiente. De esta forma si el receptor pierde alguna vez la sincronización, solo busca dos bytes para encontrar el fin de la trama actual.

Si el patrón de bits FLAG aparece dentro del **campo de carga útil**, el receptor podría pensar que es un delimitador y **romper la trama**. Por ello se **agrega un byte de escape (ESC)**. El receptor, al ver ESC FLAG, sabe que **ese FLAG forma parte de los datos**, no del límite de trama. **Si aparece un ESC en los datos**, se escapa de la misma forma: cada ESC real se envía como **ESC ESC**. Así, siempre que veas un solo ESC → indica que el siguiente byte está **escapado**.

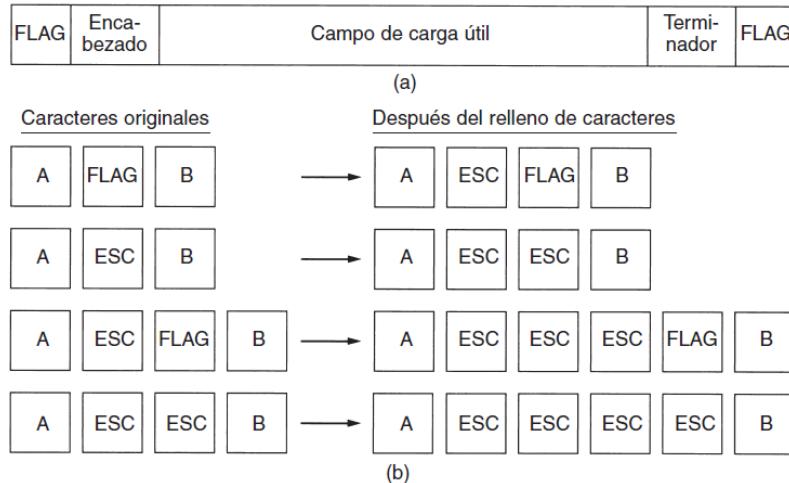


Figura 3-5. (a) Una trama delimitada por banderas. (b) Cuatro ejemplos de secuencias de bytes antes y después del relleno de caracteres.

Entonces:

- Con ESC ESC sé que el segundo es parte de los datos
- Con ESC FLAG sé que el segundo forma parte de los datos
- Si tuviese dos ESC, uso cuatro, los primeros avisan que el que sigue es dato.

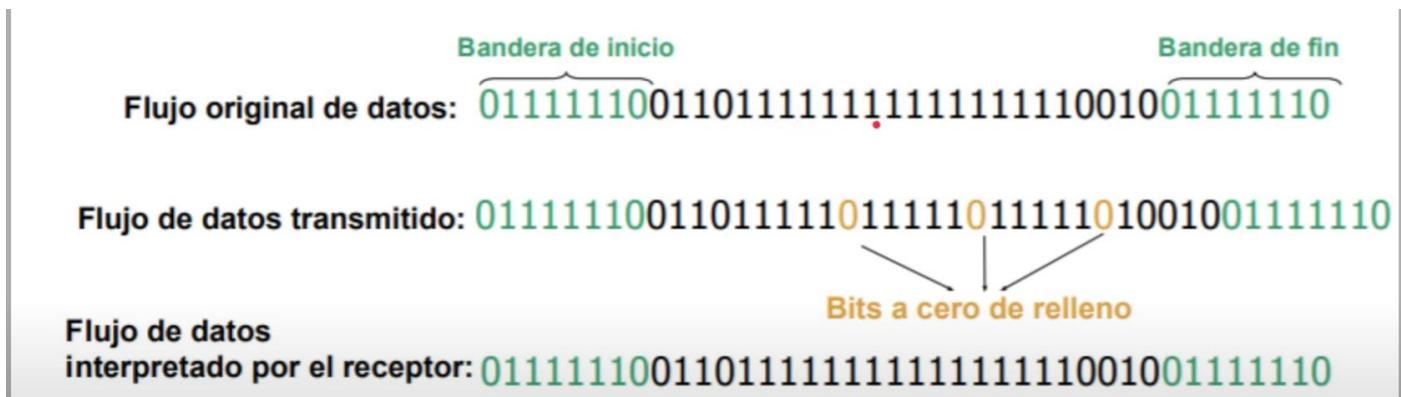
Ventaja principal: Nunca pierdo la sincronía: si pierdo el conteo, solo busco el próximo FLAG válido (no escapado) y retomo desde ahí.

Desventaja: Está ligado a **códigos de 8 bits**, no sirve para otros. Usa caracteres que requieren estandarización. Además hay sobrecarga de información cuando hay casos especiales. Esto hace que se hagan muy largos.

3. Banderas + relleno de bits:

- salva al algoritmo anterior.
- Es mas eficiente porque solo rellena con un solo bit en vez de bytes.
- Cada trama comienza y termina con **01111110 (patrón de banderas de seis unos)**.
- Problema: Si ese patrón **aparece en los datos** El receptor se confundiría, pensando que es el fin (o inicio) de la trama.
- **Permite tamaños arbitrarios** de bits.

Relleno de bits (bit-stuffing), Regla: Siempre que el emisor vea cinco “1” seguidos (11111) en los datos, inserta un “0” extra inmediatamente después. Al recibir, cada vez que el receptor ve 111110, elimina ese “0” de relleno. Evitando confundirlo con una bandera



<https://www.youtube.com/watch?v=pOW1B4N8Ubk&t=2s>

- Violaciones de codificación: se usan **secuencias inválidas o imposibles** del código físico como delimitadores de tramas. Aplicable sólo si el medio tiene **redundancia** en su codificación.

La “redundancia” acá significa que el esquema de codificación físico **genera más formas de señal** de las estrictamente necesarias para representar los datos. Es decir:

Codificación expandidora

- Por ejemplo, en **4B/5B**: cada bloque de **4 bits de datos** se convierte en **5 bits físicos**.
- Hay $2^4=16$ combinaciones de datos posibles, pero $2^5=32$ combinaciones físicas disponibles.
- Sobran 16 códigos que nunca** se usan para datos reales → ¡esos son los “ilegales”!

Muchos protocolos combinan **conteo + delimitador + verificación** para mayor seguridad.

Control de errores

- **Detecta errores de transmisión**
- **Retransmite toda trama perdida o errónea**

Una vez que se resuelve el problema de marcar el **inicio y fin de cada trama**, se debe asegurar que las tramas lleguen en **orden correcto** a la **capa de red del destino**.

En un servicio **confiable y orientado a conexión**, se necesita **retroalimentación** del receptor. Esto se hace mediante **tramas de control** que contienen **confirmaciones de recepción**:

- **Positivas**: indican que la trama llegó bien.
- **Negativas**: indican que algo falló y debe **retransmitirse**.

Surge un problema cuando se **pierde una trama** o su **confirmación**, por ejemplo, por una **falla de hardware**. En este caso, ninguna de las partes sabe qué pasó.

Por ejemplo, en una ráfaga de ruido, se pierde una trama completa. En este caso, el receptor no reaccionará en absoluto, ya que no tiene razón para reaccionar. De manera similar, si se pierde la trama de confirmación de recepción, el emisor no sabrá cómo proceder. Se quedaría esperando eternamente si se perdiera por completo una trama debido a, por ejemplo, una falla de hardware o un canal de comunicación defectuoso.

Para manejar esto, se usan **temporizadores**. Cuando el emisor envía una trama, inicia un temporizador. Si la **confirmación** llega antes de que expire, se **cancela**. Si no llega, el **emisor retransmite** la trama.

Para evitar que el receptor acepte la **misma trama varias veces**, se usan **números de secuencia**, que permiten identificar si una trama es **nueva o repetida**.

La gestión de **temporizadores** y **números de secuencia** es esencial para que **cada trama llegue una sola vez**, y forma parte clave del trabajo de la **capa de enlace de datos**.

Control de flujo.

- **Controla el flujo de transmisión para emisores rápidos y receptores lentos.**

El **control de flujo** es un mecanismo esencial en la **capa de enlace de datos** (y también en capas superiores) para evitar que un **emisor rápido** sature a un **receptor lento** con demasiadas tramas, **sobrecarga**. Por ejemplo, puede ocurrir cuando un **servidor potente** envía muchos datos a un **dispositivo móvil**, generando pérdidas, aunque no haya errores de transmisión.

Métodos principales de control de flujo:

1. Basado en retroalimentación:

El **receptor** informa al **emisor** cuándo puede seguir enviando datos o cuál es su estado.

Ejemplo: "Podés mandarme n tramas, pero no más hasta que te lo indique".

2. Basado en tasa:

Se fija una **velocidad máxima de transmisión** para el emisor, **sin necesidad de retroalimentación** del receptor.

(Este tipo se trata más en la **capa de transporte**).

Implementaciones prácticas:

- Muchas veces el **hardware de la capa de enlace**, como las **NIC** (Tarjetas de Interfaz de Red), se diseña para operar a "**velocidad de alambre**", es decir, tan rápido como los datos pueden llegar, evitando así pérdidas en esta capa.
- En esos casos, el control de flujo se maneja en **capas superiores**, como la **capa de transporte**.

DETECCIÓN Y CORRECCIÓN DE ERRORES

<https://www.youtube.com/watch?v=ICQzmPySkUQ>

En los canales de comunicación, los **errores de transmisión** son inevitables en ciertos entornos (p. ej., enlaces inalámbricos). Para **manejar** estos errores, se usan dos **estrategias** que añaden **redundancia** a los datos enviados:

1. Códigos de corrección de errores (FEC)

- Incluyen **suficiente información redundante** para que el **receptor reconstruya** los datos originales sin pedir retransmisión.
- Útiles en **canales ruidosos** donde las **retransmisiones** tienen alta probabilidad de fallar igual que el bloque inicial.
- Utiliza **códigos de corrección de errores, FEC (Corrección de Errores hacia Adelante, del inglés Forward Error Correction)**.

2. Códigos de detección de errores

- Añaden **sólo la redundancia necesaria** para que el receptor sepa que hubo un error (pero no sabe cual), y entonces **solicite retransmisión** del bloque defectuoso.
- Óptimos en **canales confiables** (p. ej., fibra óptica) donde los errores son raros. Mas economico
- usa **códigos de detección de errores**

Consideraciones clave:

- Los **bits de datos** y los **bits redundantes** sufren la misma tasa de error; por eso el código debe ser **robusto** frente a los tipos de errores esperados.
- Se distinguen dos **modelos de error**:
 - **Errores aislados** (daña un solo bit): valores extremos de ruido térmico que satura la señal breve y ocasionalmente.
 - **Errores en ráfaga** (múltiples bits consecutivos), más difíciles de corregir, pero con ventajas estadísticas en bloques grandes. deriva de los procesos físicos que los generan (como un desvanecimiento pronunciado en un canal inalámbrico o una interferencia eléctrica transitoria en un canal alámbrico)
- Existe además el **canal de borrado**, donde se sabe la **ubicación** del bit perdido, lo que **facilita la corrección**. La capa física detecta una señal muy alejada del valor esperado y lo declara como bit perdido. **No** es fácil cuando el canal volteá un bit (cambia un bit por otro), ya que no nos damos cuenta.

Ámbito de aplicación:

- Los **códigos de corrección** se usan en la **capa de enlace**, en la **capa física** (para canales ruidosos) y en **capas superiores** (medios de tiempo real, distribución de contenido).
- Los **códigos de detección** se emplean frecuentemente en las **capas de enlace, red y transporte**.

Por último, estos **códigos** se basan en **matemáticas aplicadas** (teoría de códigos, campos de Galois). En la práctica, conviene usar **estándares probados** en lugar de diseñar desde cero. Más adelante se estudiará en detalle un **código simple** y luego se describirán los **códigos avanzados** utilizados en la industria.

Redundancia

Esencialmente, son **bits adicionales** que no forman parte del mensaje “útil”, pero que permiten al receptor:

- **Verificar** si los datos llegaron bien (códigos de detección, p. ej. sumas de comprobación o CRC).
- **Reconstruir** los datos correctos cuando hay errores (códigos de corrección, p. ej. bits de paridad en Hamming).

Códigos de corrección de errores

Códigos de bloque lineales sistemáticos (n, m):

- Cada **trama** lleva **m bits de datos + r bits de redundancia** ($n = m + r$). n es la longitud total del bloque, describiremos esto como un código (n, m) . Se conoce como **palabra codificada**
- La **tasa de código** es m/n . Las tasas que se utilizan en la práctica varían mucho. Podrían ser $1/2$ para un canal ruidoso, en cuyo caso la mitad de la información recibida es redundante, o podrían estar cerca de 1 para un canal de alta calidad, en donde sólo se agrega un pequeño número de bits de verificación a un mensaje extenso.

- Los **bits de verificación** se calculan como una función **lineal** (XOR) o la suma de módulo 2, de los bits de datos. Es posible determinar cuántos bits correspondientes difieren. Para determinar la cantidad de bits diferentes, basta aplicar un XOR a las dos palabras codificadas y contar la cantidad de bits 1 en el resultado, por ejemplo:

$$\begin{array}{r} 10001001 \\ 10110001 \\ \hline 00111000 \end{array}$$

Distancia de Hamming (d):

<https://www.youtube.com/watch?v=-jlofQSr1gs>

Número de bits diferentes entre dos palabras codificadas. Es la **cantidad de bits**, que se deben cambiar para ir de una combinación invalida a una **valida**

Detección de errores

Para una distancia haming dada d_h min, nosotros podemos detectar d (**nro de bits errados**) que es uno menos que esa distancia d_h min.

$$D_{\text{min}} = D + 1$$

Corrección de errores

Si solo quiero **corregir c bits**, mi distancia haming queda:

$$D_{\text{min}} = 2c + 1$$

Es decir, separar las combinaciones validas una cantidad impar de veces

- No puedo asociar: V-I-V
- Puedo asociar, pero no corregir con exactitud: V-I-I-V -> (v-i)-(i-v)
- Puedo asociar y corregir si está más cerca de una: V-I-I-I-V -> (v-i)-i-(i-v)

Detección y correcciones:

Para **detectar y corregir** de forma genérica c bits, **detectar d** bits errados (adicionales a los que ya se detectan)

$$D_{\text{min}} = 2c + d + 1$$

Entonces si yo quisiera detectar y corregir errores de un bit $c=1$, y no detectar errores adicionales $d=0$. Ósea, solo se puede errar un bit. No quiero detectar bits adicionales, entonces $D_{\text{min}}=3$. Si quisiese detectar bits adicionales, $d=1$, sería capaz de detectar errores de hasta dos bits, y corregir errores de uno, entonces $D_{\text{min}}=4$

HAMMING DISTANCIA 1		
Mensaje		
d0	d1	
0	0	VALIDO
0	1	VALIDO
1	0	VALIDO
1	1	VALIDO

No detecta ni corrige

HAMMING DISTANCIA 2		
Mensaje		
d0	P0	
0	0	VALIDO
0	1	INVALIDO
1	0	INVALIDO
1	1	VALIDO

Agrego un bit de paridad, hay dos combinaciones validas. No puedo detectar errores de dos bits, porque sino salto de una valida a otra valida. Puedo detectar errores de un bit. No puedo corregir, porque necesitaría una combinación invalida más, en medio de las dos, para que me separe.

HAMMING DISTANCIA 3			
Mensaje			
d0	P0	P1	
0	0	0	VALIDO
0	0	1	INVALIDO
0	1	0	INVALIDO
0	1	1	INVALIDO
1	0	0	INVALIDO
1	0	1	INVALIDO
1	1	0	INVALIDO
1	1	1	VALIDO

Puedo detectar errores de hasta dos bits, y corregimos errores de hasta un bit. Todas las combinaciones que distan un bit errado de la valida, la podemos asociar a la valida más próxima.

Reglas de haming:

El número de bits de información, más los números de bits de paridad (que queremos determinar), más uno, debe ser menor o igual que dos elevado a p (nro de bits de paridad)

$$2^p \geq I + P + 1$$

Y resolvemos la inecuación.

- A cada bit de paridad, asociamos a uno o más conjuntos de bits de información.
- Cada bit de información tiene que pertenecer a **más de un bit** de paridad como mínimo.
- Y dos bits de información no pueden compartir un bit de paridad si es que es el **único** bit de paridad asociado.

Ejemplo: uso un código de dos bits y lo quiero expandir. Remplazo los datos en la inecuación y resuelvo, vamos probando valores de p hasta encontrar el primer valor que cumpla. Entonces agregamos tres bits de paridad, así tenemos un mecanismo que nos permite corregir. Luego asociamos bits, en la imagen podemos notar que p2 es compartido por d0 y d1. Pero no es el único grupo que tiene.

Mensaje	CODIGO		HAMMING DISEÑO				
	d0	d1	CODIGO		PARIDAD HAMMING		
0	0		X	-	X	X	X
0	1		X	-	X	X	X
1	0						
1	1						

Redundancia

- De los 2^n posibles patrones de n bits, solo 2^m (con $m < n$) son válidos: la **redundancia** ($r = n - m$ bits) los separa en el espacio, dejando “huecos” que ayudan a detectar y corregir errores.

- La **fracción** de palabras válidas es $2^m / 2^n = 1/2^r$: cuanta más redundancia (r grande), más separación y mejor detección/corrección, pero menos eficiencia en datos útiles.

Códigos de Hamming

- <https://www.youtube.com/watch?v=gQK9nROFX20>
 - <https://www.youtube.com/watch?v=ay9LRC4ldpM>
 - Bits en posiciones de potencia de 2 (1, 2, 4, 8...) son **de paridad**; el resto son **datos**.

Primero busco las posiciones pares en la tabla, y los impares lo relleno con los datos. Luego para agregar los bits de paridad, uso el número de posiciones en binario, y veo el bit menos significativo igual a uno. Luego bajamos quienes tengan el segundo bit menos significativo en uno. Para calcular el bit de paridad veo los bits que baje, y si cuento la cantidad de 1 y es par, agrego cero en la posición de paridad. Sino 1 para forzar paridad

Ejemplo: (11,7) p: bit de paridad datos= 0101001
d: bit de datos

	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7
Posición	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)
Palabra original			0		1	0	1		0	0	1
p_1	1		0		1		1		0		1
p_2		0	0			0	1			0	1
p_3				0	1	0	1				
p_4									1	0	1
Palabra+paridad	1	0	0	0	1	0	1	1	0	0	1

- Diseñan cada paridad para que su suma módulo 2 sea par. **Distancia = 3** → corrige 1 bit (o detecta 2).
 - El **síndrome** (resultado de recalcular paridades) indica la posición del bit erróneo. Para ello repetimos el proceso y comparamos. Si interpretamos el error como 1 y el ok como 0 y leemos de abajo a arriba, lo pasamos a decimal, nos da la ubicación del bit de error. Lo cambiamos y listo.

Palabra almacenada = 10001011001
Introducimos un error en un bit = 10001011000

2. Códigos convolucionales binarios

- No hay bloque fijo: la salida depende de los bits **actuales y previos** (memoria).
- Definidos por **tasa** (por ej. 1/2) y **longitud de restricción k**.
- Se decodifican con **algoritmo de Viterbi**, que busca la secuencia de entrada más probable.
- Muy usados en **GSM, satélite y 802.11**.

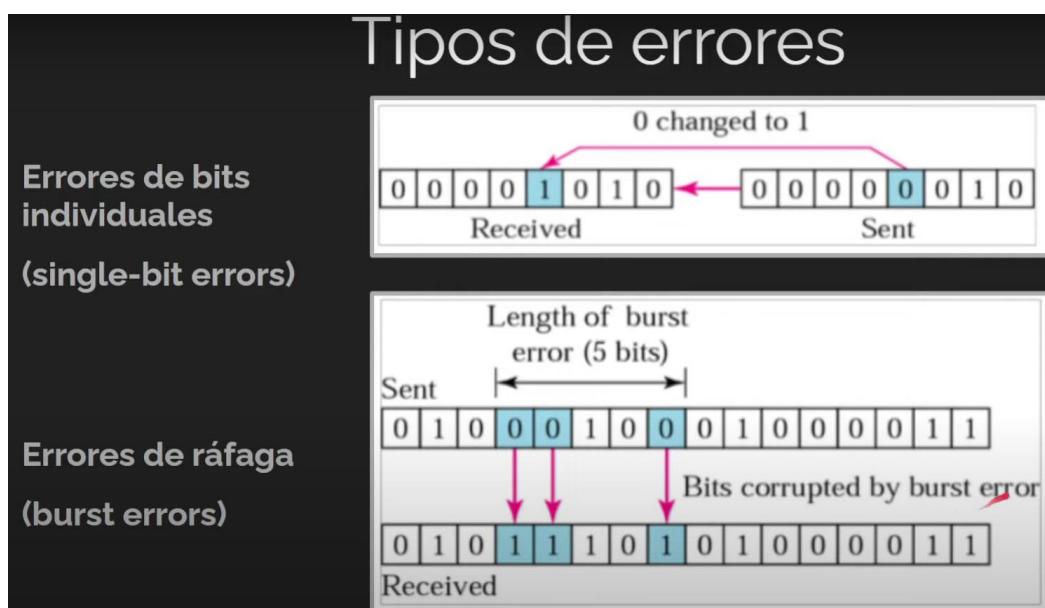
3. Códigos de Reed-Solomon

- **Bloque** sobre **símbolos** de m bits (por ej. bytes).
- Basados en polinomios de grado $\leq n - 1$: basta con n puntos para definirlo.
- Código (255, 233): añade 32 símbolos redundantes \rightarrow corrige hasta 16 símbolos (hasta 128 bits de ráfaga).
- Ideales para **errores en ráfaga** y canales de borrado. Uso en **CD/DVD/Blu-ray, DSL, satélite**.

4. Códigos LDPC (Low-Density Parity-Check)

- Matriz de paridad muy **dispersa** (baja densidad de 1s).
- Cada bit de salida depende de pocos bits de entrada.
- Decodificación **iterativa** que mejora gradualmente el ajuste.
- Excelente rendimiento en bloques grandes; ya estándar en **video digital, Ethernet 10 Gbps, 802.11ax**.

Códigos de detección de errores



1. Paridad

<https://www.youtube.com/watch?v=ICQzmPySkUQ>

<https://www.youtube.com/watch?v=hEhp8qhFU90>

Idea básica

La paridad es el método más simple, y se usa cuando la probabilidad de error es mínima, ósea errores individuales: se agrega **un solo bit** (el bit de paridad) al final de un bloque de datos, de modo que el número total de bits 1 sea par (o impar, según lo acuerdes).

1. Bit de paridad par:

- **Vale 0 si el número de unos en el bloque es par (o cero).**
- **Vale 1 si el número de unos en el bloque es impar.**

2. Bit de paridad impar:

- **Vale 1 si el número de unos en el bloque es par (o cero).**
- **Vale 0 si el número de unos en el bloque es impar.**

Con esto, si llega la palabra y la paridad no concuerda, te das cuenta de que hubo al menos un error de un bit.

Ejemplo sencillo

Ejemplos:	Carácter (7 bits)	Paridad par	Paridad ímpar
	0000000	0	1
	1011010	0	1
	0110100	1	0
	1100111	1	0

Supongamos que tenés la secuencia de datos:

1 0 1 1 0 1 0 (7 bits)

1. Contás cuántos bits 1 hay:

- En “1011010” hay 4 bits 1 (porque $1+0+1+1+0+1+0 = 4$).

2. Si usás **paridad par**, el bit de paridad va a ser 0, porque ya tenés 4 (un número par). Entonces la palabra final es:

3. 1 0 1 1 0 1 0 | 0 → “10110100”

4. Si usás **paridad ímpar**, el bit de paridad va a ser 1, para que queden 5 bits 1 (un número ímpar). Entonces:

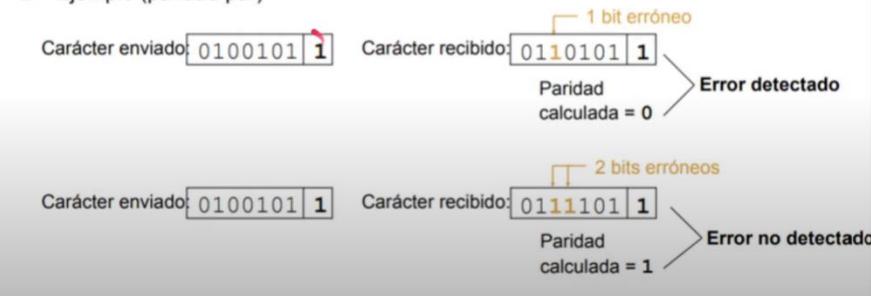
5. 1 0 1 1 0 1 0 | 1 → “10110101”

Detección de errores:

- Si en el canal se invierte un solo bit (por ejemplo, el tercer bit pasa de 1 a 0), la nueva palabra recibida “tiene la paridad equivocada” y lo detectás.
- **Limitación:** si se invierten dos bits en posiciones estratégicas, podría ser que la paridad siga cuadrando y no te des cuenta. Por ejemplo, si en “10110100” se invierten dos bits de 1 a 0 y de 0 a 1, podés terminar con una palabra que siga teniendo un número par de 1, y no detectas nada.

Limitaciones de la paridad

- Los bits de paridad sólo son capaces de detectar los "errores impares"
 - Es decir, aquellos errores que sólo afectan a un número impar de bits.
 - Si el error afecta a un número par de bits, entonces no podrá ser detectado
- Ejemplo (paridad par)



2. Paridad bidimensional (intercalado)

<https://www.youtube.com/watch?v=Wj5ZLtBJ3j8>

Para mejorar la detección cuando hay ráfagas de errores (por ejemplo, varios bits equivocados seguidos), podés organizar los datos en una “matriz” de k filas y n columnas, y calcular paridad por fila y por columna. Sobre todo, si aplicás **intercalado**, los errores consecutivos se dispersan.

Paso a paso

- Elegís n (ancho) y k (alto). Supongamos $n = 4$ y $k = 3$.
- Tenés 12 bits de datos. Por ejemplo, los datos (sin paridad) son:

Fila 1: 1 0 1 1
Fila 2: 0 1 0 0
Fila 3: 1 1 0 1

- Calculás **paridad por fila** (paridad par en este ejemplo):

- Fila 1 (1 0 1 1) → tres bits 1 → paridad = 1 (para que queden 4 en total).
- Fila 2 (0 1 0 0) → un bit 1 → paridad = 1 (para que queden 2).
- Fila 3 (1 1 0 1) → tres bits 1 → paridad = 1 (para que queden 4).

Entonces las filas con su bit extra quedan:

Fila 1: 1 0 1 1 | 1
Fila 2: 0 1 0 0 | 1
Fila 3: 1 1 0 1 | 1

- Ahora calculás **paridad por columna**. Como tenés 4 columnas de datos (sin contar la columna de parity de cada fila), calculás paridad de cada columna “verticalmente”:

- Columna 1 (bits en fila 1,2,3): 1, 0, 1 → hay 2 bits 1 → paridad columna = 0 (para que quede par).
- Columna 2: 0, 1, 1 → hay 2 bits 1 → paridad = 0.
- Columna 3: 1, 0, 0 → hay 1 bit 1 → paridad = 1.
- Columna 4: 1, 0, 1 → hay 2 bits 1 → paridad = 0.

- Armás la “última fila” con esos bits de paridad de columnas:

Paridad columna: 0 0 1 0

Entonces la trama final que transmitís, en **orden normal** (filas de arriba hacia abajo, de izquierda a derecha) y sumándole la fila de paridad al final, es:

1	0	1	1	1
0	1	0	0	1
1	1	0	1	1
0	0	1	0	← esta es la última fila: paridad de cada una de las 4 columnas

En total son 3 filas de datos con 1 bit de paridad cada una (4 bits de paridad horizontal) + 4 bits de paridad vertical = 8 bits de paridad en un bloque de 12 bits de datos.

¿Por qué ayuda contra ráfagas?

Si hay un error de ráfaga de hasta n bits (ej.: 4 bits seguidos), esos bits “caen” en diferentes columnas cuando los enviás fila por fila. Entonces, al recibir, la paridad de alguna(s) columna(s) ya no va a coincidir y detectás el error.

Limitación: si la ráfaga cubre exactamente la medida horizontal y vertical (es decir, invierte el primer bit de cada fila y el último de cada fila de tal manera que la paridad siga “cuadrando”), podría haber casos muy puntuales que no detectás. Pero, en general, mejora muchísimo la detección de ráfagas largas.

3. Sumas de verificación (Checksum)

<https://www.youtube.com/watch?v=zgLWUmofMeE>

Idea básica

En lugar de “contar” bits individuales, agrupás los datos en “palabras” de cierto tamaño (por ejemplo, 16 bits) y hacés la **suma** de esas palabras, normalmente usando aritmética de complemento a uno. Luego ponés el resultado (o su complemento) al final del mensaje. Al llegar, el receptor suma de nuevo todas las palabras (incluida la suma) y, si da “todo 1s” o “todo 0s” (según se diseñe), asume que no hubo errores.

Ventajas sobre paridad

- Detectás mejor algunos errores que paridad simple no ve:
 - Si dos bits de pesos distintos se cambian en diferentes palabras, la paridad por columna quizás no detecte, pero la suma total sí (porque el valor numérico cambia).
 - Más cobertura de datos “agrupados”.

Ejemplo (sumita de 16 bits “estilo Internet”)

Supongamos que queremos calcular el checksum de 4 palabras de 16 bits cada una (por simplificar el ejemplo).

Digamos que las 4 palabras son (en binario):

$W_1 = 00000000\ 00001111$ ($0x000F = 15$)
 $W_2 = 00000000\ 00110011$ ($0x0033 = 51$)
 $W_3 = 00000000\ 01010101$ ($0x0055 = 85$)
 $W_4 = 00000000\ 11110000$ ($0x00F0 = 240$)

1. Sumás $W_1 + W_2 + W_3 + W_4$ en **complemento a uno**. Primero hacemos “sumita normal” y después ajustamos overflow: = $0x0187$ (391 en decimal)
- Ahora, ojo: cada palabra tiene 16 bits, o sea que modulo $0x10000$ (65536). $0x0187$ cabe sin desbordar: no pasamos de 16 bits. Pero si hubiésemos pasado, digamos que la suma fuera $0x10023$, tomaríamos el “carry” y lo sumaríamos de vuelta al resultado (esa es la gracia del complemento a uno).

2. Tomamos el **complemento a uno** de 0x0187:

- 0x0187 en binario de 16 bits: 0000 0001 1000 0111
- Su complemento a uno es 1111 1110 0111 1000 (que equivale a 0xFE78).

3. Esos 16 bits (1111 1110 0111 1000) es el **checksum** que van a mandar al final del mensaje.

4. El receptor hace la misma suma: $W_1 + W_2 + W_3 + W_4 + (\text{checksum}) = 0x0187 + 0xFE78$

- $0x0187 + 0xFE78 = 0x100FF$. Ahora, $0x100FF > 0xFFFF$, hay desbordamiento:
 - Sacás el “1” alto y lo sumás al resto $\rightarrow 0x00FF + 0x0001 = 0x0100 \rightarrow$ y volvés a hacer complemento a uno (si así se define el protocolo).
- Idealmente queda todo 1s (o todo 0s, según la convención). Si no da ese patrón, sabés que hubo error.

Limitaciones:

- No detecta bien si “eliminas” o “insertás” palabras que sumen exactamente cero o si movés bloques de tamaño múltiplo de 16 bits de lugar.
- Por eso se inventaron variantes como la **suma de verificación de Fletcher**, que suma también la posición de cada palabra para captar reordenamientos.

5. CRC (Cyclic Redundancy Check)

<https://www.youtube.com/watch?v=Wj5ZLtBJ3j8>

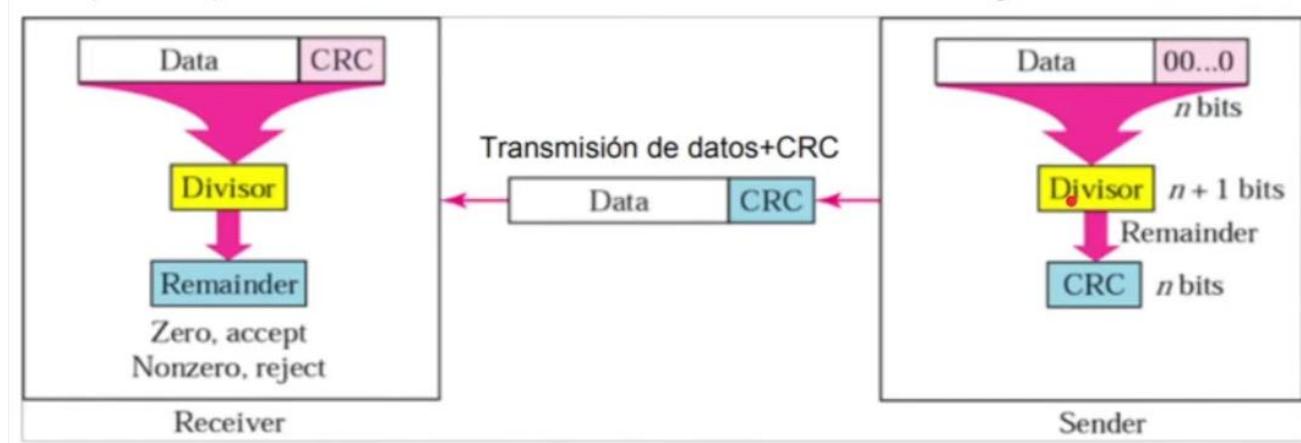
Idea básica

• Comprobación de redundancia cíclica

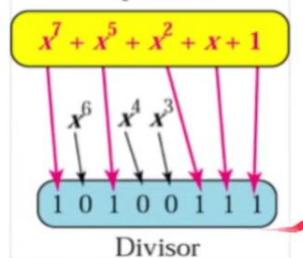
- Cuando se producen errores de tipo ráfaga es necesario utilizar técnicas basadas en **códigos polinomiales**
 - Estas técnicas se denominan **comprobación de redundancia cíclica** (CRC, *Cyclic Redundance Check*)
 - También se conocen con el nombre de **secuencia de comprobación de trama** (FCS, *Frame Check Sequence*)
- Funcionamiento

Receptor: comprobación del CRC

Emisor: generación del CRC



- Ejemplo de generador polinomial (8 bits)



- Algunos generadores polinomiales estándar

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
ITU-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
ITU-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

- Si da residuo $\neq 0$, hubo error.
- Si da residuo = 0, asume que llegó bien (pero existe un pequeñísimo riesgo si el error origina un polinomio E(x) que sea múltiplo de G(x); esos casos quedan “colados”, pero la probabilidad es bajísima si elegís bien G(x)G(x)).

¿Por qué es tan bueno un CRC?

- Detecta **todos** los errores de ráfaga de longitud $\leq r$ (en nuestro ejemplo, ráfagas ≤ 4 bits).
- Detecta **casi** todos los errores de ráfaga más largos (la probabilidad de dejar pasar uno que no se note es $\approx 1/2^r$).
- Detecta cualquier error impar de bits si el polinomio generador tiene el factor $(x + 1)$.
- Tiene implementaciones muy simples en hardware (usando registros de desplazamiento con XORs).

¿Cuál polinomio usar?

- El estándar **Ethernet/IEEE 802** usa un generador de grado 32: que en binario se escribe (33 bits): (obviamente es largo; por eso se implementa con registros de desplazamiento).
- **Castagnoli/Koopman** encontraron mejores generadores de grado 16, 32, etc., que tienen más distancia de Hamming y detectan más patrones de error.

Ejemplo:

Dividimos el polinomio generador por la trama, rellenado por la cantidad de bits de generador menos uno. En este caso tiene 5, así que rellenamos con 4 ceros. Tomamos el generador desde el bit más significativo, y dividimos. Es básicamente hacer una OR exclusiva. 0 cuando son (11 o 00) y distintos 0. Luego bajamos un bit. Buscamos que tengan en total 5 bits para poder dividirlo por el generador. Repetimos. En el último paso, ya no nos queda más para bajar, obteniendo 1 0. Ese residuo sumado a los 4 ceros de relleno. Ósea 0010

Trama:	1 1 0 1 0 1 1 1 1 1	
X^4+X^1+1 Generador:	1 0 0 1 1	
	1 1 0 0 0 0 1 1 1 0 ← Cociente (se descarta)	
1 0 0 1 1 /	1 1 0 1 0 1 1 1 1 0 ← Trama con cuatro ceros adjuntos	
	1 0 0 1 1	
	1 0 0 1 1	
	0 0 0 0 1	
	0 0 0 0 0	
	0 0 0 1 1	
	0 0 0 0 0	
	0 0 1 1 1	
	0 0 0 0 0	
	0 1 1 1 1	
	0 0 0 0 0	
	1 1 1 1 0	
	1 0 0 1 1	
	1 1 0 1 0	
	1 0 0 1 1	
	0 0 0 1 0	
	0 0 0 0 0	
	1 0	← Residuo
Trama transmitida:	1 1 0 1 0 1 1 1 1 1 0 0 1 0 ← Trama con cuatro ceros adjuntos menos el residuo	

Resumen comparativo y cuándo usar cada uno

Método	Bits de verificación	Detecta errores de ráfaga	Ventajas / Limitaciones
Paridad simple	1 bit por bloque	Poco	Muy sencillo, detecta errores de 1 bit. No detecta ráfagas (si hay 2 bits cambiados parejos).
Paridad 2D	k bits horizontales + n bits verticales	Ráfagas $\leq n$ (si intercalás)	Mejora la detección de ráfagas, pero hace más sobrecarga que paridad simple.
Checksum 16 bits	16 bits por bloque (1 palabra)	Moderado	Detecta más errores que paridad simple; pero si se mueven bloques de datos puede fallar.
Fletcher	Varía (ej. 16 o 32 bits)	Mejor que checksum convencional	Suma valores y posiciones, detecta reordenamientos "suaves".
CRC	r bits (p. ej. 16, 32)	Sí, ráfagas $\leq r$	Muy potente: detecta ráfagas largas, errores impares. Implementable en hardware.

PROTOCOLOS ELEMENTALES DE ENLACE DE DATOS

- **Separación en capas y procesos independientes**

- Se considera que las capas físicas, de enlace de datos y de red operan como entidades separadas que intercambian mensajes.
- La parte física y algunos mecanismos de enlace suelen implementarse en hardware especializado (ej. una tarjeta de red), mientras que el resto de la capa de enlace y la capa de red se ejecutan en el sistema operativo. Esto permite cambiar el hardware de red sin afectar el funcionamiento interno de la capa de red.

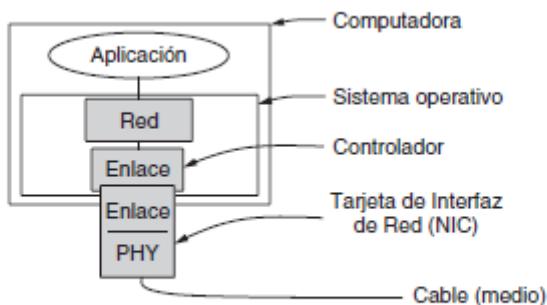


Figura 3-10. Implementación de las capas física, de enlace de datos y de red.

- **Flujo básico de datos**

1. La capa de red entrega datos listos para enviar a la capa de enlace (en protocolos sencillos se asume que siempre hay datos disponibles).
2. La capa de enlace crea una unidad llamada **trama**, que incluye:
 - Información de control (tipo de trama, números para llevar conteo de orden y acuses).
 - El contenido de la capa de red (el paquete).
 - Una comprobación de integridad (normalmente un CRC) para detectar errores en la transmisión.
3. La trama se envía a través del medio físico.
4. En el receptor, si la comprobación de integridad es correcta, se extrae el contenido y se entrega a la capa de red local. En ningún caso la capa de red recibe los detalles de control de la trama; solo ve los datos que le corresponden.

- **Manejo de eventos en la capa de enlace**

- La capa de enlace permanece esperando que ocurra alguno de estos sucesos:
 - **Llegada de trama correcta:** el receptor recibe una trama con integridad verificada.
 - **Llegada de trama con error:** la comprobación (CRC) falla y la trama se descarta o se informa del error.
 - **Capa de red lista:** la capa de red indica que tiene datos para enviar (en versiones más avanzadas del protocolo).
 - **Temporizador expirado:** significa que se esperaba un acuse y no llegó a tiempo, por lo que habrá que retransmitir.

- **Temporizador auxiliar para acuse:** se usa para agrupar o retrasar la generación de confirmaciones cuando no llega tráfico de datos.
 - Cuando llega una trama válida, la capa de enlace analiza sus campos de control y, si corresponde, entrega el contenido a la capa de red.
 - Si llega una trama dañada, la capa de enlace puede ignorarla o activar mecanismos de recuperación según el protocolo (por ejemplo, planear una retransmisión).
- **Temporizadores y recuperación de pérdidas**
 - Se asume que el canal puede perder tramas completas.
 - Cada vez que se envía una trama, se pone en marcha un temporizador.
 - Si antes de que expire se recibe el acuse correspondiente, se detiene dicho temporizador.
 - Si el temporizador expira, se entiende que el acuse no llegó y se retransmite la trama.
 - Un temporizador auxiliar sirve para decidir cuándo enviar confirmaciones de recepción si no hay tráfico adicional, evitando mandar un acuse inmediato por cada trama recibida.
- **Control de flujo**
 - Para evitar que la capa de red envíe más datos de los que la capa de enlace puede manejar, existe un mecanismo de **habilitar/deshabilitar** la capa de red.
 - Cuando la capa de enlace está saturada o sin espacio en búfer, indica que la capa de red debe esperar.
 - Cuando vuelve a tener espacio libre (por ejemplo, tras procesar una trama o recibir un acuse), habilita la capa de red para que reanude el envío.
- **Supuestos generales para protocolos elementales**
 1. **Máquinas confiables:** no se consideran fallas de los propios nodos, solo errores de transmisión.
 2. **Servicio orientado a conexión:** inicialmente, la comunicación es unidireccional (A a B), con datos siempre listos. Después se extiende a tráfico bidireccional.
 3. **Intercambio de tramas transparente para la capa de red:**
 - La capa de red nunca ve ni genera los campos de control de enlace: solo interactúa mediante “paquete a enviar” y “paquete recibido”.
 - Esto permite cambiar o mejorar el protocolo de enlace sin tocar la capa de red.

Resumen de lo esencial para entender la estructura de un protocolo de enlace de datos:

1. **Independencia de capas:** física, enlace y red operan como entidades separadas que sólo intercambian unidades de datos.
2. **Encapsulamiento en tramas:** cada paquete de red se envuelve en una trama que incluye control (números de secuencia/acuse) y un mecanismo de detección de errores (CRC).
3. **Eventos clave:** llegada de trama (correcta o con error), capa de red lista para enviar, expiración de temporizadores.
4. **Temporizadores obligatorios:** para retransmitir tramas ante ausencia de acuse y para regular el envío de confirmaciones.

5. **Control de flujo:** se habilita o deshabilita la capa de red según la capacidad de la capa de enlace para evitar congestión.
6. **Transparencia hacia la capa de red:** la capa de enlace oculta por completo su formato interno de tramas, entregando únicamente el contenido (“paquete”) a la capa superior.

1. Protocolo Simplex Utópico

- **Contexto ideal:**

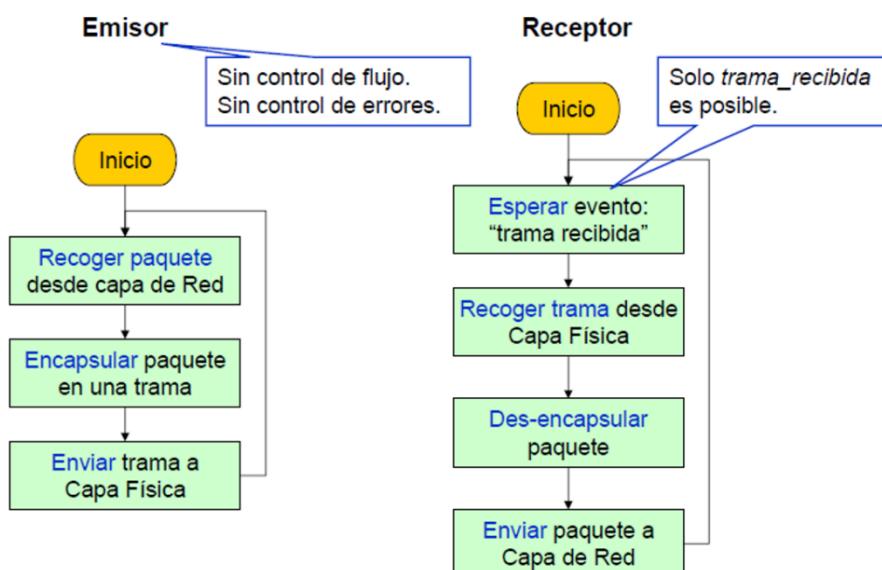
- Se asume un canal perfecto (0 errores, 0 pérdidas, latencia cero o despreciable, buffers infinitos).
- Además, la capa de red (o la aplicación) entrega datos “sin parar” y el receptor siempre los procesa de inmediato.

- **Funcionamiento:**

- El emisor manda tramas una tras otra sin esperar ninguna confirmación.
- El receptor recibe todo y lo procesa sin chequear nada.

- **Limitaciones reales:**

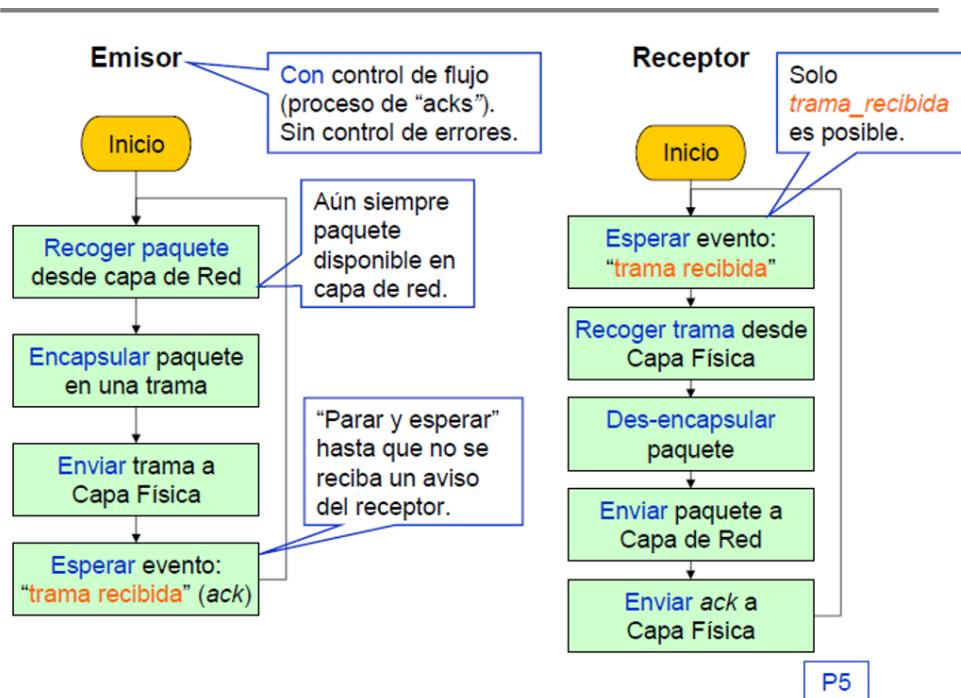
- Es completamente irrealista: no hay control de errores, no hay control de flujo y la saturación o fallas del canal no se manejan.
- Únicamente sirve para entender el caso más básico; en la práctica, ningún sistema funciona así.



2. Protocolo Simplex de Parada y Espera (sin errores)

- **Objetivo principal:** Controlar el flujo para no desbordar al receptor, asumiendo que el canal no comete errores.
- **Funcionamiento:**
 1. El emisor envía **una única trama** y luego invoca un “espera” (por ejemplo, para recibir un ACK).

2. Cuando llega el ACK, el emisor sabe que puede enviar la siguiente trama.
 3. El receptor, al recibir la trama, la procesa y envía inmediatamente un ACK (acuse de recibo) de vuelta.
- **Puntos clave:**
 - La capa de red siempre tiene datos listos, y el emisor no envía nada hasta recibir el ACK anterior.
 - El receptor, una vez que procesa la trama, manda el ACK, liberando al emisor para la siguiente.
 - Simplex: datos en una sola dirección.
 - Comunicación bidireccional.
 - **Limitaciones y comportamiento real:**
 - Si el ACK se pierde, el emisor se queda esperando para siempre (o hasta que expire un temporizador, si se agregara uno).
 - El canal está subutilizado: el emisor pasa mucho tiempo “parado” mientras espera al ACK, especialmente en enlaces de alta latencia.
 - No maneja errores de transmisión ni pérdidas, pues asume canal perfecto.



3. Protocolo Simplex de Parada y Espera (con ruido)

Cuando el canal **puede perder o corromper** tramas o ACKs, hace falta agregar:

1. **Números de secuencia de 1 bit (0/1).**
2. **Temporizador en el emisor** para detectar ausencia de ACK.
3. El receptor debe ignorar duplicados y reenviar el último ACK.

Funcionamiento paso a paso

- El emisor envía una trama con su número de secuencia (sec = 0 o 1).
- Arranca un temporizador.
- El receptor, al llegar la trama:
 1. Verifica CRC/suma de verificación.
 2. Si está bien y el número de secuencia es el esperado, entrega los datos a la capa de red y envía un ACK con ese mismo número de secuencia.
 3. Si llega una trama con número de secuencia duplicado (porque el ACK anterior se perdió y el emisor reenvía), el receptor descarta la trama y vuelve a enviar el ACK para confirmar la que ya procesó.
- El emisor, al recibir un ACK con el número correcto, sabe que la trama llegó bien, alterna su bit de secuencia ($0 \rightarrow 1$, $1 \rightarrow 0$) y envía la próxima trama.
- Si el temporizador vence antes de recibir el ACK, el emisor **retransmite** la misma trama (con el mismo sec).

Puntos importantes para entender:

- El uso de **1 bit de secuencia** (0/1) basta para distinguir “trama nueva” de “repetida”.
- El temporizador resuelve la falta de ACK: cuando expira, asume que la trama se perdió (o el ACK) y retransmite.
- El receptor no envía ACK a tramas duplicadas; simplemente reenvía el último ACK para que el emisor avance.
- Aun así, la eficiencia sigue siendo baja: si la latencia canal–receptor es grande, el emisor queda “parado” esperando ACK, por más que el canal esté libre.

■ Errores:

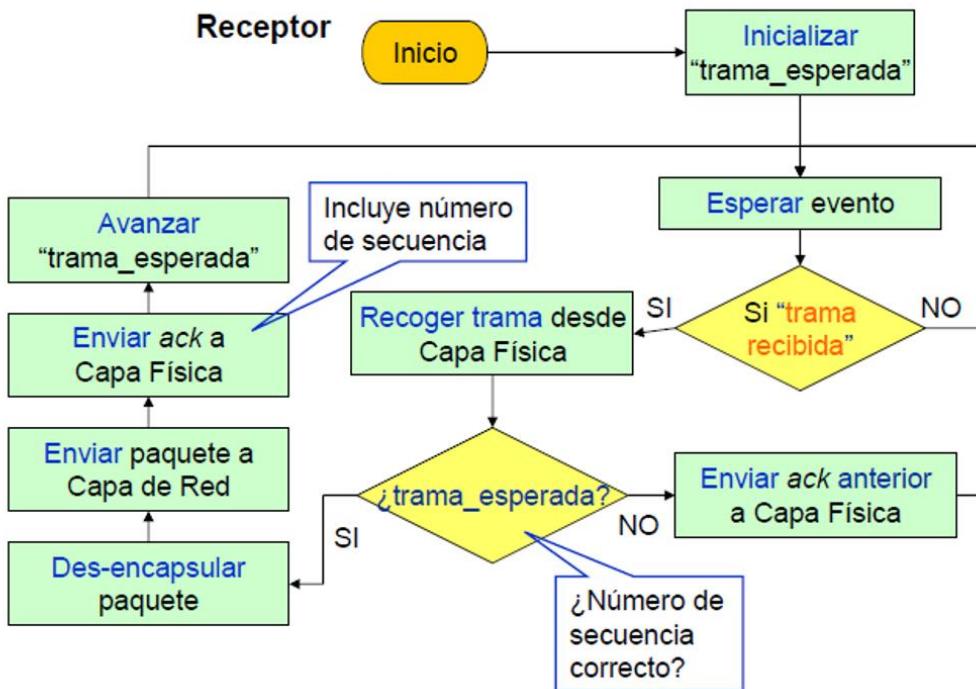
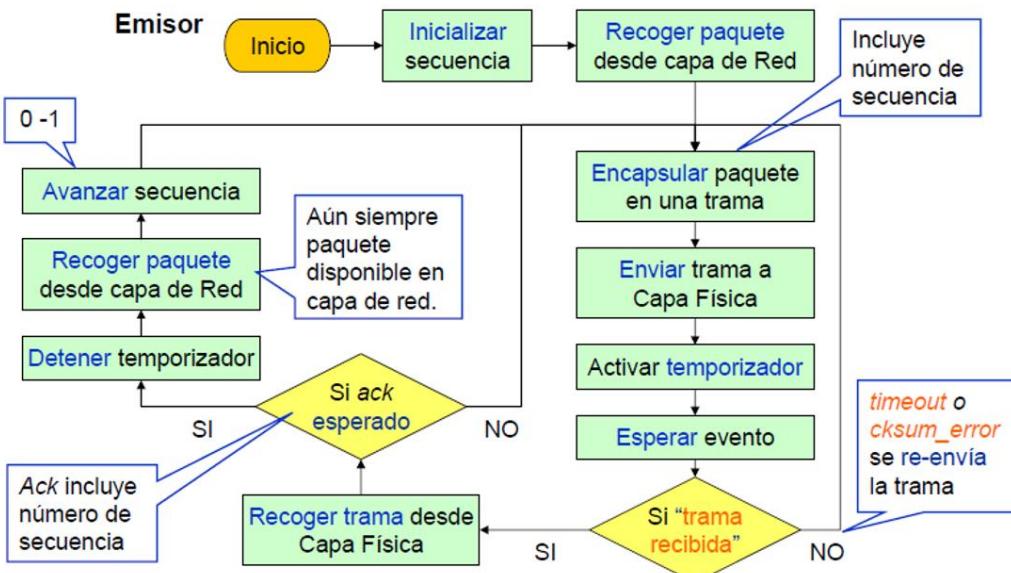
- Tramas perdidas.
- Hardware del receptor detecta errores por *checksum*.

■ Capa de *enlace de datos*:

- Debe **garantizar** no enviar dos veces el mismo mensaje a capa de red.
- Se necesita algún tipo de identificación:
 - **Número de secuencia**.
 - Secuencia mínima: 1-bit (0 –1).

■ Protocolos **PAR** (*Positive Acknowledgement with Retransmission*) o **ARQ** (*Automatic Repeat reQuest*):

- Emisor espera un **acuse de recibo positivo** antes de avanzar en la secuencia de emisión.



Protocolo de Ventana Deslizante (generalidades)

<https://www.youtube.com/watch?v=LcITnU58e1M>

<https://www.youtube.com/watch?v=XuUEeYWEXg4>

La idea central de la **ventana deslizante** es lograr un mejor aprovechamiento del canal, enviando varias tramas sin esperar un ACK individual por cada una.

Conceptos básicos

1. **Ventana de emisor:** conjunto de números de secuencia que el emisor tiene permitido “tener en vuelo” (tramas enviadas y pendientes de ACK).
2. **Ventana de receptor:** conjunto de números de secuencia que el receptor acepta; las tramas que llegan fuera de esa ventana se descartan (o guardan, según el protocolo).

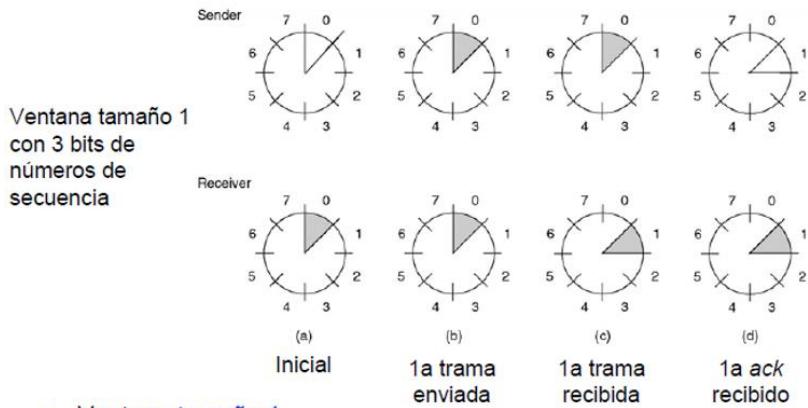
3. Ambos extremos mantienen un contador (o puntero) que indica el límite inferior de su ventana y su tamaño (generalmente denotado como N).
4. Los números de secuencia se manejan de forma **circular**: después de MAX_SEQ viene 0.

Funcionamiento general

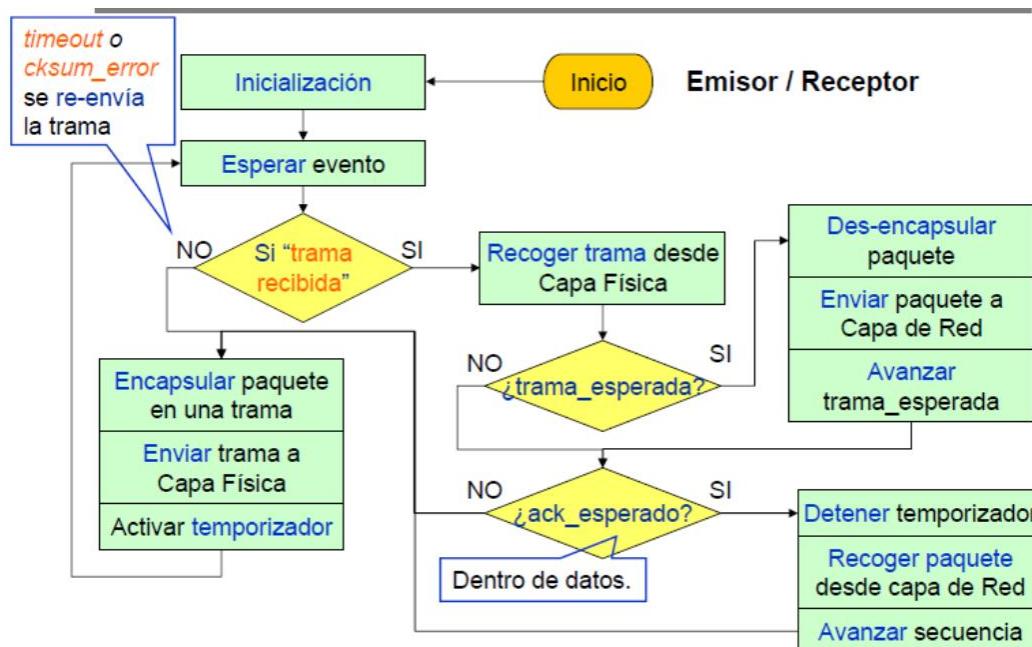
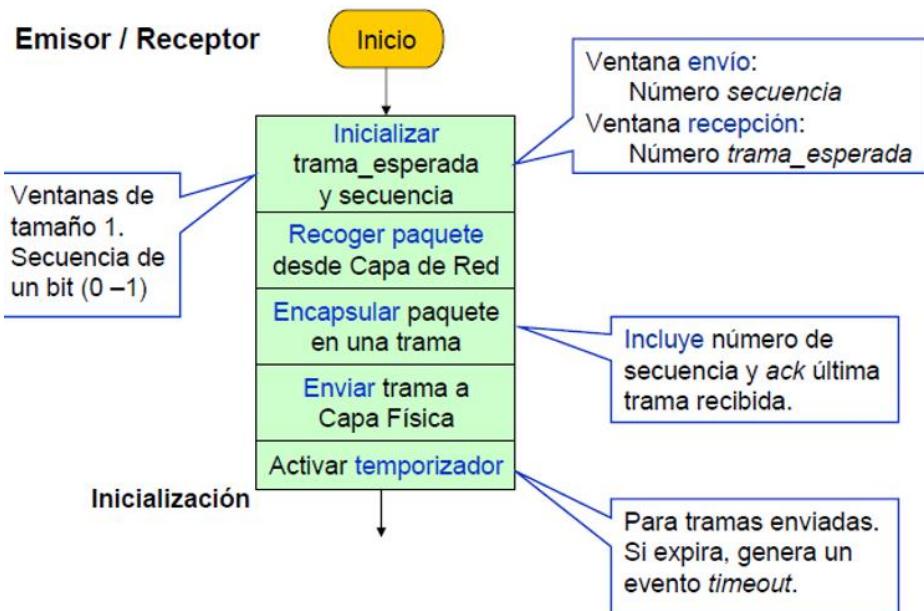
- El emisor puede enviar hasta N tramas seguidas (con secuencias consecutivas) antes de detenerse.
- Cada vez que llega un ACK (posiblemente **acumulativo**, es decir, confirma todas las tramas hasta cierto número), el emisor “desliza” la ventana y puede enviar tantas tramas nuevas como el espacio que quedó libre.
- El receptor, al recibir tramas, comprueba la secuencia:
 - En algunos protocolos (Go-Back-N) solo acepta la siguiente trama esperada y descarta las posteriores, pidiendo que se retransmita todo a partir del error.
 - En otros (Selective Repeat) puede almacenar tramas fuera de orden dentro de su ventana, confirmar individualmente cada una y reordenar antes de entregar a la capa superior.

Ventajas sobre parada-y-espera

- Aprovechamiento mucho mayor del canal, sobre todo si la latencia de ida y vuelta es alta.
- Hasta que se recibe un ACK, se mantienen “N” paquetes en vuelo, maximizando el uso de la capacidad.
- Permite un control de flujo más fino: el tamaño de ventana limita cuántas tramas pendientes puede haber sin saturar buffers.
- Transmisión en **ambas direcciones**:
 - Se podría utilizar protocolos *simplex* con dos canales.
 - **Mismo canal** en ambas direcciones:
 - Protocolos de **ventana deslizante**.
- **Ventana**:
 - Conjunto de **números de secuencia** de tramas **permitidas**
 - **Ventana emisora:** tramas permitidas para enviar.
 - **Ventana receptor:** tramas permitidas para aceptar.
- **Ack incluido** en un mensaje de datos de vuelta (*piggybacking*):
 - **Aprovecha** mejor el ancho de banda del canal.
 - **Esperar** un cierto tiempo a enviar un paquete, sino, enviar un mensaje de **ack explícito**.



- Ventana tamaño 1:
 - Capa enlace recibe los paquetes en orden.
 - En general, tamaño n , no.
- A capa de red: hay que garantizar orden de paquetes.



Entubamiento/Canalización (Pipelining)

Llenar un canal como si fuese un tubo.

Problema de ineficiencia en transmisión

Ocurre cuando se combinan:

1. **Tiempo de tránsito elevado:** Retrasos largos en la propagación de señales.
2. **Gran ancho de banda:** Capacidad alta para enviar datos.
3. **Tramas cortas:** Paquetes pequeños de datos.

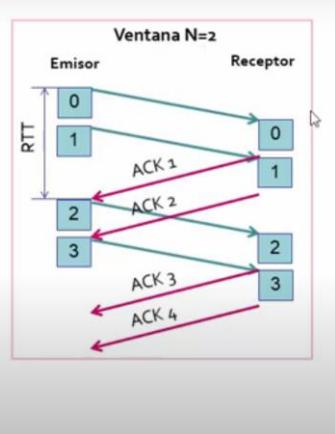
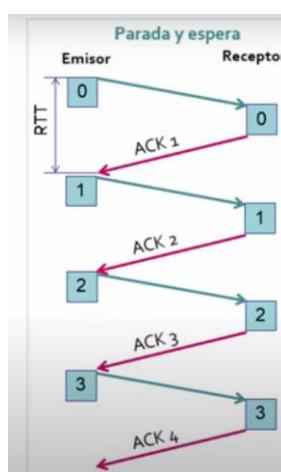
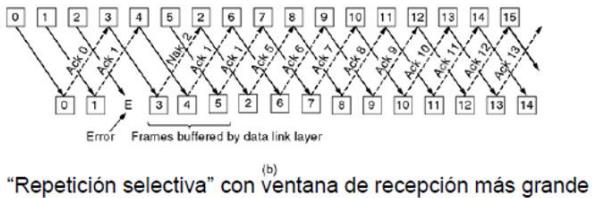
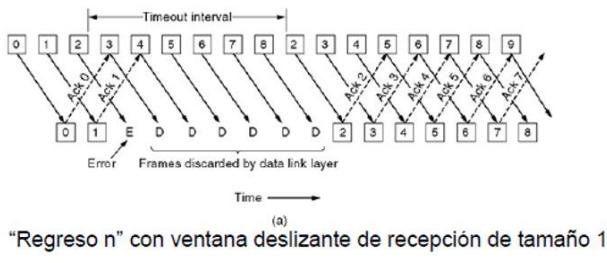
Esta combinación puede dejar gran parte del canal ("tubo") sin utilizarse, reduciendo la eficiencia.

Solución: Técnicas de entubamiento

Para enviar información de forma continua y eficiente, se usan métodos como:

1. **"Regreso N" (Go-Back-N):**
 - Si falla un paquete, se retransmiten **todos los paquetes posteriores**.
 - Simple pero ineficiente si hay muchos errores.
2. **"Repetición selectiva" (Selective Repeat):**
 - Solo se retransmiten **los paquetes perdidos o corruptos**.
 - Más eficiente, pero con mayor complejidad.

Recuperación del error en técnicas de entubamiento



5. Protocolo Go-Back-N (Retroceso N)

Esta es una versión concreta de ventana deslizante con comportamiento específico ante errores.

Características clave

1. El emisor tiene una ventana de tamaño N. Puede enviar tramas 1,2,...,N sin esperar ACK.
2. El receptor solo acepta la **siguiente trama esperada** (por ejemplo, si le falta la secuencia 3, descarta la 4, 5, 6, etc.).

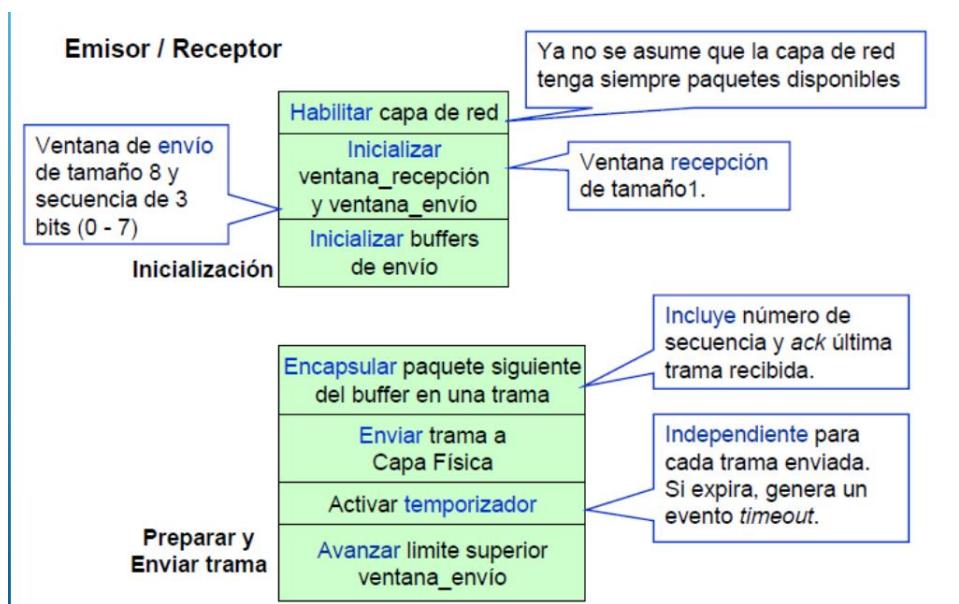
3. El receptor envía ACK **acumulativo**: si recibe la trama 3 (esperada), envía “ACK = 3” (o “ACK = 4”, dependiendo de la convención), que significa “todo hasta la 3 llegó bien; espero la 4”.
4. Si el emisor detecta que la trama 2 no fue confirmada (por ejemplo, no recibe ACK en tiempo), espera el timeout, retrocede (go back) al número de secuencia 2 y **retransmite** desde 2 en adelante (2, 3, 4, ... hasta donde alcance su ventana).

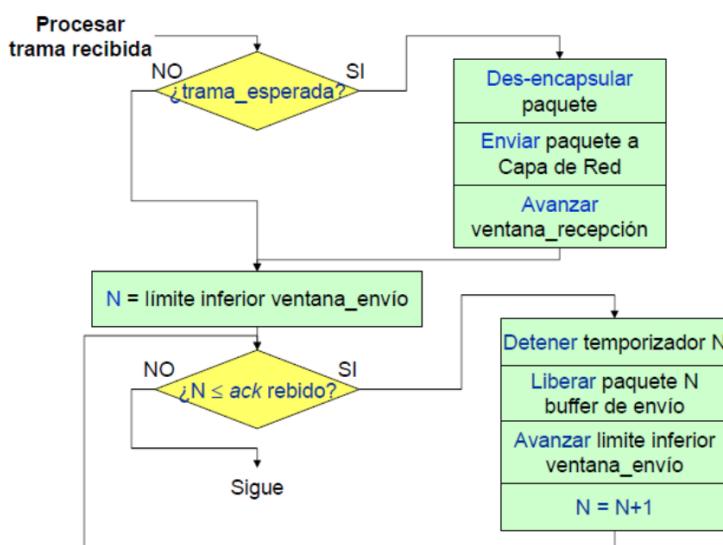
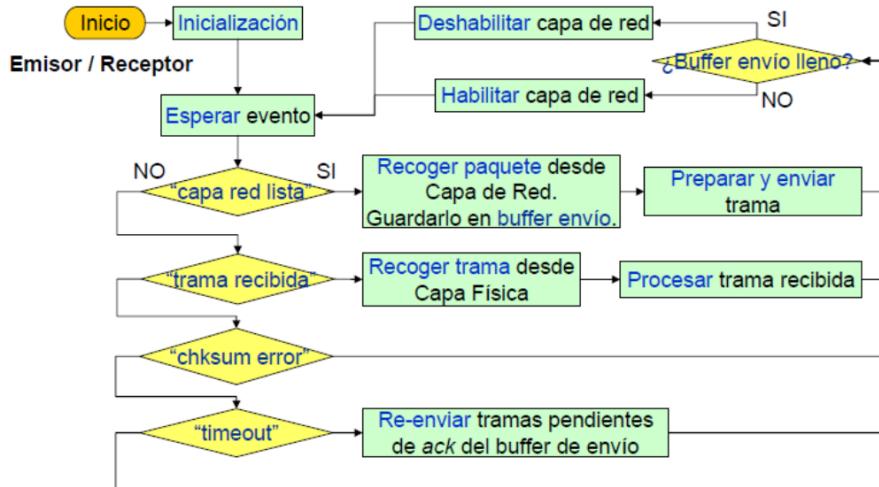
Ventajas

- Con un canal razonablemente bueno (pocas pérdidas), se reduce la cantidad de “paradas” en comparación con parada-y-espera.
- Muy sencillo de implementar: el receptor no necesita almacenar tramas fuera de orden, ni lidiar con reordenamientos.
- El ACK acumulativo simplifica el proceso: con un solo ACK, el emisor sabe que todas las anteriores fueron recibidas.

Desventajas

- Si se pierde una trama intermedia (p. ej. la 2), el receptor descarta todas las posteriores (3, 4, 5...) aun cuando pudieran llegar sin error.
- Cuando hay un error, el emisor debe retransmitir **todas** las tramas desde el punto de falla, aunque algunas hubiesen llegado bien.
- En canales muy ruidosos, el rendimiento cae porque el número de retransmisiones aumenta y la ventana se deshace con frecuencia.





Protocolo de Repetición Selectiva (Selective Repeat)

- Emisor y receptor:
 - Ventana de tamaño aceptable.
- Permite recibir tramas desordenadas:
 - Asegurar orden a capa de red.
- Si no hay tráfico de vuelta:
 - Temporizador para acuses de recibo.
 - Enviar "ack" cuando fuera de tiempo.
 - "Timeout" apreciablemente menor que para envío de tramas.
- Si receptor tiene sospechas de error en recepción:
 - Devuelve un acuse de recibo negativo "nak".
 - Sólo una vez por trama esperada.

Es una mejora sobre Go-Back-N para reducir retransmisiones innecesarias.

Características clave

1. El emisor también usa una ventana de tamaño N, pero solo retransmite **la(s) trama(s) errónea(s)**.
2. El receptor:

- **Almacena** (buffer) tramas que llegan fuera de orden, siempre que estén dentro de su ventana de recepción.
 - Envía **ACK individual** para cada trama correcta recibida (no es acumulativo).
 - Entrega a la capa de red solo cuando puede reconstruir la secuencia completa en orden.
3. El emisor, al recibir un ACK para, digamos, la 4, asume que solo hubo error en otras tramas no confirmadas (p. ej. la 2 o la 3, según el caso) y retransmite solo esas.

Condición importante sobre el tamaño de ventana

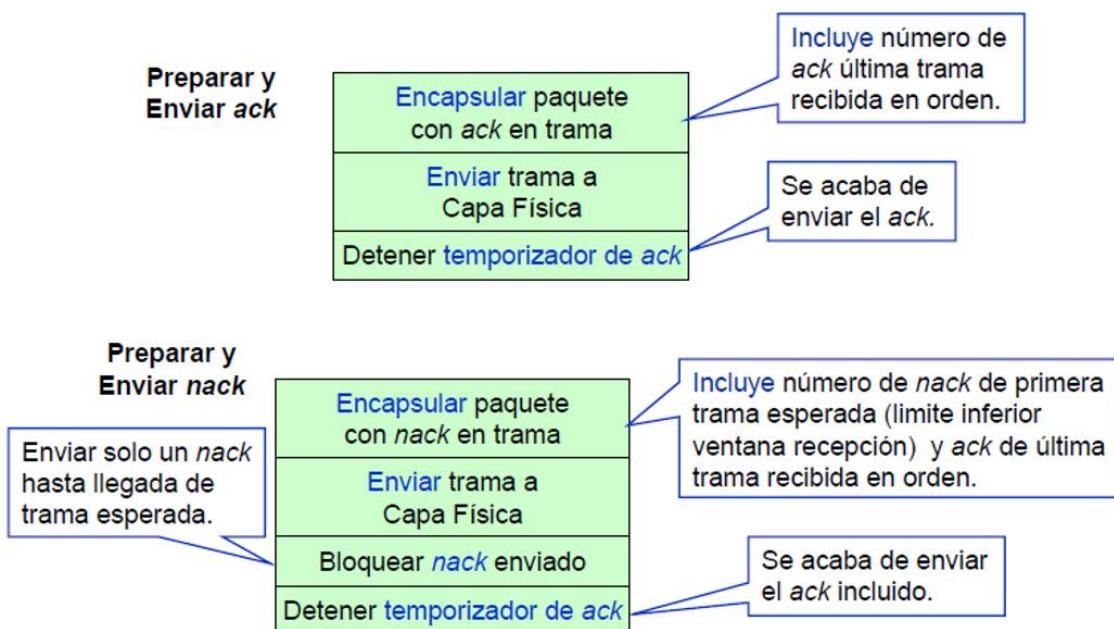
- Para evitar ambigüedades de secuencias, el tamaño máximo de ventana debe ser $\leq \frac{\text{MAX_SEQ}+1}{2}$.
- Esto garantiza que, cuando el emisor y receptor "avancen", no haya confusión sobre si un ACK o una trama corresponde a un ciclo antiguo de secuencias.

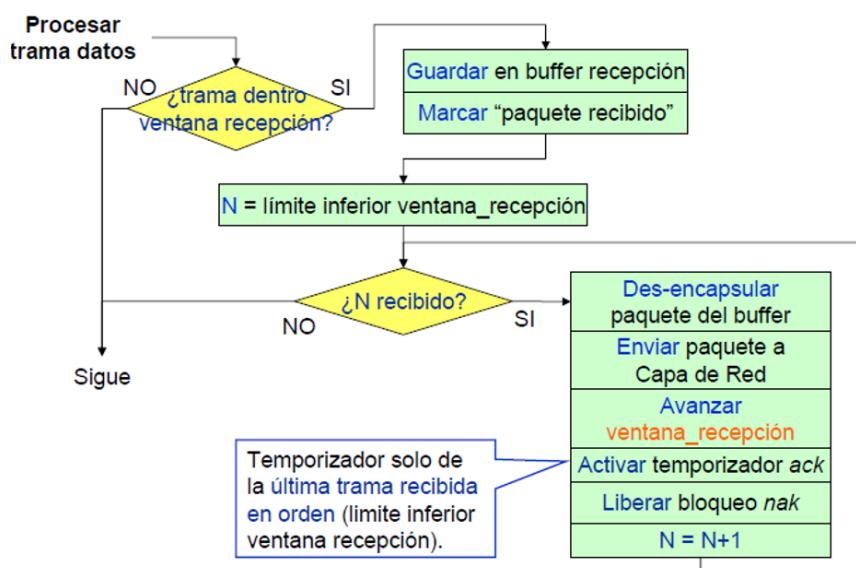
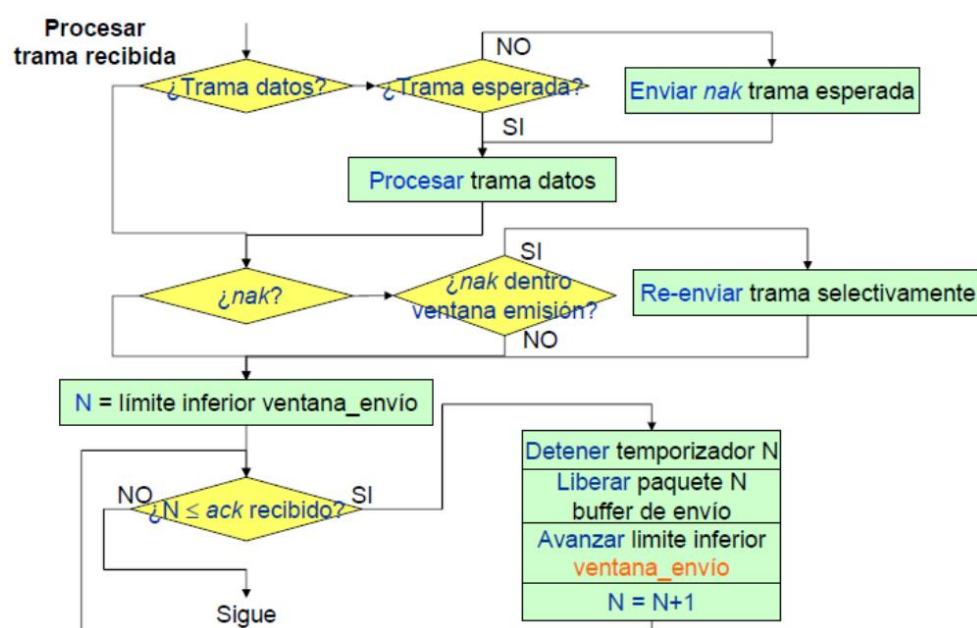
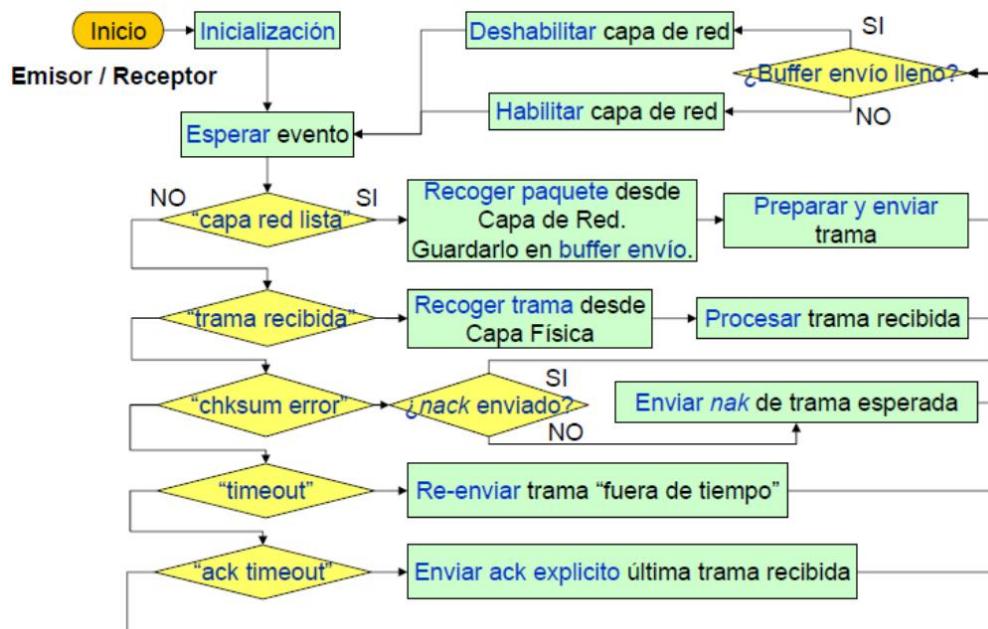
Ventajas

- Retransmite solo las tramas específicas con error, mejorando la eficiencia en canales con pérdidas moderadas.
- El receptor puede aprovechar tramas correctas fuera de orden en lugar de descartarlas.

Desventajas

- Requiere **mayor complejidad**:
 - Almacenamiento de tramas fuera de orden en el receptor.
 - Control/seguimiento de ACKs individuales en el emisor.
 - Mecanismos para manejar huecos en la recepción (mantener buffers, despejar espacio cuando se entregan tramas continuas).
- Se necesitan más cuentas de temporizadores (uno por cada trama en vuelo) o lógicas más complejas para decidir cuándo retransmitir.





7. Caso de Estudio: TCP/IP

TCP (Transmission Control Protocol) implementa una versión de ventana deslizante con características adicionales de control de errores, flujo y **congestión**. Aunque no es idéntico a Selective Repeat puro, sí combina varias ideas:

Aspectos de control de errores y flujo en TCP

1. Números de secuencia por byte:

- No numeran tramas, sino bytes. Cada segmento TCP lleva un “Número de secuencia” que indica el primer byte de datos que transporta.

2. ACK acumulativo por byte:

- El receptor envía un ACK cuyo número de reconocimiento (ACK number) es el siguiente byte que espera. Eso implica que, si recibió bytes 1–100, el ACK es 101 (aunque hubiera recibido bytes 101–150 fuera de orden, solo confirmará cuando tenga todo en orden hasta el último continuo).

3. Ventana de recepción (rwnd):

- Cada ACK lleva un campo “Ventana” que indica cuántos bytes más puede aceptar el receptor en su buffer. Eso controla el flujo: el emisor no envía más de “lo que le permite” la ventana.

4. Temporizadores:

- **RTO (Retransmission Timeout)**: si expira antes de recibir ACK, retransmite todo el segmento pendiente.
- **Retransmisión rápida (Fast Retransmit)**: si llegan 3 ACK duplicados consecutivos (mismo número de reconocimiento), se asume que falta ese segmento intermedio y se retransmite sin esperar al RTO.

Control de congestión en TCP

- TCP monitoriza la red para evitar “reventar” routers o enlaces intermedios. Introduce dos conceptos clave:
 1. **Ventana de congestión (cwnd)**: tamaño dinámico que el emisor ajusta según cómo evolucione la red.
 2. **Slow Start y Congestión Avoidance**: algoritmos que regulan cwnd.
 - **Slow Start**: al iniciar una conexión o después de detectar pérdida, parte de $cwnd = 1 \text{ MSS}$ (Max Segment Size). Por cada ACK recibido, incrementa cwnd en 1 MSS (crece exponencialmente).
 - **Congestión Avoidance**: cuando cwnd alcanza un umbral (ssthresh), en lugar de duplicar, lo incrementa de forma lineal ($\sim 1 \text{ MSS}$ por RTT).
 - Si hay timeout, se reduce drásticamente: $cwnd = 1 \text{ MSS}$ y $ssthresh = cwnd/2$.

- Si hay triple ACK duplicado, se aplica **Fast Recovery**: $ssthresh = cwnd/2$, $cwnd = ssthresh + 3 \text{ MSS}$, retransmite el segmento perdido y entra a fase de Congestión Avoidance.

Ventajas de TCP

- **Fiabilidad completa**: combina ACKs acumulativos, retransmisiones selectivas (implícitas), control de flujo y congestionamiento.
- **Full-duplex**: datos y acuses viajan en ambas direcciones si la aplicación lo requiere.
- **Adaptabilidad**: ajusta dinámicamente el envío según las condiciones del canal y la red más amplia.
- **Compatibilidad**: es el estándar en Internet para garantizar entrega ordenada, sin pérdidas y con manejo de congestión.

Cosas faltantes que conviene remarcar

1. **ACKs atrasados (Delayed ACKs)**: el receptor a veces retrasa el envío de ACK unos cientos de milisegundos, esperando tener datos “de regreso” para hacer piggybacking y reducir overhead.
2. **Fin de conexión (Handshake de cierre)**: TCP no solo abre con un “three-way handshake” (SYN, SYN-ACK, ACK), también cierra la conexión con un intercambio de FIN/ACK que asegura que no queden datos pendientes.
3. **Ventana de congestión vs. ventana de flujo**: ambas regulan cuánta información puede haber en vuelo, pero la primera está orientada a no saturar la red, y la segunda a no saturar al receptor. El emisor usa el valor mínimo entre ambas.
4. **Mecanismos modernos de mejora**:
 - **TCP Fast Open, Window Scaling, Selective Acknowledgments (SACK)** avanzados, etc., que permiten usar ventanas muy grandes o reconocer rangos de bytes fuera de orden con mayor precisión.
 - **Timestamps** en el header para estimar RTT con más exactitud y afinar el cálculo del RTO.

Resumen final y conceptos esenciales

1. **Protocolo utópico**: solo para entender el escenario “sin restricciones”.
2. **Parada y espera sin errores**: introduce el concepto de ACK y evita saturar al receptor, pero sin manejar fallas del canal.
3. **Parada y espera con ruido**: agrega números de secuencia y temporizadores para detectar pérdidas, aunque sigue con baja eficiencia en canales de alta latencia.
4. **Ventana deslizante** (genérico): permite tener varias tramas en vuelo y deslizar la ventana al recibir ACK, mejorando el throughput.
5. **Go-Back-N**: retransmite todas las tramas desde el punto de falla; es sencillo pero puede desperdiciar ancho de banda si hay pérdidas.

6. **Selective Repeat:** retransmite solo las tramas erradas, almacena tramas fuera de orden, mucho más eficiente en canales ruidosos, pero más complejo de implementar.
7. **TCP/IP:** protocolo real de Internet que mezcla ventana deslizante, ACKs acumulativos, retransmisiones rápidas y control de congestión, asegurando un servicio confiable y adaptable en cualquier red.

RED DE AREA LOCAL - Unidad 4 Subcapa de acceso al medio.

Tipos de enlaces de red

1. **Conexión punto a punto**
2. **Canales de difusión** (varios dispositivos comparten el medio)

Problema en redes de difusión

¿Cómo decidir quién usa el canal cuando varias estaciones quieren transmitir al mismo tiempo?

En redes de computadoras, una "estación" se refiere simplemente a **un dispositivo que se conecta a la red y puede enviar o recibir datos**

Solución: Subcapa MAC

- **MAC (Control de Acceso al Medio):** subcapa de la **Capa de Enlace de Datos**.
- Decide qué estación puede transmitir en un canal compartido (multiacceso).

PROBLEMA DE ASIGNACIÓN DEL CANAL

Asignación estática de canal en LANs y MANs

Asignación dinámica de canales en LANs y MANs

Modelo de estación.

Supuesto de canal único.
Supuesto de colisión.
Tiempo continuo.

Tiempo ranurado.

Detección de portadora.

Sin detección de portadora.

Protocolos de acceso al medio

ALOHA (Acceso aleatorio)

Permite que las estaciones transmitan libremente. Si hay colisión, deben reintentar.

ALOHA Puro

- Las estaciones **envían sin esperar**, en **cualquier momento**.

- No hay sincronización de tiempo.
- Si hay colisión, se reintenta luego de esperar un **tiempo aleatorio**.
- La estación sabe si su mensaje fue recibido porque el sistema lo retransmite a todos.
- **Eficiencia máxima:**

$$S = G \cdot e^{-2G}$$

ALOHA Ranurado

- El tiempo se **divide en ranuras** (slots).
- Las estaciones **esperan el inicio del siguiente slot** para enviar.
- Requiere **sincronización global** (por ejemplo, un reloj central).
- Reduce las colisiones respecto al ALOHA puro.
- **Eficiencia máxima:**

$$S = G \cdot e^{-G}$$

Emisión de tramas en ALOHA puro

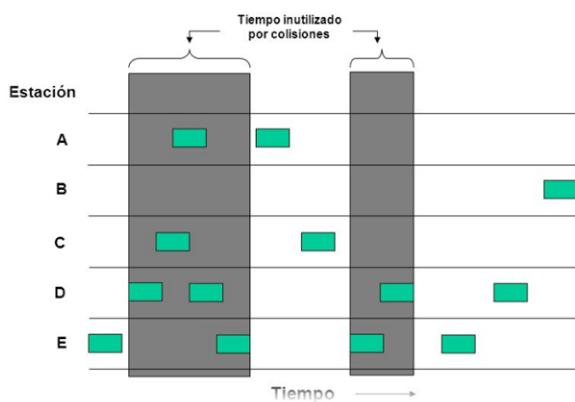


Diagrama De Flujo Del Protocolo ALOHA Puro

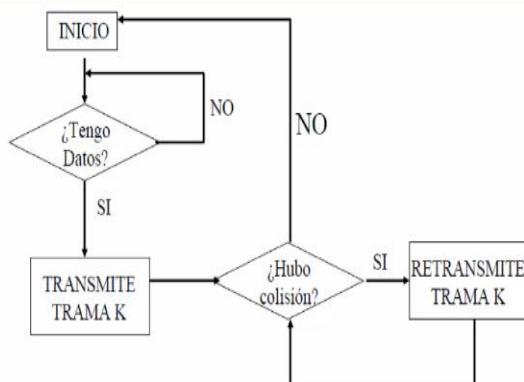
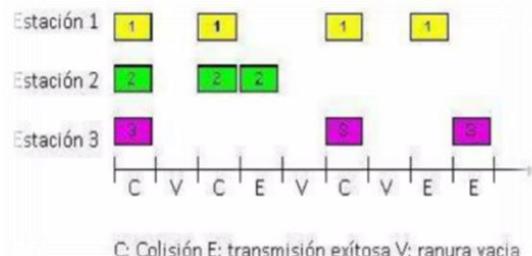
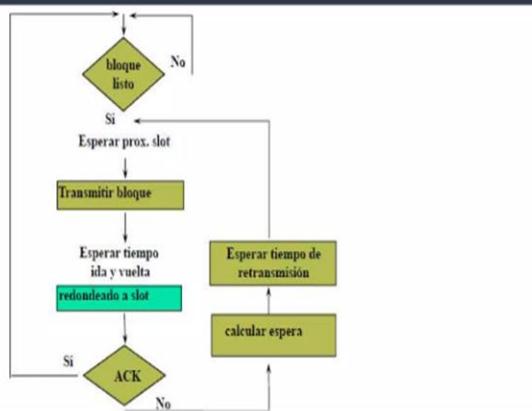
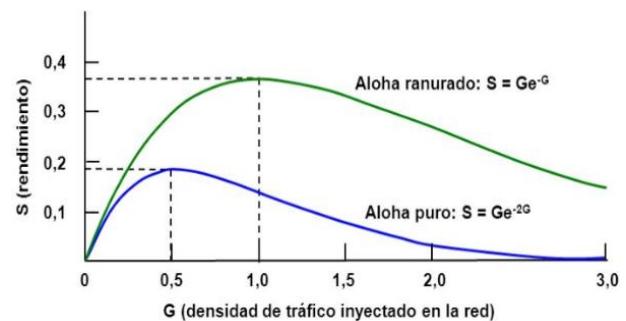
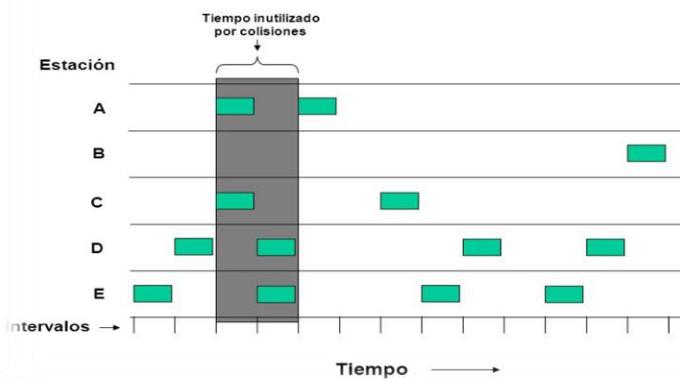


Diagrama De Flujo Del Protocolo ALOHA Ranurado



Emisión de tramas en ALOHA ranurado



Característica	ALOHA Puro	ALOHA Ranurado
Tiempo	Continuo	Dividido en ranuras
Sincronización	No	Sí
Colisiones	Más probables	Menos probables
Eficiencia máx. teórica	18.4%	36.8%

Protocolos de acceso múltiple con detección de portadora

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

CSMA/CD (Carrier Sense Multiple Access with Collision Detection):

- **Carrier Sense (CS):** la estación **escucha** el canal antes de transmitir.
- **Multiple Access (MA):** varias estaciones **comparten** el mismo canal.
- **Collision Detection (CD):** si detectan que hay una **colisión** durante la transmisión, interrumpen la comunicación y **esperan un tiempo aleatorio** para volver a intentarlo.

La meta de este protocolo es de evitar al máximo las colisiones.

Consiste en: ser educado y prudente

1. Oír antes de empezar a hablar (CS, Carrier Sense)

2. Hablar solo cuando los demás callan
3. Si mientras hablamos oímos que otro habla nos callamos (CD, Collision Detect)

Funcionamiento

1. Una estación que tiene un mensaje para enviar escucha al medio para ver si otra estación está transmitiendo un mensaje.
2. Si el medio está tranquilo (ninguna otra estación está transmitiendo), se envía la transmisión y se espera el ACK (acuse de recibo). La estación que recibe comprueba el CRC (detección de errores) y si es correcto envía el ACK. Si tras un tiempo no ha sido recibido el ACK, se pasa al paso 1. Si se recibe, la operación ha sido un éxito.
3. Cuando dos o más estaciones tienen mensajes para enviar, es posible que transmitan casi en el mismo instante, resultando en una colisión en la red.
4. Cuando se produce una colisión, todas las estaciones receptoras ignoran la transmisión confusa.
5. Si un dispositivo de transmisión detecta una colisión, envía una señal de expansión para notificar a todos los dispositivos conectados que ha ocurrido una colisión.
6. Las estaciones transmisoras detienen sus transmisiones tan pronto como detectan la colisión.
7. Cada una de las estaciones transmisoras espera un periodo de tiempo aleatorio e intenta transmitir otra vez.

Ventaja frente a ALOHA

ALOHA (y Slotted ALOHA) simplemente **transmite sin escuchar**, lo que produce muchas colisiones. CSMA mejora mucho esto porque las estaciones primero **escuchan** y en CSMA/CD incluso **detienen la transmisión si se detecta una colisión**.

TIPOS DE CSMA/CD

En función de cómo actúe la estación, el método CSMA/CD se puede clasificar en:

- **CSMA no-persistente:** si el canal está ocupado espera un tiempo aleatorio y vuelve a escuchar. Si detecta el canal libre, emite.
- **CSMA 1-persistente:** con el canal ocupado, la estación pasa a escuchar constantemente el canal sin esperar ningún tiempo. Cuando lo detecta libre emite. Podría ocurrir que emitiera otra estación durante un retardo de propagación o latencia de la red posterior a la emisión de la trama, produciéndose una colisión.
- **CSMA p-persistente:** después de encontrar el canal ocupado y quedarse escuchando hasta encontrarlo libre, la estación decide si emite. Para ello ejecuta un algoritmo o programa que dará orden de transmitir con una probabilidad p , o de permanecer a la espera. Si no transmitiera, en la siguiente ranura o división de tiempo volvería a ejecutar el mismo algoritmo hasta transmitir. Así se reduce el número de colisiones.

Tipo de CSMA	¿Qué hace cuando el canal está ocupado?	¿Qué hace cuando el canal está libre?
1-persistente	Escucha constantemente sin parar hasta que quede libre.	Transmite de inmediato (puede chocar con otra que estaba esperando también).
No persistente	Espera un tiempo aleatorio antes de volver a escuchar.	Si lo encuentra libre, transmite. Menos colisiones.
p-persistente (en redes con tiempo en ranuras)	Espera hasta que quede libre.	Transmite con probabilidad p, espera si no transmite. Repite hasta que lo logre o hasta que otra estación empiece.

CSMA persistente-1.

Cuando una estación tiene datos por enviar, primero escucha el canal para saber si alguien más está transmitiendo en ese momento. Si el canal está inactivo, la estación envía sus datos. Por el contrario, si el canal está ocupado, la estación espera hasta que se desocupa. A continuación, la estación transmite una trama. Si ocurre una colisión, la estación espera una cantidad aleatoria de tiempo y comienza de nuevo.

Si dos estaciones están listas a la mitad de la transmisión de una tercera estación, ambas esperarán amablemente hasta que termine la transmisión y después ambas empezarán a transmitir exactamente al mismo tiempo, lo cual producirá una colisión, Por lo que este esquema no evita las colisiones

Otro aspecto delicado es que el retardo de propagación tiene un efecto importante sobre las colisiones. Existe la posibilidad de que, justo después de que una estación comienza a transmitir, otra estación esté lista para enviar y detecte el canal. Si la señal de la primera estación no ha llegado aún a la segunda, esta última detectará un canal inactivo y comenzará también a enviar, lo que dará como resultado una colisión. Esta posibilidad depende del número de tramas que quepan en el canal. Aun así, este protocolo tiene un mejor desempeño que el ALOHA puro.

CSMA no persistente.

Como antes, una estación escucha el canal cuando desea enviar una trama y, si nadie más está transmitiendo, comienza a hacerlo. Pero si el canal ya está en uso, la estación no lo escuchará de manera continua con el fin de tomarlo de inmediato, sino que esperará un periodo aleatorio y repetirá el algoritmo.

El último protocolo es el CSMA persistente-p.

Cuando una estación está lista para enviar, escucha el canal. Si se encuentra inactivo, la estación transmite con una probabilidad p. Con una probabilidad q 5 1 2 p, se posterga hasta la siguiente ranura. Si esa ranura también está inactiva, la estación transmite o posterga una vez más, con probabilidades p y q. Este proceso se repite hasta que se transmite la trama o hasta que otra estación comienza a transmitir. En el segundo caso, la desafortunada estación actúa como si hubiera ocurrido una colisión (es decir, espera un tiempo aleatorio y comienza de nuevo)

CSMA con detección de colisiones:

Dominio de Colisión

- **Definición:** Segmento físico de una red donde las estaciones comparten un medio de transmisión y usan **CSMA/CD** (Carrier Sense Multiple Access with Collision Detection) para acceder al canal.
- **Limitaciones:**
 - No puede ser arbitrariamente grande para garantizar la eficacia de CSMA/CD.
 - Si hay demasiadas estaciones, aumentan las colisiones y disminuye la velocidad de transmisión

¿Cómo funciona?

- Escucha previa (Carrier Sense)** Antes de transmitir, la estación **verifica si el canal está libre**.
- Transmisión (Multiple Access)** Si el canal está libre, la estación **envía los datos**.
- Colisión (Collision Detection)** Si otra estación también comienza a transmitir al mismo tiempo, se produce una **colisión**.
- Detección de colisión** Mientras transmite, la estación **compara lo que envía con lo que recibe**. Si detecta diferencias, significa que hay una colisión.

La computadora (tarjeta de red) **mide lo que pasa en el cable físico mientras transmite**. **Mientras transmite, la estación también "lee" el cable (lo que está recibiendo)**

- Acción tras la colisión** La estación:
 - Detiene la transmisión.
 - Envía una **señal de interferencia** para notificar a las demás estaciones.
 - **Espera un tiempo aleatorio (Backoff exponencial)** antes de intentar nuevamente.
- Contención y transmisión** El canal pasa por ciclos:
 - **Contención**: las estaciones intentan transmitir, pueden colisionar.
 - **Transmisión**: una estación logra enviar.
 - **Reposo**: nadie transmite.

Ejemplo con colisión

- En el tiempo **t0**, una estación A termina de transmitir.
- Otras estaciones B y C también quieren transmitir.
- Ambas detectan el canal libre y **empiezan a enviar al mismo tiempo**.
- Se produce una **colisión**.
- Ambas estaciones **detienen su transmisión** y **esperan un tiempo aleatorio diferente** para volver a intentarlo.

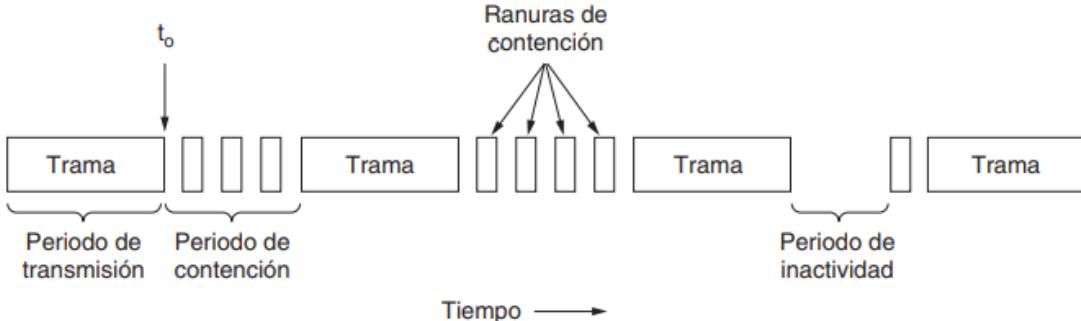


Figura 4-5. CSMA/CD puede estar en estado de contención, de transmisión o inactivo.

COLISIONES

Las colisiones ocurren con frecuencia y forman parte del funcionamiento normal de una red de área local (LAN). A medida que una red aumenta de tamaño, aumenta también la probabilidad de que se produzcan colisiones. En este caso, el tamaño se refiere no solo al número de estaciones, sino también a la longitud de los medios de transmisión.

Debido a las largas colas, aumentan las probabilidades de que una estación haya enviado una señal pero que esta no haya sido detectada por otra estación. Esto, a su vez, se traduce en más colisiones y ralentiza la red en su conjunto.

Prevención de Colisiones

- **En sistemas semidúplex:**
 - Las colisiones son inevitables, pero CSMA/CD las gestiona.
 - Demasiadas colisiones reducen la eficiencia (hasta un **30% de la velocidad real**).
- **Soluciones:**
 - Dividir el dominio de colisión usando **switches** o **puentes** (trabajan en capa 2 OSI).
 - Crear subredes para agrupar estaciones que se comunican frecuentemente.

4. Semidúplex vs. Dúplex Completo

- **Semidúplex:**
 - Comunicación en un solo sentido a la vez (ej. Ethernet con coaxial).
 - Requiere CSMA/CD para manejar colisiones.
- **Dúplex completo:**
 - Transmisión y recepción simultáneas (ej. cables de par trenzado o fibra óptica).
 - No hay colisiones → CSMA/CD es innecesario.

CSMA/CD vs. CSMA/CA

- **CSMA/CD:**
 - Usado en redes cableadas; **detecta** colisiones.
- **CSMA/CA:**

- Es un protocolo de acceso al medio utilizado principalmente en redes **inalámbricas** (como Wi-Fi) para **evitar colisiones** antes de que ocurran, a diferencia de CSMA/CD, que las detecta y corrige.
- Resuelve el **problema del nodo oculto**: **No se pueden detectar colisiones eficientemente** debido al **problema del nodo oculto** (dos dispositivos no "escuchan" las transmisiones del otro pero interfieren en un punto intermedio).
- La señal inalámbrica no es uniforme (interferencias, distancias variables).

Funcionamiento de CSMA/CA

El protocolo sigue estos pasos para evitar colisiones:

1. Escucha antes de transmitir (Carrier Sense)

- El dispositivo verifica si el canal está libre.
- Si está ocupado, espera un tiempo aleatorio antes de reintentar.

2. Espera un tiempo aleatorio (Backoff)

- Si el canal está libre, el dispositivo espera un período corto (**DIFS** en Wi-Fi) y luego un tiempo aleatorio adicional (para evitar que varios dispositivos transmitan al mismo tiempo).

3. Uso de ACK (Reconocimiento)

- El receptor envía un **ACK** (acknowledgment) para confirmar la recepción.
- Si no llega el ACK, se asume una colisión y se reintenta.

4. RTS/CTS (Opcional, para nodos ocultos)

- **RTS (Request to Send)**: El emisor solicita permiso para transmitir.
- **CTS (Clear to Send)**: El receptor autoriza la transmisión.
- Esto evita que otros nodos interfieran durante la comunicación.

Paso de token:

pasar un pequeño mensaje conocido como **token** de una estación a otra. 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

En un protocolo **token ring**, la topología de la red se utiliza para definir el orden en el que las estaciones envían información. Las estaciones están conectadas una con otra en un solo anillo. Así, el proceso de pasar el token a la siguiente estación consiste en recibir el token proveniente de una dirección y transmitirlo hacia la otra dirección

Cabe mencionar que no necesitamos un anillo físico para implementar el paso del token. El canal que conecta a las estaciones podría ser también un solo bus extenso. Así, cada estación puede usar el bus para enviar el token a la siguiente estación en la secuencia predefinida. Al poseer el token, una estación puede usar el bus para enviar una trama, como antes. A este protocolo se le conoce como **token bus**

“Token Ring”, que se estandarizó como IEEE 802.5

ETHERNET

Muchos de los diseños para las redes personales, locales y de área metropolitana se han estandarizado bajo el nombre de IEEE 802. Los sobrevivientes más importantes son el 802.3 (Ethernet) y el 802.11 (LAN inalámbrica)

Existen dos tipos de Ethernet:

Ethernet clásica, que resuelve el problema de acceso múltiple mediante el uso de las técnicas **Ethernet conmutada**, en donde los dispositivos llamados switches se utilizan para conectar distintas computadoras

IEEE 802.3: Ethernet

IEEE 802.3 define el estándar para Ethernet, que es la tecnología LAN (red de área local) más extendida en el mundo.

Aunque las primeras implementaciones de Ethernet usaban una topología de bus, las implementaciones modernas se basan en topologías en estrella, en las cuales cada nodo se conecta a un concentrador o switch.

- **Carrier Sense:** Cada estación “escucha” el medio antes de transmitir para asegurarse de que esté libre.
- **Multiple Access:** Todas las estaciones comparten el mismo medio de transmisión.
- **Collision Detection:** Si dos estaciones transmiten al mismo tiempo, se detecta la colisión; ambas detienen la transmisión y reintentan después de un tiempo aleatorio (mediante un algoritmo de retroceso exponencial).

IEEE 802.4: Token Bus

IEEE 802.4 define el estándar para redes basadas en la técnica de paso de token sobre una topología de bus. Aunque el medio físico es un bus, la “lista de orden” o secuencia lógica de acceso a los nodos se establece de forma tal que se simula una topología de anillo lógico sobre un medio compartido.

IEEE 802.5: Token Ring

IEEE 802.5 estandariza la tecnología Token Ring, Utiliza una topología en anillo, en la que los nodos se conectan de manera secuencial formando un circuito cerrado. Un token circula continuamente por el anillo. Cuando un nodo desea transmitir, debe esperar a recibir el token; una vez obtenido, puede enviar su información. Tras completar la transmisión, el token se libera para que otro nodo lo utilice.

IEEE 802.2: Logical Link Control (LLC)

IEEE 802.2 define la subcapa de Control de Enlace Lógico (LLC) dentro de la capa de enlace de datos. Mientras que la subcapa MAC (Media Access Control) se ocupa del acceso físico al medio y la

dirección física, la LLC se encarga de proporcionar servicios de enlace lógicos y de abstraer las particularidades de la tecnología de acceso al medio.

Ofrece los siguientes servicios:

- **Multiplexación:** Permite que múltiples protocolos de capa de red (por ejemplo, IP, IPX, AppleTalk) coexistan sobre el mismo medio físico, utilizando identificadores llamados Service Access Points (SAPs) para direccionar el tráfico adecuadamente.
- **Control y manejo de errores:** Proporciona funciones de control de enlace, como la identificación de tramas y, en algunos casos, control de errores
- **Conexión orientada y no orientada:** Puede ofrecer servicios tanto de transmisión orientada a conexión como de transmisión sin conexión

Puentes (Bridges)

Definición y función:

Un puente es un dispositivo de red que opera en la capa de enlace de datos. Su función principal es interconectar dos segmentos de red (por ejemplo, dos LAN) y filtrar el tráfico de datos para reducir el dominio de colisión. Esto se logra mediante la inspección de las direcciones MAC contenidas en las tramas que circulan por la red.

Conmutadores (Switches)

Definición y función:

El conmutador es una evolución del puente. Se puede considerar como un puente multipuerto, diseñado para interconectar múltiples dispositivos dentro de una misma red (LAN) de forma eficiente. Al igual que los puentes, los switches mantienen una tabla de direcciones MAC para saber por dónde reenviar las tramas. Cada puerto del switch constituye un dominio de colisión independiente.

Redes de Alta Velocidad: FDDI y Fast Ethernet

FDDI (Fiber Distributed Data Interface)

Definición y características:

Tecnología de alta velocidad: FDDI es un estándar para redes de área local que utiliza fibra óptica como medio físico

FDDI implementa una topología de anillo dual, lo que significa que cuenta con dos anillos de fibra. Este diseño permite una alta disponibilidad y redundancia: si un anillo falla, el otro puede seguir operando.

Fast Ethernet es el conjunto de estándares Ethernet que operan a 100 Mbps, una mejora significativa respecto al Ethernet original a 10 Mbps. Se implementa principalmente sobre cableado de par trenzado. En configuraciones tradicionales de Fast Ethernet que utilizan hubs, se sigue utilizando CSMA/CD. Sin embargo, en las implementaciones modernas con switches y enlaces full-duplex, las colisiones son prácticamente inexistentes, lo que maximiza el rendimiento de la red.

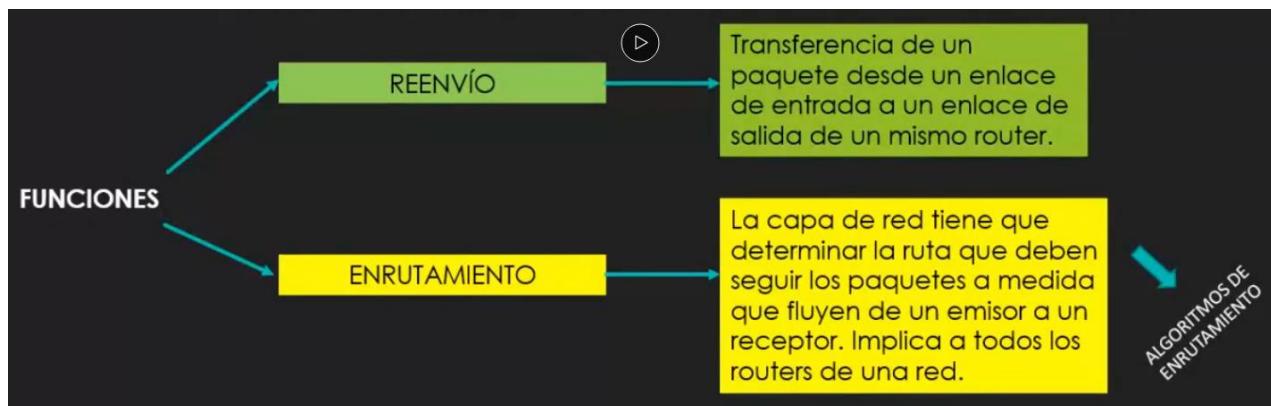
El protocolo de subcapa MAC de la Ethernet clásica

Una dirección MAC (Media Access Control) es un identificador único asignado a la interfaz de red de un dispositivo. La dirección MAC estándar se compone de **48 bits** (6 octetos) y se suele representar en notación hexadecimal

El formato utilizado para enviar tramas. Primero viene un Preámbulo de 8 bytes, cada uno de los cuales contiene el patrón de bits 10101010. Después vienen dos direcciones, una para el destino y una para el origen. Cada una de ellas tiene una longitud de 6 bytes. El primer bit transmitido de la dirección de destino es un 0 para direcciones ordinarias y un 1 para direcciones de grupo. Las direcciones de grupo permiten que varias estaciones escuchen en una sola dirección.

RUTEO, ESTABLECIMIENTO DE SESIONES Y EJECUCIÓN DE APLICACIONES

Unidad 5: Capa de Red



Conceptos preliminares

Direccionamiento IP

- Hoy en día se utilizan **IPv4** (32 bits) y **IPv6** (128 bits).
- Una dirección IP identifica de manera lógica y jerárquica a una **interfaz de red** de un dispositivo.
- El organismo **ICANN** administra la asignación de rangos de direcciones IP.
- Ejemplo IPv4: 158.26.45.1 (4 octetos de 8 bits cada uno → valores 0 a 255).
- Ejemplo IPv6: 2001:0db8:85a3:0000:0000:8a2e:0370:7334 (128 bits en hexadecimal, agrupados en 8 bloques separados por :).

Rangos extremos en IPv4:

- Mínimo: 0.0.0.0 (dirección no asignada).
- Máximo: 255.255.255.255 (broadcast).
- Total, aproximado: **4,3 mil millones de direcciones posibles**.



PROTOCOLO IP

El protocolo de IP (Internet Protocol) es la base fundamental de la Internet. Porta datagramas de la fuente al destino. Estructura los paquetes para su envío.

- **IPv4**
- **IPv6**

IPv4 es la versión original que fue lanzada en 1983. Sin embargo, su formato de 32 bits solo permite ~4,3 mil millones de direcciones únicas, que no pueden servir a las necesidades del mundo moderno.

Para hacer frente a la falta de direcciones IPv4 únicas (y hacer algunos otros cambios técnicos), se creó el IPv6. IPv6 utiliza un formato de direcciones de 128 bits que ofrece 1.028 veces más direcciones únicas que IPv4.

CARACTERÍSTICAS PROTOCOLO IP

- Protocolo orientado a **no conexión**.
- **Fragmenta paquetes** si es necesario. Tamaño máximo del paquete de 65635 bytes. Depende del tipo de enlace.
- Direccionamiento mediante **direcciones lógicas IP de 32 bits**.
- Si un paquete no es recibido, este permanecerá en la red durante un tiempo finito. **Tiene un contador**.
- Realiza el "**mejor esfuerzo**" para la distribución de paquetes.
- Es **independiente de los medios de enlace**.

IPv4 y su estructura básica:

- Consta de 4 números llamados octetos que van del 0 al 255 separados por un punto.
- Cada empresa de Internet tiene asignado un rango. **ICANN**
- La dirección **X.X.X.1** generalmente se le asigna al router

CONFIGURACIÓN DE UNA RED LOCAL

Cuando conectás una computadora a una red (por ejemplo, el WiFi del hogar o una red de oficina), necesitas ciertos datos para poder comunicarse con otros equipos e Internet.

Esa configuración puede hacerse **manualmente** (vos escribís los valores) o **automáticamente** (el router los asigna).

DHCP (Dynamic Host Configuration Protocol)

- Es un **servidor** (casi siempre el **router**) que **asigna automáticamente** una dirección IP a cada dispositivo que se conecta.
- También le da los otros datos necesarios: **máscara de red, puerta de enlace y DNS**.

- Cada vez que te conectás, puede darte un número IP diferente.

Por ejemplo:

- Hoy → 192.168.0.15
- Mañana → 192.168.0.21

Así evitás tener que configurar cada equipo a mano.

DNS (Domain Name System)

- Es un **sistema de nombres** que traduce direcciones web como www.google.com a números IP (por ejemplo 142.250.190.68).
- Es como la **agenda telefónica de Internet**: vos buscás por nombre, y el DNS te dice qué número IP corresponde.
- Estos servidores son dados por tu **proveedor de Internet (ISP)**, aunque podés usar otros públicos (como los de Google o Cloudflare).

Puerta de enlace (Gateway)

- Es la **dirección IP del router** o dispositivo que te conecta al exterior (Internet).
- Funciona como la **salida de tu red local**.

Por ejemplo: si tu router es 192.168.0.1, esa es tu puerta de enlace.

Máscara de subred

Cada dispositivo en una red tiene una **dirección IP**, por ejemplo: 192.168.1.10

Esa dirección tiene **dos partes**:

1. **Parte de red** → identifica a la red a la que pertenece el equipo.
2. **Parte de host** → identifica al dispositivo dentro de esa red.

La **máscara de subred** sirve para **separar** esas dos partes.

Le dice al sistema: “hasta acá es la red, y desde acá son los hosts”.

Ejemplo simple

Supongamos esta configuración:

Dato	Valor
Dirección IP	192.168.1.10
Máscara de subred	255.255.255.0

Si lo escribimos en binario:

IP: 11000000.10101000.00000001.00001010

Máscara: 11111111.11111111.11111111.00000000

- Los **1** en la máscara (los 255) indican la **parte de red**
- Los **0** (el último grupo) indican la **parte de host**

Entonces:

- Parte de red: 192.168.1
- Parte de host: 10

Eso significa que **todos los dispositivos cuya IP empiece con 192.168.1.** están en la **misma red local** y pueden comunicarse entre sí **sin pasar por un router**.

CLASES DE RED — CAPACIDAD — MÁSCARA

Las direcciones IP se agrupan por “**clases**”, que originalmente definían el tamaño de la red (esto hoy se maneja más con subredes, pero sigue sirviendo para entenderlo).

Clase	Rango de IP	Cantidad de hosts posibles	Máscara	Uso típico
Clase A	1.0.0.0 – 126.255.255.255	≈ 16 millones de hosts	255.0.0.0	Redes muy grandes (grandes empresas, gobiernos)
Clase B	128.0.0.0 – 191.255.255.255	≈ 65 mil hosts	255.255.0.0	Redes medianas (universidades, instituciones)
Clase C	192.0.0.0 – 223.255.255.255	254 hosts	255.255.255.0	Redes pequeñas (hogares, oficinas)

La capa de red

<https://www.youtube.com/watch?v=BhxdLoc8a-0>

<https://www.youtube.com/watch?v=r-K5yZe2Ea8>

Introducción:

La **capa de red** es la responsable de la **transmisión de extremo a extremo** de los paquetes, desde el origen hasta el destino final, incluso si esto requiere múltiples saltos a través de enrutadores intermedios.

A diferencia de la capa de enlace de datos, que solo mueve tramas a través de un mismo medio físico (como un cable), la capa de red opera a una escala global, manejando la complejidad de toda la red.

Para cumplir su función, la capa de red debe:

1. **Conocer la topología** de la red (todos los enrutadores y enlaces).
2. **Elegir las rutas** óptimas para evitar la congestión y equilibrar la carga.
3. **Resolver los problemas** que surgen cuando el origen y el destino están en redes diferentes.

El protocolo IP de Internet es el ejemplo principal de esta capa.

Función principal

Es responsable de **mover paquetes entre dispositivos** a través de distintas redes. Para ello:

- Asigna direcciones lógicas (IP) a los dispositivos.
- Encapsula los datos recibidos de la capa de transporte en **paquetes (datagramas)**.
- Determina la mejor ruta mediante el **enrutamiento**.
- Entrega los paquetes al destino, donde se realiza el **desencapsulamiento**.

Funciones básicas:

1. DIRECCIONAMIENTO:

- **Qué es:** Asignar direcciones lógicas únicas a los dispositivos en una red. La dirección más común en la capa de red es la **dirección IP**.
- **Para qué sirve:** Permite identificar de manera única el origen y el destino de un paquete de datos en una red, similar a cómo una dirección postal identifica una casa.
- Primero, la Capa de red debe proveer un mecanismo para direccionar estos dispositivos finales. Si las secciones individuales de datos deben dirigirse a un dispositivo final, este dispositivo debe tener una dirección única. Cuando se agrega esta dirección a un dispositivo, al dispositivo se lo denomina **host**.

2. ENCAPSULAMIENTO:

- **Qué es:** El proceso de tomar los datos de la capa de transporte (segmentos) y añadirles una **cabecera** que contiene información crucial de la capa de red, como las direcciones IP de origen y destino. El paquete resultante se llama **datagrama**.
- **Para qué sirve:** La cabecera proporciona la información necesaria para que los routers puedan enrutar el paquete correctamente a través de la red.
- Cuando se crea un paquete, el encabezado debe contener, entre otra información, la dirección del host hacia el cual se lo está enviando. A esta dirección se la conoce como **dirección de destino**. El encabezado de la Capa 3 también contiene la dirección del host de origen. A esta dirección se la llama **dirección de origen**.

3. ENRUTAMIENTO (o RUTEO):

- **Qué es:** El proceso de determinar la **mejor ruta** que debe tomar un paquete para viajar desde el origen hasta el destino a través de una red de routers interconectados.
- **Para qué sirve:** Los routers utilizan tablas de enrutamiento y algoritmos para tomar decisiones sobre el camino más eficiente, evitando congestión y fallos en la red.

- El enrutamiento es el proceso realizado por un dispositivo intermedio llamado **router** para enviar paquetes a la red de destino desde su red de origen.

4. DESENCAPSULAMIENTO:

- **Qué es:** El proceso inverso al encapsulamiento. Cuando un paquete llega a su destino final (o al dispositivo correcto dentro de una red), la capa de red **elimina su cabecera**.
- **Para qué sirve:** Una vez removida la información de la capa de red, los datos restantes (el segmento de la capa de transporte) se pasan a la capa superior correspondiente para su posterior procesamiento.
- El host destino examina la dirección de destino para verificar que el paquete fue direccionado a ese dispositivo. Si la dirección es correcta, el paquete es desencapsulado por la capa de Red.

Comutación de paquetes de almacenamiento y reenvío

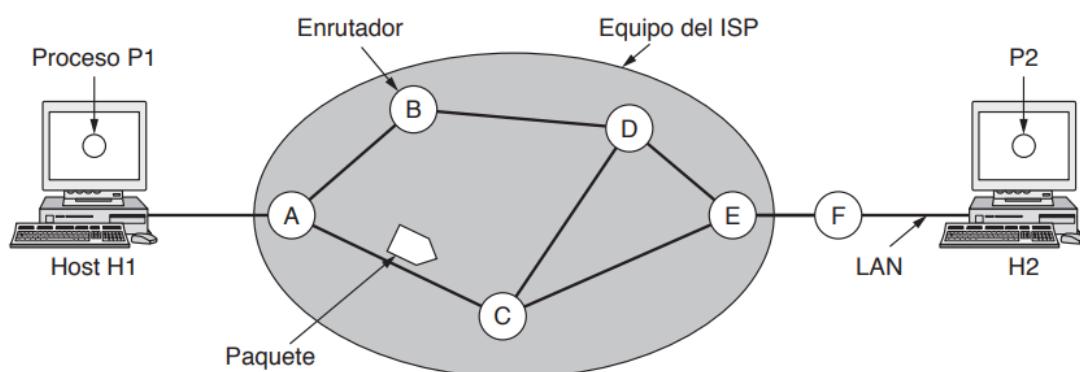


Figura 5-1. El entorno de los protocolos de la capa de red.

1. Dos Grupos de Equipos:

- **Equipo del ISP (dentro del óvalo):** Son los enrutadores y las líneas de transmisión que forman la **red troncal** del proveedor. Es la infraestructura central de Internet.
- **Equipo del Cliente (fuera del óvalo):** Son los dispositivos del usuario final (como H1) y los enrutadores de borde (como F que gestiona la LAN de una oficina). Aunque son propiedad del cliente, para el estudio de los algoritmos de enrutamiento, el enrutador del cliente (F) **se considera parte de la red lógica del ISP** porque ejecuta los mismos protocolos.

2. Tipos de Conexión:

- **Conexión Directa (H1):** Un host puede conectarse directamente a un router del ISP (ej: mediante DSL o fibra óptica).
- **Conexión mediante LAN (H2):** Un host puede estar en una red local (LAN) gestionada por un enrutador del cliente, que a su vez tiene un enlace (ej: línea arrendada) hacia la red del ISP.

3. Mecanismo Fundamental: Comutación de Almacenamiento y Envío (Store-and-Forward):

- **Paso 1: Transmisión.** Un host envía un paquete completo al enrutador más cercano (el "primer salto").
- **Paso 2: Almacenamiento y Verificación.** El router **recibe y almacena** el paquete completo. Luego, **verifica su integridad** (ej: con la suma de verificación) antes de procesarlo.
- **Paso 3: Reenvío (Forwarding).** El router consulta su tabla de enrutamiento para decidir el siguiente salto y **transmite el paquete completo** al siguiente router en el camino.
- Este proceso de almacenar, verificar y luego reenviar se repite en cada enrutador a lo largo de la ruta hasta que el paquete llega a su host destino.

Servicios proporcionados a la capa de transporte

El diseño de los servicios que la **capa de red** debe proporcionar a la **capa de transporte** se guía por tres objetivos fundamentales:

1. **Independencia tecnológica:** Los servicios no deben depender de la tecnología específica de los enrutadores.
2. **Aislamiento de la capa de transporte:** Esta capa no debe verse afectada por la cantidad, tipo o topología de los enrutadores.
3. **Direccionamiento uniforme:** Debe existir un plan de numeración coherente para las direcciones de red, válido tanto para redes LAN como WAN.

Relación entre capa de transporte y capa de red:

La capa de transporte **confía** sus segmentos de datos a la capa de red para que los entregue. La capa de red **sirve** a la capa de transporte proporcionándole el servicio de entrega.

- **Entrega garantizada:** este servicio garantiza que el paquete terminara por llegar a destino
- **Entrega garantizada con retardo de tiempo:** garantiza la entrega del paquete, pero además tendrá un límite de retardo especificado de host a host.

Para un flujo de paquetes:

- **Entrega de paquetes en orden:** los paquetes llegan a destino en el orden en el que fueron enviados
- **Ancho de banda mínimo garantizada:** mientras el host emisor transmite los bits a una velocidad sea menos a la velocidad bit específico no se perderá ningún paquete
- **Fluctuación máxima garantizada:** el intervalo de tiempo transcurrido entre dos paquetes sucesivos, en el emisor es igual al intervalo de tiempo que transcurre entre su respectiva recepción en el destino

- **Servicio de seguridad:** utiliza una clave secreta de sesión solo conocida por el host de origen y destino. Cifra la carga útil de todos los datagramas que están siendo enviados al host de destino.

Aspecto	Capa de Transporte	Capa de Red	Explicación
Unidad de Datos	Segmento (TCP) / Datagrama (UDP)	Paquete (IP)	La capa de transporte le pasa un segmento a la capa de red. Esta le añade su propia dirección (la IP), creando un paquete.
Responsabilidad	Comunicación de extremo a extremo (entre aplicaciones)	Salto a salto (entre routers)	La capa de transporte se preocupa de que los datos lleguen <i>a la aplicación correcta</i> en el destino. La red se preocupa de que el paquete llegue <i>al dispositivo correcto</i> .
Direccionamiento	Puertos (identifican la aplicación)	Direcciones IP (identifican el dispositivo)	La transporte dice " <i>lleva esto a WhatsApp en el celular de María</i> ". La red dice " <i>lleva esto al celular con la IP 192.168.1.5</i> ".
Confianza	Puede ser confiable (TCP) o no confiable (UDP)	Es inherentemente no confiable (Best-effort)	La capa de red (IP) hace su mejor esfuerzo por entregar el paquete, pero no garantiza nada. Si se necesita confiabilidad (que no se pierdan datos), la capa de transporte (TCP) debe implementarla ella misma.

1. **La capa de transporte es el cliente.** La capa de red es el servicio.
2. **La de transporte le dice a la red:** "Toma estos datos y entrégalos a esta dirección IP. Yo me encargo de todo lo demás (orden, pérdida de datos, control de flujo)".

3. **La red responde:** "Lo intentaré. Aquí tienes tu paquete con la dirección IP puesta. Ahora lo enviaré por la mejor ruta que encuentre."

Posturas:

El debate central entre los diseñadores es cómo lograr estos objetivos. La controversia se centra en la **naturaleza fundamental del servicio**:

- **Servicio orientado a conexión:** Similar a una llamada telefónica, establece un camino dedicado (una "conexión") antes de enviar los datos, garantizando orden y confiabilidad.
- **Servicio sin conexión:** Similar al envío de cartas por correo, cada paquete se maneja de forma independiente y puede seguir rutas distintas, sin garantías previas de entrega.

Esta "batalla campal" define la arquitectura central de la red, eligiendo entre un modelo que garantiza el orden y la fiabilidad (orientado a conexión) frente a uno que prioriza la simplicidad y la robustez (sin conexión). Internet se decantó por el **servicio sin conexión** con el protocolo IP.

1. Postura de la comunidad de Internet

- **Rol del enrutador:** solo debe mover paquetes, nada más.
- **Naturaleza de la red:** se asume que siempre será poco confiable, sin importar el diseño.
- **Responsabilidad:** los **hosts** (dispositivos finales) deben encargarse del **control de errores** y del **control de flujo**.
- **Servicio de red:**
 - Debe ser **sin conexión**.
 - Solo se necesitan dos primitivas: **SEND PACKET** y **RECEIVE PACKET**.
 - No se requiere ordenamiento ni control de flujo en la red (eso lo hacen los hosts).
 - Cada paquete viaja de manera **independiente** y lleva la dirección de destino completa.
- **Principio clave:** esto es un ejemplo del **argumento extremo a extremo (end-to-end argument)**, formulado por Saltzer et al. (1984).

2. Postura de las compañías telefónicas

- **Red confiable y orientada a conexión.**
- Se basan en el éxito del sistema telefónico mundial, que lleva más de 100 años garantizando calidad.
- La **calidad del servicio (QoS)** es lo más importante.
- Aseguran que, sin conexiones, es muy difícil garantizar servicios de **tiempo real** (voz, video, etc.).

3. Historia y evolución

- En los **años 70 y 80**, se usaron ampliamente redes orientadas a conexión como **X.25 y Frame Relay**.
- Internet y ARPANET apostaron por el enfoque **sin conexión** con **IP**, que terminó imponiéndose.
- **ATM (Asynchronous Transfer Mode)**, diseñado para reemplazar IP con un modelo orientado a conexión, fracasó en la masificación y quedó limitado a nichos.
- Hoy, **IP domina incluso en redes telefónicas**.

4. Situación actual

- Aunque IP es **sin conexión**, Internet ha ido incorporando **características orientadas a conexión** para mejorar la calidad del servicio.
- Ejemplos modernos de estas tecnologías:
 - **MPLS (Multiprotocol Label Switching)**.
 - **VLANs (Redes de Área Local Virtuales)**.

Tipos de servicios

Servicio sin conexión (connectionless)

- **Cómo funciona:** cada mensaje (paquete) se envía por separado, sin reservar un “camino” fijo en la red.
- La red **no garantiza** que llegue en orden ni sin errores.
- Cada paquete debe llevar la dirección completa del destino.
- Si algo se pierde o se repite, el **receptor o el emisor se encargan de arreglarlo** (no la red).

Ejemplo: mandar varias **cartas por correo común**.

- Cada sobre lleva la dirección completa.
- Puede que lleguen desordenadas o alguna se pierda.
- El cartero solo entrega, no revisa ni corrige nada.

En Internet: **IP** funciona así.

Servicio con conexión (connection-oriented)

- **Cómo funciona:** antes de enviar datos, se establece una **conexión** (como una llamada telefónica).
- Los paquetes siguen un **camino fijo** durante toda la comunicación.
- La red garantiza que lleguen en orden y sin errores.
- El control lo hace la propia red.

Ejemplo: hacer una **llamada de teléfono**.

- Primero se establece la conexión.
- Mientras hablas, la línea está reservada para vos.
- Lo que decís llega en orden y en tiempo real.

En Internet: **TCP** funciona así (sobre IP).

Implementación del servicio sin conexión

Organización de la red:

- En el **servicio sin conexión**, los paquetes (llamados **datagramas**) viajan **de manera independiente** y cada uno se enruta por separado.
- **No se necesita** establecer una conexión previa. **No se necesita una configuración por adelantado**
- La red se llama **red de datagramas**.
- **Comparación:**
 - **Sin conexión** → datagramas (como en Internet con IP).
 - **Con conexión** → se crea un **círculo virtual (VC)** antes de enviar los datos (similar al sistema telefónico).
- **Funcionamiento de la red de datagramas (ejemplo):**

1. El host origen divide un mensaje grande en varios paquetes (ej: 4 paquetes).
2. Cada paquete se envía a un enrutador de inicio.
3. Los enrutadores usan **tablas de enrutamiento** para decidir la salida de cada paquete.
4. Los paquetes pueden seguir **la misma ruta o rutas diferentes**, según las decisiones de los enrutadores.
5. Finalmente, todos llegan al host destino, que se encarga de reconstruir el mensaje.

- **Características clave:**
 - Un mismo mensaje puede llegar **por diferentes rutas**.
 - Los enrutadores deciden la ruta en base a sus **tablas de enrutamiento**, que pueden cambiar por congestión u otros factores.
 - El **algoritmo de enrutamiento** es el que gestiona estas decisiones.
- **Ejemplo real:**
 - **IP (Internet Protocol)** → cada paquete lleva la dirección IP completa de destino.
 - **IPv4**: direcciones de 32 bits.
 - **IPv6**: direcciones de 128 bits.

Ejemplo

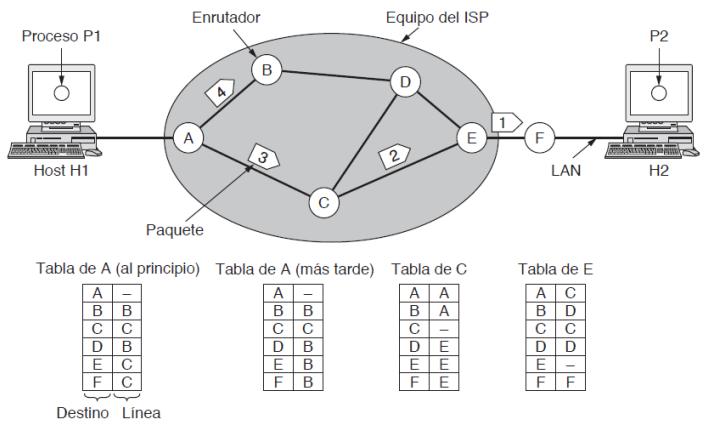


Figura 5-2. Enrutamiento dentro de una red de datagramas.

- En A, los paquetes 1, 2 y 3 se almacenan unos momentos, después de haber llegado por el enlace entrante y de haber comprobado sus sumas de verificación.
- Después cada paquete se reenvía de acuerdo con la tabla de A, por el enlace de salida a C dentro de una nueva trama.
- Después, el paquete 1 se reenvía a E y después a F. Cuando llega a F, se envía dentro de una trama a H2 a través de la LAN.
- Los paquetes 2 y 3 siguen la misma ruta.
- Sin embargo, ocurre algo diferente con el paquete 4. Cuando llega a A se envía al enrutador B, aun cuando también está destinado a F. Por alguna razón, A decidió enviar el paquete 4 por una ruta diferente a la de los primeros tres paquetes. Tal vez se enteró de que había alguna congestión de tráfico en alguna parte de la ruta ACE y actualizó su tabla de enrutamiento, como se muestra bajo la leyenda “más tarde”.

Implementación del servicio orientado a conexión - Red de circuitos virtuales

1. Idea principal

- Se evita elegir una nueva ruta para cada paquete.
- Al establecer la conexión, se define una **ruta fija** de origen a destino.
- Esa ruta se guarda en las **tablas de los enrutadores** y se usa durante toda la conexión.
- Al liberar la conexión, se elimina el circuito virtual.

2. Identificación de paquetes

- Cada paquete lleva un **identificador de conexión** que indica a qué circuito virtual pertenece.
- Los enrutadores pueden cambiar este identificador en los paquetes de salida para evitar conflictos.

3. Conmutación mediante etiquetas

- El proceso de reemplazar identificadores también se conoce como **conmutación mediante etiquetas**.

- **MPLS (Multiprotocol Label Switching)** es un ejemplo de este enfoque:
 - Los paquetes IP llevan un **encabezado MPLS** con una etiqueta de 20 bits.
 - El ISP puede establecer conexiones de largo plazo para administrar mejor el tráfico.
 - Se usa para mejorar la **calidad del servicio (QoS)** y otras tareas de gestión de tráfico.

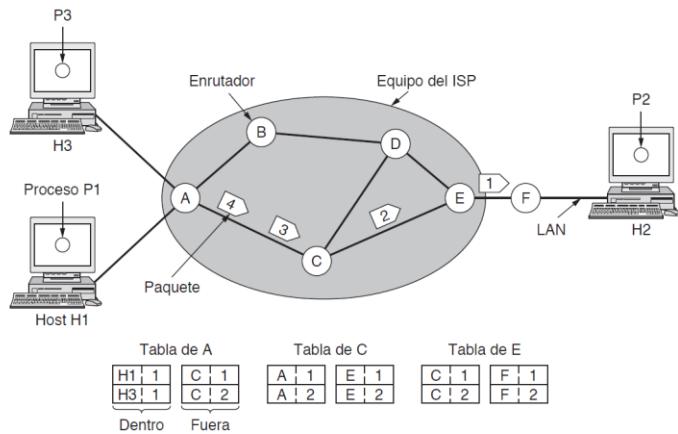


Figura 5-3. Enrutamiento dentro de una red de circuitos virtuales.

El host *H1* ha establecido una conexión 1 con el host *H2*.

- Esta conexión se recuerda como la primera entrada en cada una de las tablas de enrutamiento.
- La primera línea de la tabla A indica que si un paquete con el identificador de conexión 1 viene de *H1*, se enviará al enrutador *C* y se le dará el identificador de conexión 1.
- De manera similar, la primera entrada en C enruta el paquete a *E*, también con el identificador de conexión 1.

Si *H3* también desea establecer una conexión con *H2*.

- Elige el identificador de conexión 1 (debido a que está iniciando la conexión y a que ésta es su única conexión) y le indica a la red que establezca el circuito virtual. Esto nos lleva a la segunda fila de las tablas.
- Observe que aquí surge un problema, pues, aunque A sí puede saber con facilidad cuáles paquetes de conexión 1 provienen de *H1* y cuáles provienen de *H3*, *C* no puede hacerlo. Por esta razón, A asigna un identificador de conexión diferente al tráfico de salida para la segunda conexión.

Comparación entre las redes de circuitos virtuales y las redes de datagramas

Asunto	Red de datagramas	Red de circuitos virtuales
Configuración del circuito.	No necesaria.	Requerida.
Direccionamiento.	Cada paquete contiene la dirección de origen y de destino completas.	Cada paquete contiene un número de CV corto.
Información de estado.	Los enrutadores no contienen información de estado sobre las conexiones.	Cada CV requiere espacio de tabla del enrutador por cada conexión.
Enrutamiento.	Cada paquete se enruta de manera independiente.	La ruta se elige cuando se establece el CV; todos los paquetes siguen esa ruta.
Efecto de fallas del enrutador.	Ninguno, excepto para paquetes perdidos durante una caída.	Terminan todos los CVs que pasaron por el enrutador defectuoso.
Calidad del servicio.	Difícil.	Fácil si se pueden asignar suficientes recursos por adelantado para cada CV.
Control de congestión.	Difícil.	Fácil si se pueden asignar suficientes recursos por adelantado para cada CV.

Figura 5-4. Comparación entre las redes de datagramas y de circuitos virtuales.

1. Tiempo y configuración

- **Circuitos virtuales:** requieren fase de configuración (consume tiempo y recursos), pero luego los enrutadores procesan los paquetes más rápido usando solo el número de circuito.
- **Datagramas:** no necesitan configuración, pero cada paquete requiere análisis de dirección más complejo.

2. Direcciones y sobrecarga

- **Datagramas:** usan direcciones de destino largas (significado global), lo que puede generar sobrecarga en paquetes cortos.
- **Circuitos virtuales:** usan números de circuito más cortos y simples.

3. Uso de tablas en enrutadores

- **Datagramas:** requieren una entrada para cada destino posible.
- **Circuitos virtuales:** solo necesitan una entrada por circuito.
- Pero: los paquetes de configuración de circuitos también usan direcciones completas, como los datagramas.

4. Calidad del servicio y congestión

- **Circuitos virtuales:** permiten reservar recursos (búferes, ancho de banda, CPU) en la configuración, facilitando evitar congestiones y garantizar QoS.
- **Datagramas:** más difícil evitar congestión.

5. Tipo de tráfico

- **Circuitos virtuales:** poco eficientes para transacciones cortas (ej. validación de tarjeta de crédito).

- Útiles para conexiones largas o permanentes (ej. VPN entre oficinas).

6. Vulnerabilidad

- **Circuitos virtuales:** si falla un enrutador o línea, todos los circuitos que lo usan se interrumpen.
- **Datagramas:** más robustos; si un enrutador falla, solo se pierden los paquetes en cola y el tráfico puede redirigirse dinámicamente.

7. Flexibilidad del enrutamiento

- **Datagramas:** permiten balancear el tráfico porque las rutas pueden cambiar en cada transmisión.
- **Circuitos virtuales:** rígidos; dependen de la ruta establecida al inicio.

Síntesis:

- **Circuitos virtuales** → mejor control, QoS y eficiencia en conexiones largas, pero más vulnerables y con sobrecarga en configuraciones cortas.
- **Datagramas** → más simples, flexibles y resistentes a fallas, aunque con más sobrecarga por dirección y menos garantías de servicio.

Ancho de banda

- **Qué es:** la **capacidad máxima de transmisión de datos** de un canal de comunicación (por ejemplo, una conexión de Internet o un cable de red).
- **Unidad de medida:** normalmente en **bits por segundo (bps)**. Hoy se usan múltiplos como **Mbps** (megabits por segundo) o **Gbps** (gigabits por segundo).
- **Ejemplo:**
 - Si tenés 100 Mbps de ancho de banda, significa que **como máximo** podés transferir 100 megabits de información cada segundo.
- Ojo: ancho de banda **no siempre** es igual a velocidad real, porque influyen otros factores como congestión, latencia o limitaciones del servidor al que te conectás.

Analogía:

- Es como una **autopista**: el ancho de banda es la **cantidad de carriles**.
- Cuantos más carriles, más autos (datos) pueden pasar a la vez.

Puertos

- **Qué son:** en redes, un **puerto** es un número que identifica un servicio específico dentro de un dispositivo.
- Sirven para que, aunque todos los datos entran por la misma dirección IP, el sistema sepa a **qué aplicación** debe enviarlos.

- **Rango:**

- Van del **0 al 65535**.
- Los primeros (0–1023) se llaman **puertos bien conocidos** (*well-known ports*).

- **Ejemplos comunes:**

- **80** → HTTP (web).
- **443** → HTTPS (web segura).
- **25** → SMTP (correo saliente).
- **21** → FTP.
- **22** → SSH (acceso remoto seguro).

Analogía:

- La **dirección IP** sería como la **dirección de un edificio**.
- El **puerto** sería como el **número de departamento**: indica a qué puerta concreta tenés que tocar para que te atienda la aplicación correcta.

ALGORITMOS DE ENRUTAMIENTO

El algoritmo de enrutamiento es aquella parte del software de la capa de red responsable de decidir por cuál línea de salida se transmitirá un paquete entrante. Si la red usa datagramas de manera interna, esta decisión debe tomarse cada vez que llega un paquete de datos, dado que la mejor ruta podría haber cambiado desde la última vez. Si la red usa circuitos virtuales internamente, las decisiones de enrutamiento se toman sólo al establecer un circuito virtual nuevo. En lo sucesivo, los paquetes de datos simplemente siguen la ruta ya establecida. Este último caso a veces se llama enrutamiento de sesión, dado que una ruta permanece vigente durante toda una sesión (por ejemplo, durante una sesión a través de una VPN).

Enrutamiento vs Reenvío

Podemos considerar que un enrutador tiene dos procesos internos. Uno de ellos maneja cada paquete conforme llega, y después busca en las tablas de enrutamiento la línea de salida por la cual se enviará. Este proceso se conoce **como reenvío**. El otro proceso es responsable de llenar y actualizar las tablas de enrutamiento

- **Enrutamiento:** elegir qué rutas usar → mantiene y actualiza las tablas de enrutamiento.
- **Reenvío:** acción de enviar un paquete a la salida correspondiente → usa las tablas ya existentes.

Tipos de redes internas

- **Datagramas:** cada paquete puede tomar rutas distintas → decisión de enrutamiento en cada paquete.

- **Circuitos virtuales:** ruta se decide al inicio y se mantiene durante toda la sesión (enrutamiento de sesión).

Propiedades que debe cumplir un algoritmo de enrutamiento

- **Exactitud y sencillez.**
- **Robustez:** tolerar fallas de hardware, software y cambios de topología sin reiniciar toda la red.
- **Estabilidad:** converger rápido a un equilibrio en las rutas.
- **Equidad y eficiencia:** balancear justicia entre usuarios con el uso óptimo de recursos (a menudo en conflicto).

Objetivos de optimización

- Minimizar el **retardo promedio** de los paquetes.
- Maximizar la **velocidad de transferencia total** de la red.
- Reducir la **distancia** o el **número de saltos** → mejora retardo y uso de ancho de banda.

Clasificación de algoritmos de enrutamiento

1. No adaptativos (estáticos)

- No usan información del tráfico o topología actual (mediciones o estimaciones de tráfico).
- Rutas precalculadas fuera de línea y se descarga en los enrutadores al arrancar.
- No responde a fallas. Útiles cuando las rutas son claras y poco cambiantes.
-

2. Adaptativos (dinámicos)

- Ajustan rutas según cambios de topología o tráfico.
- Se diferencian en:
 - **De dónde obtienen información** (local, vecinos, toda la red).
 - **Cuando cambian rutas** (al ocurrir cambios o cada cierto tiempo).
 - **Métrica usada** (distancia, número de saltos, tiempo estimado).

Principio de optimización

Árbol sumidero y DAG

- **Árbol sumidero:** conjunto de rutas óptimas desde todos los orígenes hacia un destino.
 - No tiene ciclos → cada paquete llega en un número finito de saltos.
- **No es único:** pueden existir varios árboles con las mismas longitudes de rutas.

- Si permitimos todas las rutas posibles → se obtiene un **DAG (gráfico acíclico dirigido)**, también sin ciclos.

Problemas en la práctica

- Los **enlaces y enrutadores pueden fallar o recuperarse** → distintos enrutadores pueden tener visiones distintas de la topología.
- Duda: ¿cada enrutador obtiene la info por sí mismo o de otros? (tema que se trata después).

Principio de optimización (Bellman, 1957)

- Si el enrutador **J** está en la ruta óptima de **I → K**, entonces la ruta óptima de **J → K** forma parte de esa misma ruta.
- Consecuencia:
 - El conjunto de todas las rutas óptimas hacia un destino forma un **árbol sumidero con raíz en ese destino**.

Final:

- Desarrollar dos estáticos (corto, inundación) y dos dinámico (vector distancia o enlace) para host móviles (es dinámico)
- Con un dinámico bien dado apruebo el punto, estudiar bien, los dos primeros.

Algoritmo de la ruta más corta (Estatico)

Para encontrar las **rutas óptimas** en una red, se representa la red como un **grafo**:

- **Nodos:** enrutadores.
- **Arcos:** enlaces de comunicación entre ellos.

El objetivo es calcular la **ruta más corta** entre dos enrutadores según un **criterio (métrica)** elegido.

Métricas posibles para medir una ruta

La “longitud” de una ruta no siempre significa distancia física. Se puede medir de distintas formas:

- **Número de saltos** (cantidad de enrutadores intermedios).
- **Distancia geográfica:** recorrido físico
- **Peso:** Menor costo total (por distancia, tráfico, etc.).
- **Retardo promedio** (tiempo que tarda un paquete o transmision).
- **Costo de comunicación o congestión.** Menor carga o saturación de enlaces
- **Ancho de banda disponible o tráfico promedio.**

Según qué métrica usemos, la “ruta más corta” puede ser la más **rápida**, la de **menor costo**, o la de **menos saltos**.

Algoritmo de Dijkstra (1959)

Es el más conocido para hallar rutas más cortas en una red.

Funcionamiento:

1. Se parte de un **nodo origen** (por ejemplo, A).
2. Se marca su **distancia como 0** y se considera **permanente**.
3. Todos los demás nodos tienen distancia **infinita** al inicio.
4. Se examinan los **nodos adyacentes** y se actualizan sus etiquetas (distancia acumulada + enlace).
5. El nodo con **menor distancia tentativa** se marca como **permanente** y se convierte en el **nuevo nodo de trabajo**.
6. Se repite el proceso hasta que todos los nodos sean permanentes o se llegue al destino.

Cada nodo guarda:

- Su **distancia mínima** desde el origen.
- El **nodo anterior** por el cual se llegó (para reconstruir la ruta final).

Justificación del algoritmo

- Una vez que un nodo se marca como **permanente**, su distancia es la **mínima posible**.
- Si existiera una ruta más corta que no se haya descubierto, el algoritmo la habría probado antes (por tener menor peso acumulado).

Dirección del cálculo

- El algoritmo puede iniciarse desde el **origen (s)** o desde el **destino (t)**, porque en un grafo **no dirigido**, las rutas $s \rightarrow t$ y $t \rightarrow s$ son iguales.
- Si se comienza desde el destino, se etiqueta con **antecesores** en lugar de sucesores, y luego la ruta final se invierte.

Inundación (Flooding - Estática)

Definición

La **inundación** es una técnica de **enrutamiento simple y local** en la que **cada enrutador**, al recibir un paquete, **lo reenvía por todas sus líneas de salida, excepto por la que lo recibió**.

Así, el paquete se “inunda” por toda la red.

La inundación siempre selecciona la ruta más corta debido a que selecciona todas las rutas posibles en paralelo

Funcionamiento

1. Cuando un paquete llega a un enrutador, este **duplica el paquete** y lo envía a **todos los vecinos** posibles.
2. El proceso se repite en cada enrutador.
3. Sin control, esto genera **copias infinitas**, por eso se agregan mecanismos para **limitar la propagación**:

Mecanismos de control

- **Contador de saltos (Hop Count o TTL):**

- Cada paquete lleva un número que **se reduce en 1 en cada salto**.
- Cuando llega a **0**, el paquete se **descarta**.
- Evita que el paquete recorra la red indefinidamente.

Lo ideal es inicializar el contador de saltos con la longitud de la ruta entre el origen y el destino. Si el emisor desconoce el tamaño de la ruta, puede inicializar el contador para el peor caso; a saber, el diámetro total de la red

- **Números de secuencia:**

- El **emisor original** pone un número de secuencia único a cada paquete.
- Cada enrutador guarda **qué paquetes ya ha visto**.
- Si llega un paquete repetido, **no se reenvía**.
- Se usa un **contador k** para no guardar todos los números, solo el más alto ya recibido.

Problemática

- Genera **muchos paquetes duplicados**, lo que:
 - Satura los enlaces.
 - Consumo ancho de banda.
 - Puede colapsar la red si no hay límites.
- No es eficiente para redes grandes ni para tráfico normal.

Mejoras

- **Contador de saltos:** limita el número de reenvíos.
- **Registro de paquetes vistos:** evita duplicación.
- **Inundación selectiva:**
 - Los enrutadores **descartan ramas innecesarias**, reenviando solo por las líneas que **probablemente llevan al destino**.
 - Reduce la cantidad de copias.

Usos de la inundación

Aunque es ineficiente, la inundación tiene **aplicaciones útiles**:

1. Redes militares o de emergencia:

- Si varios enrutadores son destruidos, los paquetes **igual encuentran una ruta disponible**, si existe.
- Es **muy robusta** y confiable.

2. Difusión de información (broadcast):

- Cuando un mensaje debe llegar a **todos los nodos**, la inundación **garantiza la entrega**.

3. Redes inalámbricas:

- Las señales se propagan naturalmente a todos los dispositivos dentro del alcance, lo que **simula la inundación**.

4. Base para otros algoritmos:

- Algunos algoritmos de enrutamiento más complejos (como **Link State**) usan la inundación para **distribuir información de estado**.

5. Referencia teórica:

- Sirve como **modelo ideal** con el cual se comparan otros algoritmos, ya que **siempre encuentra la ruta más corta** (al probar todas las rutas en paralelo).
- La inundación también se puede usar como métrica con la que se pueden comparar otros algoritmos de enrutamiento

Enrutamiento por vector de distancia (Distance Vector Routing - Dinamico)

<https://www.youtube.com/watch?v=bAQvWvpuhzY>

Es **uno de los métodos más antiguos y simples** que usan los routers para decidir por dónde enviar los paquetes de datos en una red.

Idea general:

Cada router tiene una **tabla de enrutamiento** donde guarda:

- Qué destinos existen (otros routers o redes),
- Cuál es la **mejor distancia** para llegar a cada uno (por ejemplo, en cantidad de saltos o tiempo de retardo),
- **Y por qué camino (vecino)** debe enviar los datos para llegar allí.

Esa tabla se llama **vector de distancia** porque:

- “Vector” = lista de destinos con sus distancias,
- “Distancia” = número de saltos o retardo estimado.

Cómo funciona el algoritmo

Cada router:

1. Conoce la distancia a sus vecinos directos.

(Por ejemplo: a mi vecino A me toma 10 ms llegar).

2. Intercambia periódicamente su tabla con los vecinos.

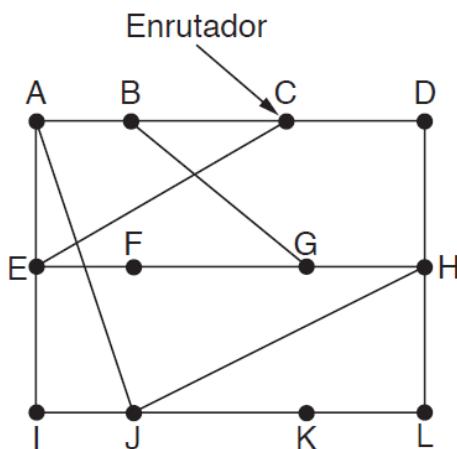
(Cada cierto tiempo les dice: “estos son los retardos que tengo hacia todos los destinos que conozco”).

3. Actualiza su tabla comparando las tablas de los vecinos.

Si un vecino conoce un camino más corto hacia un destino, el router actualiza su tabla para usar ese vecino como ruta preferida.

4. Repite el proceso hasta que todos los routers tienen la mejor ruta para todos los destinos (esto se llama **convergencia).**

Ejemplo simple



Nuevo retardo estimado desde J

A	A	I	H	K	Línea
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 -
K	24	22	22	0	6 K
L	29	33	9	9	15 K
	El	El	El	El	

retardo JA es de 8 retardo JI es de 10 retardo JH es de 12 retardo JK es de 6

Vectores recibidos de los cuatro vecinos de J

Nueva tabla de enrutamiento para J

(a)

(b)

Figura 5-9. (a) Una red. (b) Entrada de *A*, *I*, *H*, *K* y la nueva tabla de enrutamiento para *J*.

Este proceso de actualización se ilustra en la figura 5-9. En la parte (a) se muestra una red. Las primeras cuatro columnas de la parte (b) muestran los vectores de retardo recibidos de los vecinos del enrutador *J*. A indica que tiene un retardo de 12 mseg a *B*, un retardo de 25 mseg a *C*, un retardo de 40 mseg a *D*, etcétera. Suponga que *J* ha medido o estimado el retardo a sus vecinos *A*, *I*, *H* y *K* en 8, 10, 12 y 6 mseg, respectivamente.

Supón que el router **J** quiere saber cómo llegar al router **G**:

- J sabe que:

- H está a 12 ms,
 - A está a 8 ms,
 - I está a 10 ms,
 - K está a 6 ms.
- De sus vecinos recibe información:
 - A dice que llega a G en 18 ms,
 - I dice que llega a G en 31 ms,
 - H dice que llega a G en 6 ms,
 - K dice que llega a G en 31 ms.

Entonces J calcula:

- $A \rightarrow G = 8 + 18 = \mathbf{26 \text{ ms}}$
- $I \rightarrow G = 10 + 31 = \mathbf{41 \text{ ms}}$
- $H \rightarrow G = 12 + 6 = \mathbf{18 \text{ ms}}$
- $K \rightarrow G = 6 + 31 = \mathbf{37 \text{ ms}}$

Elige la menor: 18 ms (por H).

Así actualiza su tabla: “para ir a G, mando por H”.

Ventajas

- Es **simple** de implementar.
- Funciona bien en redes pequeñas.
- Permite que los routers se “autoajusten” sin configuración manual.

Desventaja principal: el problema del conteo al infinito

“reacciona rápido a las buenas noticias, pero es lento con las malas”

Qué pasa:

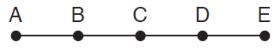
Cuando una ruta se **cae** (por ejemplo, un router o enlace deja de funcionar), las “malas noticias” **tardan mucho** en propagarse.

Por qué:

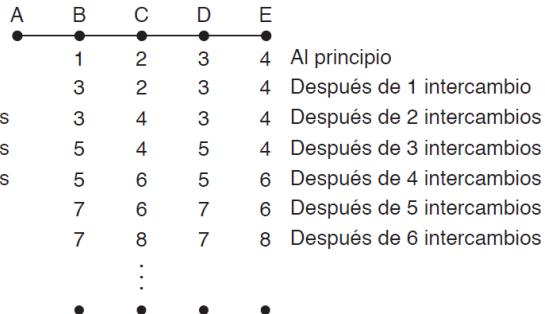
Supón que A, B, C, D y E están conectados en línea.

En el momento del primer intercambio, B se entera de que su vecino de la izquierda tiene un retardo de cero hacia A. B crea entonces una entrada en su tabla de enrutamiento para indicar que A está a un salto de distancia hacia la izquierda. Los demás enrutadores aún detectan que A está desactivado. En este punto, las entradas de la tabla de enrutamiento para A se muestran en la segunda fila de la figura 5-10(a). Durante el siguiente intercambio, C se entera de que B tiene una ruta a A de longitud 1, por lo que actualiza su tabla de enrutamiento para indicar una ruta de longitud 2, pero D y E no se enteran de las buenas nuevas sino hasta después. Como es evidente, las buenas noticias se difunden a razón de un salto por intercambio. En una red cuya ruta

mayor tiene una longitud de N saltos, en un lapso de N intercambios todo mundo sabrá sobre los enlaces y enrutadores que acaban de revivir.



(a)



(b)

Si A se apaga:

1. B deja de oír a A y piensa “ya no tengo ruta a A”. Registra un retardo infinito.
2. Pero C dice: “yo sí puedo llegar a A” (sin saber que su ruta pasa por B, que a su vez depende de A).
3. B le cree, y actualiza su tabla: “puedo llegar a A por C con distancia 3”.
4. C ve eso, y actualiza la suya: “ahora llego a A con distancia 4”.

Así, **se van sumando distancias sin fin** (3, 4, 5, 6...) hasta que llegan a un valor máximo que se define como “infinito”.

Por eso se llama **problema del conteo al infinito**.

Soluciones parciales

Existen algunas reglas para reducir el problema, aunque no lo eliminan por completo:

- **Split horizon (horizonte dividido)**: un router **no anuncia a un vecino** una ruta que **aprendió de ese mismo vecino**.
- **Poison reverse (envenenamiento inverso)**: el router **anuncia la ruta con distancia infinita** al vecino del cual la aprendió, para cortar el ciclo.

Estas estrategias están descritas en el estándar **RFC 1058** (usado en el protocolo **RIP**, Routing Information Protocol).

Enrutamiento por estado del enlace

<https://www.youtube.com/watch?v=FTLfOUZuurU>

Es un método que usan los routers para **construir un mapa completo de la red** y luego calcular las **rutas más cortas** hacia todos los destinos.

A diferencia del vector de distancia (que solo sabe lo que dicen sus vecinos), en el **estado del enlace** **cada router conoce toda la topología** de la red, igual que si tuviera un mapa completo.

Por eso es **más rápido, más preciso y más estable**.

Los 5 pasos básicos del enrutamiento por estado del enlace

1. Descubrir a los vecinos

Cuando un router se enciende, manda mensajes **HELLO** por cada una de sus conexiones.

Estos mensajes sirven para preguntar:

“¿Quién está conectado a mí?”

Los routers vecinos responden con su nombre o dirección.

Así, el router aprende **quiénes son sus vecinos directos**.

2. Medir el costo o distancia de cada enlace

El router calcula un **costo o métrica** para cada enlace.

Ese costo representa qué tan “caro” es usar ese camino.

Ejemplos:

- Un enlace rápido (1 Gbps) puede tener **costo 1**.
- Uno más lento (100 Mbps) puede tener **costo 10**.
- también puede basarse en **retardo (ms)** o **distancia física**.

Cuanto **menor el costo, mejor el enlace**.

3. Crear el paquete de estado del enlace

Una vez que el router sabe quiénes son sus vecinos y cuánto cuesta llegar a ellos, crea un **paquete de estado del enlace (Link State Packet, LSP)**.

Ese paquete incluye:

- Su nombre (identificador del router),
- Un **número de secuencia** (para saber si es nuevo o viejo),
- Una **edad** (para que caduque con el tiempo),
- La lista de sus vecinos y el **costo a cada uno**.

Ejemplo de paquete de estado del enlace del router A:

- Router: A
- Vecinos:
 - B → costo 5
 - C → costo 10

4. Enviar (inundar) los paquetes por toda la red

Aquí viene la parte clave del algoritmo:

Cada router **envía su paquete a todos sus vecinos**, y ellos **lo reenvían a los suyos**, hasta que **todos los routers** tengan una copia del paquete de cada uno.

Este proceso se llama **inundación (flooding)**.

Para evitar duplicados o errores, se usa:

- El **número de secuencia**,
- Y el **campo de edad** (cuando llega a 0, el paquete se descarta).
- Si un router recibe un paquete nuevo (número de secuencia más alto), lo reenvía.
Si recibe uno duplicado o viejo, lo descarta.

Resultado: **todos los routers terminan con la misma base de datos de topología**, sabiendo cómo está conectada toda la red.

5. Calcular las rutas más cortas (algoritmo de Dijkstra)

Una vez que todos tienen el mapa completo, **cada router ejecuta el algoritmo de Dijkstra**, que busca el **camino más corto (o de menor costo)** hacia todos los demás routers.

Así genera su **tabla de enrutamiento final**:

Destino | Costo | Siguiente salto

A | 0 | -

B | 5 | B

C | 7 | B

D | 10 | C

Luego empieza a reenviar paquetes usando esas rutas.

Protocolos que usan este método

1. OSPF (Open Shortest Path First)

- Es el más usado en redes IP modernas.
- Divide la red en áreas para optimizar el cálculo.
- Usa inundación controlada y routers designados para evitar tráfico excesivo.

2. IS-IS (Intermediate System to Intermediate System)

- Similar a OSPF, pero puede manejar **múltiples protocolos** (IP, IPX, AppleTalk...).
- Muy usado por **proveedores de Internet grandes (ISP)**.

Problemas y soluciones

Aunque es más estable, el protocolo debe manejar bien algunos detalles:

- **Números de secuencia**: evitan paquetes viejos o duplicados.
- **Campo de edad**: borra información obsoleta si un router se apaga.
- **Confirmaciones**: aseguran que cada router reciba los paquetes.
- **Retrasos controlados**: los routers esperan un poco antes de reenviar, para evitar saturar la red.

Perfecto Aquí tienes la transcripción organizada, con formato de apuntes para estudiar y con **negrita** en los conceptos clave. La estructuré en secciones claras para que puedas repasarla fácilmente.

Apuntes: Subnetting (Subredes en Redes IP)

<https://www.youtube.com/watch?v=nTxFWPUj-8&t=1s>

<https://youtu.be/SHbBso63X38?si=SRs6RCaaPYr8s3FL>

VPN: <https://youtu.be/CWy3x3Wux6o?si=r5qxL1GKX53feb4Q>

¿Qué es el Subnetting?

Definición: dividir una **red grande** en **redes más pequeñas**, llamadas **subredes**.

- **Utilidad:**
 - Mejor **organización** de dispositivos.
 - Separar por **departamentos**.
 - Aumentar la **seguridad**.
 - **Optimizar** el uso de direcciones IP.

Ejemplo:

Tienes una red de **256 IPs** (por ejemplo, una /24).

Podés dividirla en redes más pequeñas:

- Una para **cámaras**
- Otra para **móviles**
- Otra para **administración**
- Otra para **invitados**

Máscara de Red

- Cada dirección IP tiene una **máscara de red**, que indica:
 - Cuántos bits son para la **red**.
 - Cuántos bits son para los **hosts** (dispositivos finales).
- **Host:** cualquier equipo con IP (PC, cámara, celular, router, etc).

Ejemplo:

- Red: **192.168.1.0/24**
- Los **primeros 24 bits** son de red.
- Los **últimos 8 bits** son de host.
- En total: **32 bits** (4 grupos de 8).
- De 256 IPs, **solo 254 son usables**:
 - 1 IP es de **red**.
 - 1 IP es de **broadcast**.

Cómo crear subredes (paso a paso)

Ejemplo 1

Red original: 192.168.1.0/24

Queremos: 4 subredes

1. Hallamos el número de bits necesarios:

- ($2^1 = 2$) subredes → insuficiente
- ($2^2 = 4$) subredes → correcto
- **n = 2 bits** para red extra.

2. Nueva máscara:

- $/24 + 2 = /26$
- Resultado: **4 subredes /26**

3. Direcciones por subred:

- Bits para host = $32 - 26 = 6$
- ($2^6 = 64$) direcciones por subred
- 2 reservadas → **62 IPs usables**

Ejemplo 2

Red original (Supernet): 172.16.1.0/24

Queremos: 8 subredes

1. Bits necesarios:

- ($2^3 = 8$) → **3 bits extra**

2. Nueva máscara:

- $/24 + 3 = /27$

3. IPs por subred:

- Bits host = $32 - 27 = 5$
- ($2^5 = 32$) direcciones
- 2 reservadas → **30 IPs usables**

4. Número mágico:

- ($2^{\{bits_host\}} = 2^5 = 32$)
- Se suma al último octeto de la red para obtener las siguientes subredes.

Subredes resultantes /27

Nº	Dirección de red	Comentario
1	172.16.1.0/27	Subred 1
2	172.16.1.32/27	Subred 2

Nº	Dirección de red	Comentario
3	172.16.1.64/27	Subred 3
4	172.16.1.96/27	Subred 4
5	172.16.1.128/27	Subred 5
6	172.16.1.160/27	Subred 6
7	172.16.1.192/27	Subred 7
8	172.16.1.224/27	Subred 8

Cada una tiene **30 IPs usables**.

Analogía:

Una red /24 es una pizza con **256 porciones**.

Al dividirla en /27, obtienes **8 porciones más pequeñas** de **32 cada una** (30 usables).

Ejemplo 3

Red original (Supernet): 172.16.0.0/22

Queremos: 12 subredes

1. Bits necesarios:

- ($2^3 = 8$) (insuficiente)
- ($2^4 = 16$) → **4 bits extra**

2. Nueva máscara:

- $/22 + 4 = /26$

3. **Número mágico:**

- ($2^{\{32-26\}} = 2^6 = 64$)

4. **Subredes resultantes:**

- 172.16.0.0/26
- 172.16.0.64/26
- 172.16.0.128/26
- 172.16.0.192/26
- 172.16.1.0/26
- 172.16.1.64/26
- 172.16.1.128/26
- 172.16.1.192/26
- 172.16.2.0/26
- 172.16.2.64/26
- 172.16.2.128/26
- 172.16.2.192/26

- (hasta completar 16 subredes posibles)

Cada subred tiene **64 IPs totales, 62 usables**.

IPs usables

- Primera IP → dirección de **red**
- Última IP → **broadcast**
- IPs intermedias → **usables**

Ejemplo:

- Subred 1:
 - Red: 172.16.0.0
 - Usables: 172.16.0.1 – 172.16.0.62
 - Broadcast: 172.16.0.63

Aplicaciones

- En **empresas**: dividir por áreas → *staff, IT, marketing, invitados*.
- En **certificaciones**: tema clave para **CCNA, MTCNA**, y exámenes de redes.
- En **trabajo real**: optimiza IPs y mejora segmentación.

Concepto avanzado: VLSM

- **VLSM (Variable Length Subnet Mask)** = máscara de subred de longitud variable.
 - Permite **subredes de distinto tamaño**, según las necesidades.
- Ejemplo:
- Red de invitados → más IPs.
 - Red de gerencia → menos IPs.
- Mejora la **eficiencia** del direccionamiento.

Algoritmos de control de congestión.

https://www.youtube.com/watch?v=_q3gue_SL40

Concepto de Congestión

- **Congestión**: ocurre cuando **hay demasiados paquetes** en una red o en parte de ella, provocando **retardo o pérdida de paquetes** y una **degradación del desempeño**.
- **Causas**: exceso de tráfico, buffers llenos, o procesamiento lento de enrutadores.
- **Consecuencia**: la red entra en **congestión** y puede llegar al **colapso** si no se controla.

Responsabilidad de las Capas

- **Capa de red:** experimenta la congestión **directamente**; decide qué hacer con los **paquetes sobrantes**.
- **Capa de transporte:** puede **reducir la carga** que se envía a la red, siendo la forma **más efectiva** de control.
- Ambas **deben trabajar en conjunto** para manejar la congestión.

Evolución de la Congestión (Figura 5-21)

4. **Carga baja:** el **caudal útil** (paquetes/seg) es proporcional a la carga ofrecida.
5. **Carga cercana a la capacidad:** comienzan **pérdidas de paquetes y retrasos**.
6. **Colapso por congestión:**
 - Aumentar la carga **reduce drásticamente** el rendimiento.
 - Ejemplo:** en enlaces antiguos (56 kbps), los paquetes se retrasaban tanto que **expiraban o se descartaban**.
 - Retransmisiones innecesarias** agravan el problema, generando **duplicados** y desperdiциando ancho de banda.

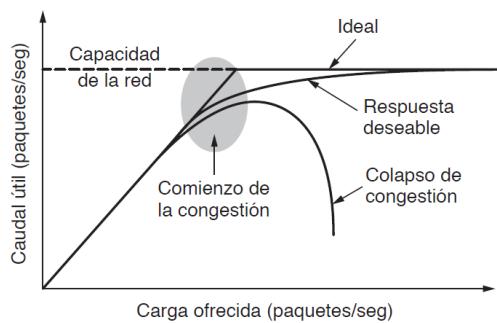


Figura 5-21. Con demasiado tráfico, el desempeño se reduce bruscamente.

Colapso por Congestión

- Ocurre cuando la **red sigue recibiendo tráfico más allá de su capacidad**, y los paquetes:
 - Se pierden o duplican.
 - Llegan **demasiado tarde** para ser útiles. Salen tan tarde, que cuando lo hacen ya no son útiles. El emisor entiende que nunca llegó, y pide duplicado.
- **Nagle (1987):** incluso con **memoria infinita**, la congestión **empeora** porque los paquetes se acumulan, exigen y se retransmiten múltiples veces. La cola se hace inmensa, y los que peor tiempo tienen son los últimos. Puede haber duplicados.

Soluciones y Estrategias

- **Redirección del tráfico:** distribuir las cargas para evitar embotellamientos.
- **Descarte de paquetes:** si toda la red está saturada, se deben **eliminar paquetes**.
- **Infraestructura más rápida:** aumentar el **ancho de banda** o la capacidad de procesamiento.

Diferencia entre Control de Congestión y Control de Flujo

Concepto	Control de Congestión	Control de Flujo
Objetivo	Evitar que la toda la red colapse.	Evitar que el receptor se sature.
Enfoque	Global (toda la red).	Punto a punto (emisor-receptor).
Causa principal	Exceso de tráfico total.	Diferencia de velocidad entre emisor y receptor.
Ejemplo	Red de 1 Mbps con muchos transmisores simultáneos.	Supercomputadora (100 Gbps) enviando a una PC (1 Gbps).

En ambos casos, la solución más común es **reducir la velocidad de transmisión**, pero la **razón es distinta**.

Etapas del Control de Congestión

1. **Prevención:** evitar que la congestión ocurra (planeamiento, control de tráfico).
2. **Detección y reacción:** actuar una vez que la congestión se presenta (reducción de caudal, descarte selectivo, etc.).
3. **Recuperación o reacción:** restablecer el rendimiento normal de la red lo mas ante posible, luego de que ya ocurrió.

1. Introducción a los Métodos de Control de Congestión

- La **congestión** surge cuando la **carga** excede temporalmente los **recursos** de la red (o parte de ella).
 - Soluciones principales: **aumentar recursos** (ej. más ancho de banda) o
 - **reducir carga** (ej. limitar tráfico).
- Buffer: es un espacio de memoria en el que se almacenan datos de manera temporal.
- Aplicación en **escalas de tiempo** (ver Figura 5-22): Preventiva (largo plazo, como meses) o reactiva (corto plazo, como milisegundos).

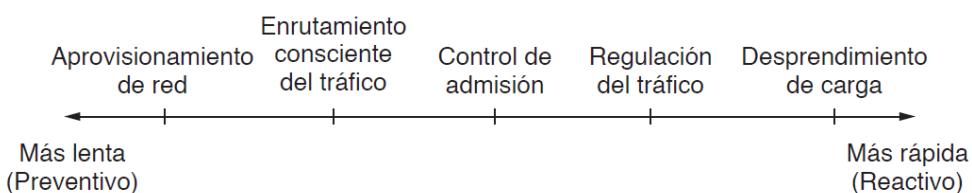


Figura 5-22. Escalas de tiempo de los métodos para el control de la congestión.

- Detección de congestión: Monitorear **carga promedio**, **retardo de encolamiento** o **pérdida de paquetes**. Números crecientes indican congestión inminente.
- Retroalimentación: Enrutadores deben notificar a **fuentes** (emisores) sin sobrecargar la red. Evitar oscilaciones: Ajustar **escala de tiempo** para respuestas oportunas (ni demasiado rápidas ni lentas).

2. Métodos Preventivos: Evitar la Congestión Antes de que Ocurra

- **Aprovisionamiento:** Construir/redimensionar la red para coincidir con **patrones de tráfico** a largo plazo (meses). Ejemplos:
 - Actualizar **enlaces** o **enrutadores** con alto uso regular.
 - Agregar recursos dinámicos: Activar **enrutadores de repuesto**, habilitar **líneas de respaldo** o comprar **ancho de banda** en mercado spot.
- **Enrutamiento Consciente del Tráfico (Traffic-Aware Routing):**

Si nosotros a lo largo del día tenemos una ruta por la cual estamos enviando la información, y llega una hora del día que sabemos que aumenta el tráfico, entonces cambiamos de ruta por otra con mas capacidad.

- Ajustar rutas dinámicamente según **carga cambiante** (ej. por zonas horarias). Dividir tráfico entre múltiples rutas.
- Ponderaciones de enlaces: Función de **ancho de banda fijo**, **retardo de propagación** y **carga variable** (retardo de encolamiento promedio).
- Riesgos: **Oscilaciones** si ponderaciones cambian rápido (ej. Figura 5-23: Tráfico alterna entre enlaces CF y EI, causando inestabilidad).

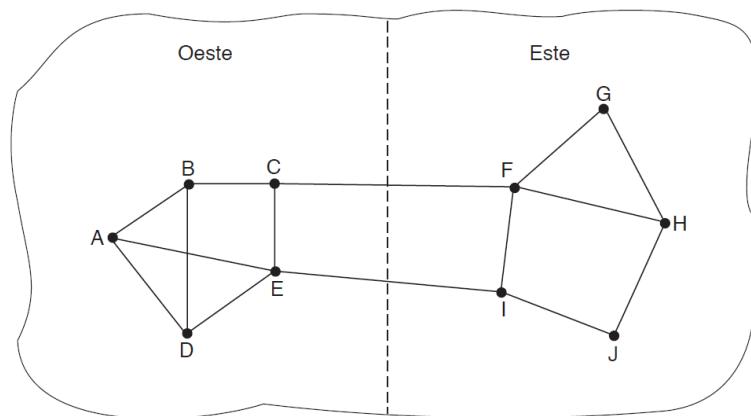


Figura 5-23. Una red en la cual las partes Este y Oeste se conectan mediante dos enlaces.

- Soluciones: **Enrutamiento multirayectoria** (esparcir tráfico), cambios lentos para convergencia (como en Gallagher, 1977). En Internet, ajustes manuales via **ingeniería de tráfico**.
- **Control de Admisión** (en redes de **circuitos virtuales**):

Cuando no es posible aumentar la capacidad, entonces reducimos la carga. En una red de circuitos virtuales se puede rechazar las nuevas conexiones si congestionan la red. Por ejemplo, si solamente puedo enviar 25 paquetes, y me llegan solicitudes de enviar más. Entonces rechazo porque estoy en mis límites. Esto tiene dos dificultades: a) identificar el comienzo de la congestión, y b) como informar a la fuente que reduzca la velocidad. Para ello a) los enrutadores deben poder monitorear la carga promedio, el retardo de encolamiento o la perdida de paquetes. Es decir, tener esta información a la mano para poder informar a la fuente cuando hay congestión

(Cuando y donde). B) Los enrutadores deben estar conectados directamente a la fuente, para que informen congestión y que parte de la red la tiene.

- Rechazar nuevas conexiones si provocan congestión. Mejor que admitir y empeorar.
- Evaluación: Basado en **descripción de tráfico** (tasa, forma). Usar **cubeta con goteo (leaky bucket)** o **cubeta con token (token bucket)** para capturar tasa promedio y ráfagas (detalles en sección 5.4).
- Opciones: Reservar capacidad (garantías QoS) o estimar estadísticamente (basado en mediciones pasadas, intercambiando riesgo por eficiencia).
- Combinado con enrutamiento: Evitar rutas congestionadas (ej. Figura 5-24: Redibujar topología omitiendo nodos saturados).

3. Métodos Reactivos: Lidiar con la Congestión Establecida

- **Regulación de Tráfico** (Evasión de Congestión):

- Enrutadores detectan congestión monitoreando **recursos**: Uso de enlaces, **búferes** (mejor opción, vía **retardo de encolamiento**), pérdidas (llega tarde).
- Cálculo de retardo: Usar **EWMA (Promedio Móvil Ponderado Exponencialmente)**:

$$d_{nueva} = \alpha d_{anterior} + (1 - \alpha)s$$

donde s = longitud de cola instantánea.

a filtra fluctuaciones).

- Retroalimentación a emisores: Identificar fuentes y notificar sin sobrecarga.
 - **Paquetes Reguladores**: Enrutador envía paquete de advertencia al origen (ej. SOURCE QUENCH en early Internet). Etiquetar original para evitar duplicados. Reducir tráfico (ej. 50%). Ignorar duplicados durante intervalo fijo.
 - **Notificación Explícita de Congestión (ECN)**: Etiquetar paquetes en encabezado (2 bits en IP). Destino retransmite marca al emisor en respuesta. Refinamiento de esquemas antiguos (Ramakrishnan y Jain, 1988). Ver Figura 5-25.

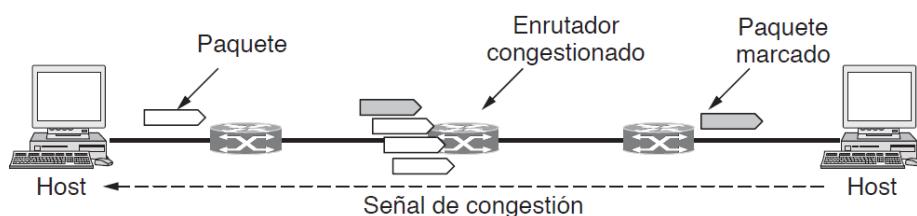


Figura 5-25. Notificación explícita de congestión.

- **Contrapresión de Salto por Salto**: Paquete regulador afecta cada salto reverso (ej. Figura 5-26). Alivio inmediato en punto de congestión, usando más búferes

upstream. Efectivo en distancias largas o altas velocidades (reduce retardo de señal).

- **Desprendimiento de Carga (Load Shedding):**

Descarta los paquetes que no se pueden entregar. Para ello se debe escoger una buena política para descartar los paquetes.

- Descartar paquetes cuando enrutadores están abrumados (último recurso, análogo a apagones eléctricos).
- Políticas inteligentes: Dependiendo de aplicación.
 - **Vino (Wine)**: Preferir paquetes viejos (mejor para transferencias de archivos).
 - **Leche (Milk)**: Preferir paquetes nuevos (mejor para medios en tiempo real).
- Cooperación: Marcar paquetes por importancia (ej. rutas de enrutamiento > datos; tramas completas MPEG > diferencias). Usar contabilidad/dinero para evitar abusos (permitir exceso si baja prioridad).

- **Detección Temprana Aleatoria (RED):**

- Descartar paquetes temprano (antes de búfer lleno) para señal implícita de congestión.
- Monitoreo: Promedio de longitud de cola (EWMA). Descartar fracción aleatoria si excede umbral (afecta más a emisores rápidos).
- Beneficios: Da tiempo a fuentes para reducir (explotando que TCP ve pérdidas como congestión). Mejor que descartar solo al llenar búferes.
- Variante: Usar con ECN para señales explícitas. Requiere ajuste (ej. fracción de descartes basada en emisores).

Interredes.

La **interconexión de redes** surge de la necesidad de **comunicar redes heterogéneas** (LAN, WAN, MAN, etc.) con distintos **protocolos**. No existe una sola tecnología universal, y siempre habrá **diversidad de redes**, lo que exige **mecanismos para integrarlas**.

Importancia de la interconexión

- El **valor de una red** crece con el número de conexiones posibles (**Ley de Metcalfe: N^2**).
- **Internet** (con “I” mayúscula) es el ejemplo máximo de una **interred global**, que permite la comunicación entre usuarios de distintas redes.
- La interconexión enfrenta **problemas de heterogeneidad** (protocolos, tamaños de paquete, QoS, seguridad, etc.) y **problemas de escala** al aumentar su tamaño.

Cómo difieren las redes

Las redes se diferencian por:

- **Técnicas de modulación, formatos de tramas y protocolos.** Capa física (No la vemos aquí)

- **Tamaño máximo de paquete (MTU), orientación a conexión o no, multidifusión, calidad de servicio (QoS) y seguridad.**
- Estas diferencias hacen difícil que los paquetes viajen de una red a otra sin modificaciones.

Aspecto	Algunas posibilidades
Servicio ofrecido.	Sin conexión vs. orientado a conexión.
Direccionamiento.	Distintos tamaños, plano o jerárquico.
Difusión.	Presente o ausente (también multidifusión).
Tamaño de paquete.	Cada red tiene su propio valor máximo.
Ordenamiento.	Entrega ordenada y desordenada.
Calidad del servicio.	Presente o ausente; muchos tipos distintos.
Confiabilidad.	Distintos niveles de pérdida.
Seguridad.	Reglas de privacidad, cifrado, etcétera.
Parámetros.	Distintos tiempos de expiración, especificaciones de flujo, etcétera..
Contabilidad.	Por tiempo de conexión, paquete, byte o ninguna.

Figura 5-38. Algunas de las diversas formas en que pueden diferir las redes.

Cómo se pueden conectar

Existen **dos formas básicas**:

1. **Traducir paquetes** entre redes (complejo y poco confiable).
2. **Agregar una capa común**, como hace **IP (Protocolo de Internet)**.

Cerf y Kahn (1974) propusieron una capa común (**TCP/IP**), que se convirtió en la base de la **Internet moderna**.

Los **enrutadores** operan en la **capa de red** y permiten la comunicación entre distintas redes, mientras que los **switches** y **puentes** trabajan en la **capa de enlace**.

Los enrutadores extraen el paquete y deciden el próximo salto según la dirección IP; los switches solo reenvían tramas basadas en la dirección MAC.

Tunelización (Tunneling)

Cuando dos redes iguales están separadas por una diferente (por ejemplo, **IPv6 sobre IPv4**), se usa la **tunelización**:

- El paquete original (IPv6) se **encapsula** dentro de otro (IPv4).
- Viaja “dentro de un túnel” hasta llegar al destino, donde se **extrae** y entrega al host.
- Este método crea **redes superpuestas (overlay)**, base de las **VPN (Redes Privadas Virtuales)**, que añaden **seguridad** al tráfico.

Enrutamiento entre redes

El **enrutamiento interred** es más complejo porque:

- Cada red puede usar **diferentes algoritmos de enrutamiento** (estado del enlace o vector de distancia).

- Existen **intereses comerciales y políticos** entre operadores o países.

Por eso se usa un modelo **jerárquico de dos niveles**:

- **Protocolo intradominio (Interior Gateway Protocol)**: dentro de cada red o **AS (Sistema Autónomo)**.
- **Protocolo interdominio (Exterior Gateway Protocol)**: entre redes; en Internet, este es el **BGP (Border Gateway Protocol)**.

Cada **AS** puede elegir sus rutas internas libremente y ocultar detalles de su topología.

Fragmentación de paquetes

Cada red o enlace impone un tamaño máximo a sus paquetes. Estos límites tienen varias razones, entre ellas:

1. El hardware (por ejemplo, el tamaño de una trama Ethernet).
2. El sistema operativo (por ejemplo, todos los búferes son de 512 bytes).
3. Los protocolos (por ejemplo, la cantidad de bits en el campo de longitud de paquete).
4. El cumplimiento de algún estándar (inter)nacional.
5. El deseo de reducir hasta cierto nivel las retransmisiones inducidas por errores.
6. El deseo de evitar que un paquete ocupe el canal demasiado tiempo.

El resultado de todos estos factores es que los diseñadores de redes no tienen la libertad de elegir cualquier tamaño máximo de paquetes que deseen. Cada red tiene un **tamaño máximo de paquete (MTU)**. Cuando un paquete es demasiado grande para una red intermedia, se debe **fragmentar**.

Existen dos métodos:

1. **Fragmentación transparente**: los enrutadores intermedios dividen y vuelven a ensamblar (ineficiente).
2. **Fragmentación no transparente (IP)**: cada fragmento viaja independiente y se **reensambla en el destino**.

Tiene varios problemas, sobrecarga el trabajo del host o enrutadores, y si se pierde un fragmento hecha a perder todo el trabajo.

Para evitar los problemas de fragmentación, en la Internet moderna se usa el **descubrimiento de MTU de la ruta**, donde los enrutadores **informan (mensaje de error) a la fuente** si un paquete es demasiado grande para adaptarlo automáticamente. Se repite el proceso hasta que tenga un tamaño que no esté en conflicto con los enrutadores que siguen.

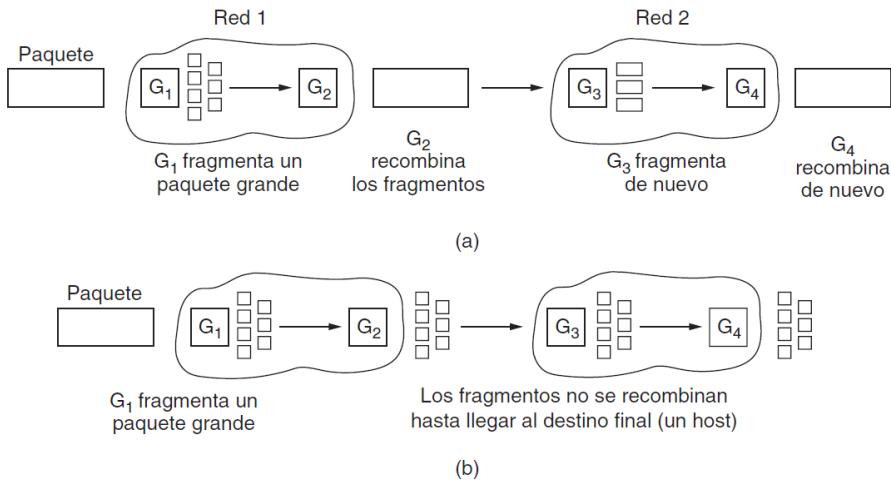


Figura 5-42. (a) Fragmentación transparente. (b) Fragmentación no transparente.

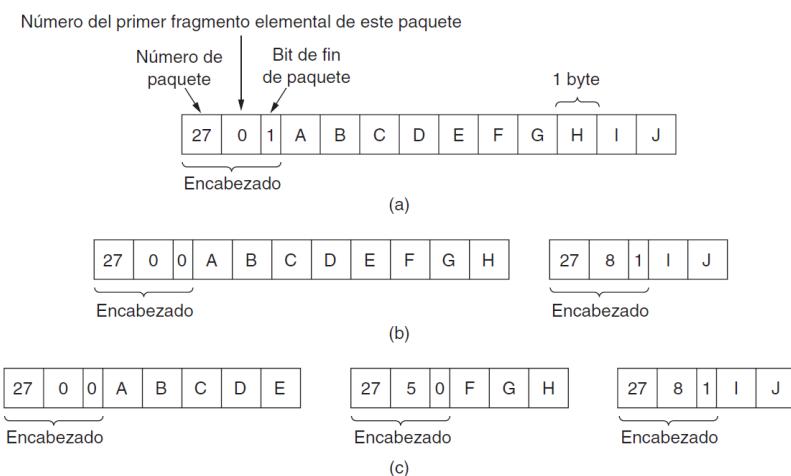


Figura 5-43. La fragmentación cuando el tamaño de datos elemental es de 1 byte. (a) El paquete original que contiene 10 bytes de datos. (b) Los fragmentos después de pasar por una red con un tamaño máximo de paquete de 8 bytes de carga útil más encabezado. (c) Fragmentos después de pasar a través de una puerta de enlace de tamaño 5.

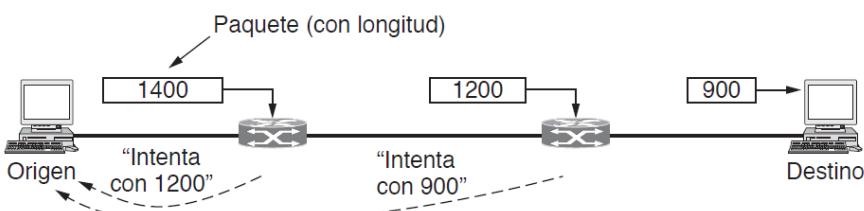


Figura 5-44. Descubrimiento de MTU de la ruta.

Capa de red en Internet.

Principios de Diseño

Internet se guía por principios clave para garantizar su éxito y escalabilidad, enumerados en el RFC 1958. Entre los más importantes se encuentran:

- Ser estricto cuando se envía y tolerante cuando se recibe.** Solo enviar paquetes que cumplan rigurosamente los estándares, pero esperar paquetes que tal vez no cumplan y tratar de lidiar con ellos.
- Pensar en la escalabilidad.** Evitar bases de datos centralizadas y distribuir la carga equitativamente.

- **Considerar el desempeño y el costo.** Si una red tiene un desempeño pobre o un costo exagerado, nadie la utilizará.

Estructura de Internet

Internet es un conjunto de **Sistemas Autónomos (AS)** interconectados. No tiene una estructura real, pero existen redes troncales principales de **alto ancho de banda y enrutadores rápidos**. Las redes más grandes se llaman **redes de Nivel 1**. Conectadas a estas troncales están los **ISP (Proveedores de Servicio de Internet), centros de datos y redes regionales**.

El Protocolo IP (Internet Protocol)

Es el "pegamento" que mantiene unida a Internet. Su trabajo es proporcionar un medio de **mejor esfuerzo (sin garantía)** para transportar paquetes de la fuente al destino.

- La capa de transporte divide los datos en **paquetes IP**.
- Los **enrutadores IP** reenvían cada paquete a través de Internet a lo largo de una ruta de un enrutador a otro.
- En el destino, la capa de red ensambla los paquetes para formar el datagrama original.

El protocolo IP versión 4 (IPv4)

Un datagrama IPv4 consiste en un **encabezado** y un **cuerpo o carga útil**.

- **Formato del Encabezado:**

- **Versión:** 4 para IPv4.
- **IHL (Longitud de Encabezado de Internet):** Indica la longitud del encabezado en palabras de 32 bits.
- **Servicio diferenciado (antes Tipo de servicio):** Distingue entre clases de servicio.
- **Longitud total:** Longitud total del datagrama (máximo 65,535 bytes).
- **Identificación, DF (No Fragmentar), MF (Más Fragmentos), Desplazamiento del fragmento:** Campos para manejar la **fragmentación** de paquetes.
- **Tiempo de vida (TTL):** Contador que limita la vida del paquete para evitar que anden vagando eternamente.
- **Protocolo:** Indica a qué protocolo de transporte (TCP, UDP, etc.) entregar el paquete.
- **Suma de verificación del encabezado:** Protección contra errores.
- **Dirección de origen y Dirección de destino:** Direcciones IP de 32 bits.
- **Opciones:** Campo de longitud variable para información adicional (rara vez usado).

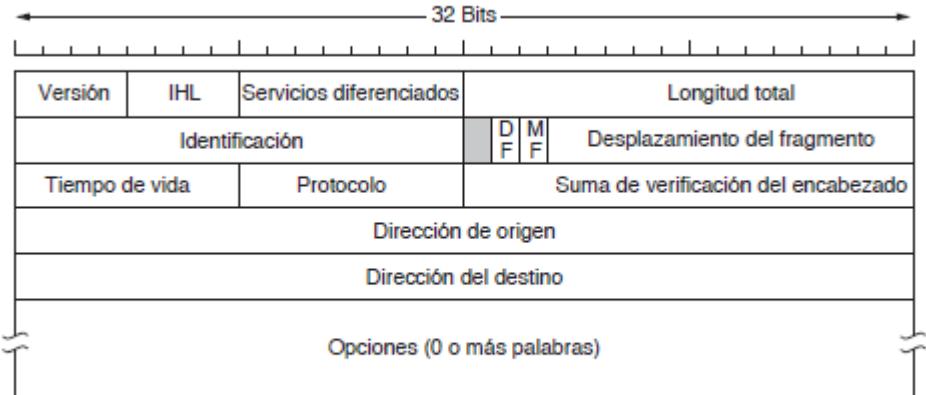


Figura 5-46. El encabezado de IPv4 (Protocolo de Internet).

Opción	Descripción
Seguridad.	Especifica qué tan secreto es el datagrama.
Enrutamiento estricto desde el origen.	Proporciona la ruta completa a seguir.
Enrutamiento libre desde el origen.	Proporciona una lista de enrutadores que no se deben omitir.
Registrar ruta.	Hace que cada enrutador adjunte su dirección IP.
Estampa de tiempo.	Hace que cada enrutador adjunte su dirección y su etiqueta de tiempo.

Figura 5-47. Algunas de las opciones del protocolo IP.

Direcciones IP

- Son **direcciones jerárquicas** de 32 bits, compuestas por una porción de **red** y una porción de **host**.
- Se escriben en **notación decimal con puntos** (ej: 128.208.2.151).
- Los bloques de direcciones se denominan **prefijos** y se escriben como DirecciónIP/LongitudDePrefijo (ej: 128.208.0.0/24).
- **Ventaja clave:** Los enrutadores pueden reenviar paquetes basándose solo en la porción de red, haciendo las tablas de enrutamiento más pequeñas.
- **Desventajas:** La dirección IP de un host depende de su ubicación y la jerarquía puede desperdiciar direcciones.

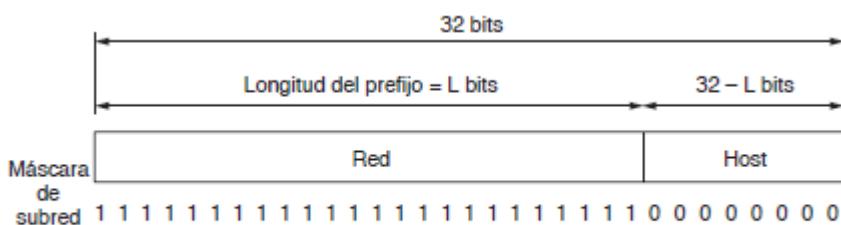


Figura 5-48. Un prefijo y una máscara de subred del protocolo IP.

Subredes

Permiten dividir un bloque de direcciones IP asignado en varias redes internas más pequeñas, actuando como una sola red hacia el exterior. Los enrutadores utilizan **máscaras de subred** para determinar a qué subred pertenece un paquete.

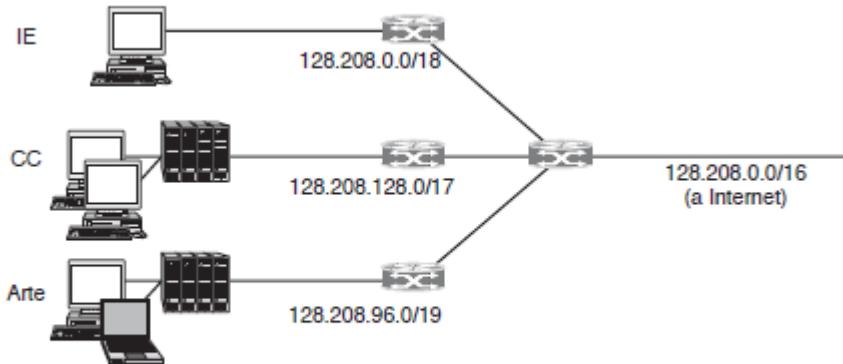


Figura 5-49. División de un prefijo IP en redes separadas mediante el uso de subredes.

CIDR (Enrutamiento Interdominio sin Clases)

- Técnica para **agregar varios prefijos pequeños en un solo prefijo más grande** (agregación de rutas).
- Reduce drásticamente el tamaño de las tablas de enrutamiento globales.
- Utiliza la regla del **prefijo más largo coincidente** para decidir la ruta, proporcionando flexibilidad.

Direccionamiento con Clases (Obsoleto)

Antes de CIDR, las direcciones IP se dividían en clases fijas (A, B, C, D, E), lo que llevó a un desperdicio masivo de direcciones, especialmente con las redes Clase B.

NAT (Traducción de Dirección de Red)

- Solución para la escasez de direcciones IPv4.
- Permite que una red interna use direcciones IP privadas (ej: 10.x.x.x, 192.168.x.x) y las traduzca a una (o unas pocas) direcciones IP públicas al salir a Internet.
- Funciona modificando los campos de dirección IP y **puertos TCP/UDP** en los paquetes.
- **Problemas:** Viola el modelo de extremo a extremo de Internet, complica algunas aplicaciones y introduce estado en la red.

IP versión 6 (IPv6)

Diseñado para reemplazar a IPv4, con las siguientes mejoras principales:

- **Direcciones más largas: 128 bits**, proporcionando un espacio de direcciones prácticamente ilimitado.
- **Encabezado simplificado:** Solo 7 campos (frente a 13 en IPv4), permitiendo un procesamiento más rápido.
- **Soporte mejorado para opciones:** Mediante **encabezados de extensión**.
- **Seguridad incorporada:** Autenticación y cifrado (aunque luego se modernizó para IPv4).
- **Mejor soporte para Calidad de Servicio (QoS):** Campo **Etiqueta de flujo**.

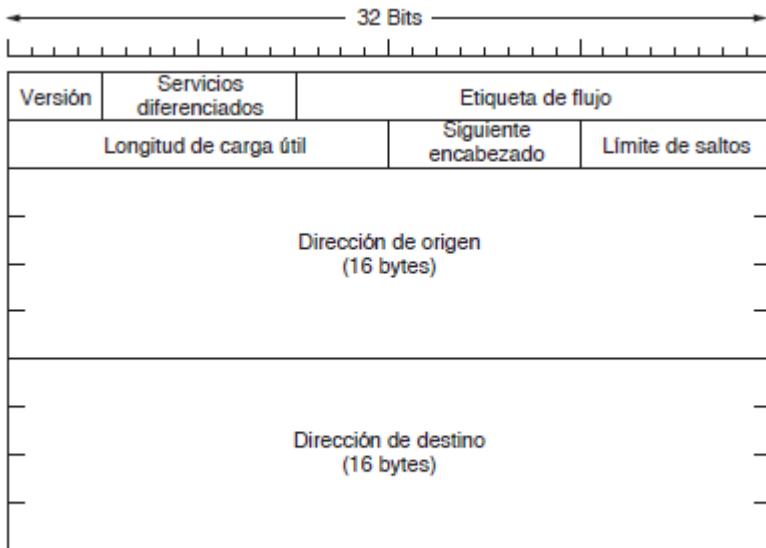


Figura 5-56. El encabezado fijo de IPv6 (requerido).

Protocolos de Control en Internet

- **ICMP (Protocolo de Mensajes de Control de Internet):** Usado por los enrutadores para informar errores y condiciones inesperadas (ej: destino inaccesible, tiempo excedido). Herramientas como ping y traceroute usan ICMP.
- **ARP (Protocolo de Resolución de Direcciones):** Se usa en redes locales (como Ethernet) para encontrar la dirección física (**MAC**) correspondiente a una dirección IP.
- **DHCP (Protocolo de Configuración Dinámica de Host):** Asigna automáticamente direcciones IP y otros parámetros de configuración a los hosts cuando se conectan a la red.

Conmutación mediante etiquetas y MPLS

- **MPLS (Conmutación Multiprotocolo mediante Etiquetas)** agrega una **etiqueta** a cada paquete. El reenvío se basa en esta etiqueta, no en la dirección IP, lo que lo hace muy rápido.
- Se considera un protocolo de **capa 2.5**.
- Se utiliza para **ingeniería de tráfico**, calidad de servicio y redes privadas virtuales.

Protocolos de Enrutamiento

- **OSPF (Abrir Primero la Ruta más Corta):** Protocolo de **estado del enlace** usado para el **enrutamiento intradominio** (dentro de un mismo Sistema Autónomo). Es dinámico, eficiente y soporta redes jerárquicas divididas en **áreas**.
- **BGP (Protocolo de Puerta de Enlace de Frontera):** Protocolo de **vector de ruta** usado para el **enrutamiento interdominio** (entre diferentes Sistemas Autónomos). Su principal preocupación son las **políticas** de enrutamiento (comerciales, de seguridad) más que la eficiencia pura.

Multidifusión de Internet

- Comunicación de **uno a muchos** usando direcciones IP **Clase D**.
- El protocolo **IGMP** gestiona la membresía a los grupos de multidifusión.

- Protocolos como **PIM** construyen árboles de enrutamiento para la multidifusión.

IP Móvil

Permite a un host móvil moverse entre redes sin cambiar su **dirección IP base**. Utiliza un **agente de base** en la red de origen y un **agente foráneo** en la red visitada para redirigir el tráfico mediante **tunelización**.

UNIDAD 6 CAPA DE TRANSPORTE

La capa de transporte intenta suministrar un servicio de transporte de datos que aísla las capas superiores de los detalles de implementación del transporte. Temas como la **confiabilidad del transporte** entre dos hosts es responsabilidad de la capa de transporte. Al proporcionar un servicio de comunicaciones, la capa de transporte **establece, mantiene y termina adecuadamente los circuitos virtuales**. Si se desea recordar esta capa en la menor cantidad de palabras posible, piense en **calidad de servicio y confiabilidad**.

Los procesos de capa de transporte se llevan a cabo entre la capa de la **capa de sesión** y la **capa de red** en el **modelo OSI**.

- Divide los datos recibidos desde la aplicación en segmentos, más fáciles de administrar (**Segmentación**)
- Agrega un encabezado para identificar cada segmento (incluyendo los números de puerto origen y destino), y poder reensamblar todos los segmentos después. (**Reensamblaje**)
- Se encarga de pasar los datos ensamblados a la aplicación correcta (y lo hace mediante los números de puertos) (**Multiplexación de conversaciones**)
- Identifica las diversas conversaciones entre los hosts
- Determina el protocolo que garantiza el envío del mensaje

Comparación con capa de red:

La **capa de transporte**, junto con la de red, constituye el núcleo de la jerarquía de protocolos.

- Mientras que la **capa de red** se encarga de la **entrega de paquetes punto a punto** entre máquinas
- la **capa de transporte** ofrece un **servicio de comunicación entre procesos** (de origen y destino) con un nivel de **confiabilidad configurable**, independientemente de la red física usada.

Su función principal es proporcionar a las **aplicaciones** un **servicio eficiente, confiable y económico** para la transmisión de datos. Para ello, utiliza los servicios de la capa de red, pero los mejora mediante mecanismos de control y corrección.

Entidad de transporte

El trabajo de la capa de transporte lo realiza una **entidad de transporte**, que puede ubicarse:

- En el **núcleo (kernel)** del sistema operativo,
 - En una **biblioteca** de red,
 - O en un **proceso de usuario**.
- Las dos primeras opciones son las más comunes.

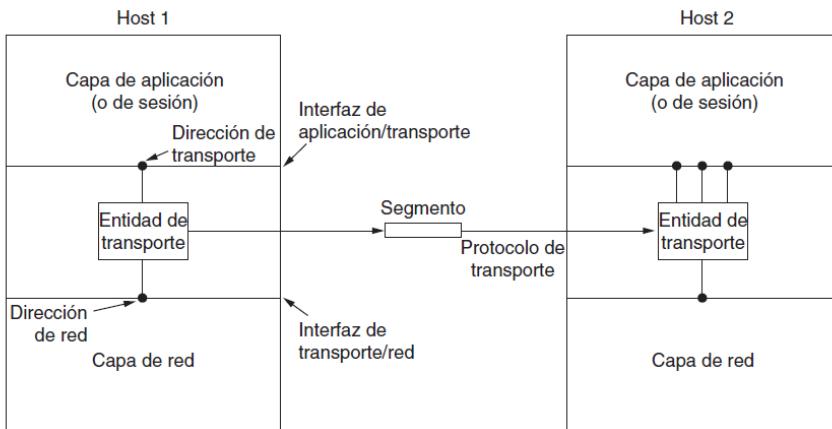


Figura 6-1. Las capas de red, transporte y aplicación.

Tipos de servicio de transporte

Existen dos tipos principales de servicio:

1. **Orientado a conexión:** tiene tres fases (establecimiento, transferencia y liberación) y ofrece control de flujo y fiabilidad.
2. **No orientado a conexión:** envía mensajes independientes sin establecer conexión previa.

¿Por qué existe una capa de transporte si la red ya ofrece servicios similares?

Aunque ambos servicios se parecen, la diferencia clave es **dónde se ejecutan los códigos**:

- La **capa de red** funciona principalmente en **enrutadores**, controlados por operadores de red.
- La **capa de transporte** se ejecuta en las **máquinas del usuario**, donde hay control directo.

Si la red es poco confiable (pierde paquetes, corta conexiones o falla), la capa de transporte puede **compensar los errores**. Ya que los usuarios no tienen control sobre los routers.

Si en una **red sin conexión** se pierden o dañan paquetes, la **entidad de transporte** puede **detectar el error y corregirlo** retransmitiendo los datos.

En una **red orientada a conexión**, si la conexión se **interrumpe abruptamente** durante una transmisión, la entidad de transporte puede **establecer una nueva conexión, consultar qué datos llegaron correctamente y reanudar el envío desde el punto en que se interrumpió**, garantizando así la continuidad y confiabilidad de la comunicación.

PRIMITIVAS CAPA DE TRANSPORTE

La capa de transporte **mejora la calidad del servicio** que ofrece la red subyacente y **oculta las diferencias** entre tecnologías de red.

Esto permite a los programadores escribir aplicaciones que funcionan en distintas redes **sin tener que modificar el código**, ya que usan las mismas **primitivas de transporte** (por ejemplo, TCP o UDP) que se pueden implementar como llamadas a procedimientos de biblioteca). Al ocultar el servicio de red detrás de un conjunto de primitivas de servicio de transporte, aseguramos que para cambiar la red simplemente hay que reemplazar un conjunto de procedimientos de biblioteca por otro que haga lo mismo con un servicio subyacente distinto.

Importancia dentro del modelo

Gracias a esta capa:

- Se logra **independencia entre las capas superiores y la red física**.
- Las **cuatro capas inferiores (física, enlace, red y transporte)** actúan como **proveedor del servicio**, mientras que las **capas superiores (aplicación, presentación y sesión)** son **usuarias**.
- La **capa de transporte** es el **límite principal entre el proveedor y el usuario del servicio confiable de transmisión de datos**.

Primitivas básicas del servicio de transporte orientado a conexión

El modelo simple (hipotético) usa **cinco primitivas principales**:

Primitiva	Paquete enviado	Significado
LISTEN	(ninguno)	Se bloquea hasta que algún proceso intenta conectarse.
CONNECT	CONNECTION REQ.	Intenta activamente establecer una conexión.
SEND	DATA	Envía información.
RECEIVE	(ninguno)	Se bloquea hasta que llegue un paquete DATA.
DISCONNECT	DISCONNECTION REQ.	Solicita que se libere la conexión

Funcionamiento básico (cliente-servidor)

1. El **servidor** ejecuta LISTEN, quedando bloqueado hasta que un cliente se conecte.
2. El **cliente** ejecuta CONNECT, enviando un segmento **CONNECTION REQUEST**.
3. El servidor responde con **CONNECTION ACCEPTED**, y se establece la conexión.
4. Ambos intercambian datos mediante SEND y RECEIVE.
5. Finalmente, cualquiera puede ejecutar DISCONNECT para cerrar la conexión.

Conceptos clave

- Los **mensajes entre entidades de transporte** se llaman **segmentos** (en TCP/UDP).
- Los **segmentos** viajan dentro de **paquetes** (nivel de red), y estos dentro de **tramas** (nivel de enlace).

- La **capa de transporte** maneja internamente los **temporizadores, retransmisiones y confirmaciones**, pero el usuario no ve esta complejidad: para él, la conexión parece un **conducto de bits confiable y ordenado**.

Liberación de conexión

Existen dos formas:

- **Asimétrica:** una parte ejecuta DISCONNECT, y ambas entidades liberan la conexión.
- **Simétrica:** cada parte cierra su lado por separado; la conexión termina solo cuando **ambas** emiten DISCONNECT.

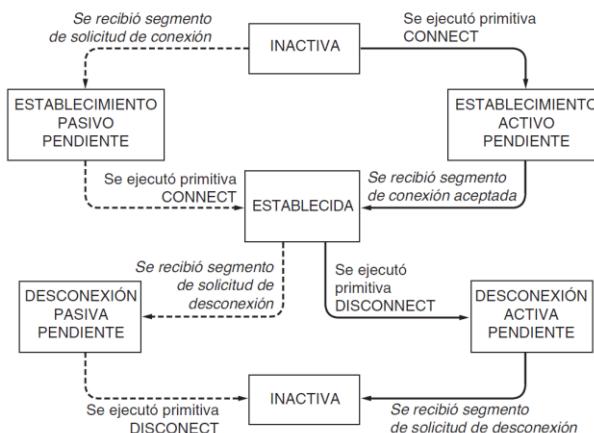


Figura 6-4. Un diagrama de estado para un esquema simple de manejo de conexiones. Las transiciones etiquetadas en cursiva se producen debido a la llegada de paquetes. Las líneas continuas muestran la secuencia de estados del cliente. Las líneas punteadas muestran la secuencia de estados del servidor.

Elementos de los protocolos de transporte.

El **servicio de transporte** se implementa mediante un **protocolo de transporte** que conecta las **entidades de transporte** de origen y destino. Este protocolo comparte ciertas funciones con los **protocolos de enlace de datos**, como el **control de errores**, la **secuenciación** y el **control de flujo**, pero existen **diferencias clave** debido a que operan en **entornos muy distintos**.

Similitudes con la capa de enlace de datos

Ambos protocolos se encargan de:

- Detectar y corregir errores.
- Mantener el orden de los datos.
- Regular el flujo de información para evitar saturar al receptor.

Principales diferencias con la capa de enlace de datos

1. Medio de comunicación

- En la **capa de enlace de datos**, la comunicación es **directa** entre dos dispositivos (por ejemplo, entre dos enrutadores conectados por un cable o enlace inalámbrico).
- En la **capa de transporte**, la comunicación se realiza **a través de toda la red**, lo que introduce **mayor complejidad** (múltiples nodos, rutas variables, posibles fallos intermedios).

2. Direcciónamiento

- En el **enlace de datos**, no hace falta especificar destino: cada enlace físico tiene un solo receptor.
- En la **capa de transporte**, es necesario **direccionar explícitamente** al host y al proceso destino (por ejemplo, mediante direcciones IP y puertos).

3. Establecimiento de conexión

- En **enlace de datos**, establecer conexión es **simple y directo**.
- En **transporte**, el **establecimiento inicial es más complicado**, ya que la red puede tener retardos, pérdida de paquetes o múltiples rutas.

4. Retrasos y duplicación

- En **enlace de datos**, un paquete solo puede **llegar o perderse**, pero no aparece fuera de orden.
- En **transporte**, los paquetes pueden **retrasarse, duplicarse o llegar desordenados**, especialmente si la red usa **datagramas** (como IP). Esto obliga a usar **protocolos especiales** para asegurar la entrega correcta.

5. Uso de búferes y control de flujo

- En **enlace de datos**, se asignan **búferes fijos** por línea, garantizando espacio para cada trama.
- En **transporte**, hay **muchas conexiones simultáneas y ancho de banda variable**, por lo que no se puede asignar un número fijo de búferes a cada conexión. Se necesitan mecanismos **dinámicos** de gestión de memoria y flujo.

Elementos: DIRECCIONAMIENTO

Veamos como ejemplo de una computadora que recibe y envía **correos electrónicos, mensajes instantáneos, páginas Web y llamadas telefónicas VoIP** de manera simultánea.

TCP y UDP mantienen un seguimiento de las aplicaciones que se comunican

Mediante identificadores únicos en los campos de encabezados que son:

NÚMEROS DE PUERTOS

En informática, los **números de puerto** son identificadores de 16 bits (de 0 a 65.535) que se combinan con una dirección IP para dirigir el tráfico de red a aplicaciones o servicios específicos en un dispositivo, de forma similar a cómo un número de apartamento dirige la correspondencia a un edificio. Los puertos bien conocidos, **como el 80 para HTTP**, están estandarizados para servicios comunes, mientras que otros rangos se usan para servicios registrados o de forma dinámica por aplicaciones cliente.

- **Identifican servicios:** Mientras que una dirección IP identifica un dispositivo en una red, un número de puerto identifica un servicio o aplicación específico dentro de ese dispositivo, como un navegador web, un servidor de correo o un cliente de chat.
- **Permiten la comunicación:** Al llegar un paquete de datos, el sistema operativo utiliza la combinación de la dirección IP y el número de puerto para saber a qué aplicación en concreto debe entregarse ese dato.
- **Aseguran la organización:** Ayudan a organizar la comunicación en una red, permitiendo que múltiples servicios (navegación web, correo electrónico, transferencias de archivos) funcionen simultáneamente en el mismo dispositivo sin mezclarse entre sí.

Cuando un proceso desea establecer una conexión con un proceso de aplicación remoto, debe especificar a cuál se conectará. (¿A quién mando el mensaje?) El método que normalmente se emplea es definir direcciones de transporte en las que los procesos pueden estar a la escucha de solicitudes de conexión.

En [Internet](#), estos puntos terminales se denominan puertos, pero usaremos el término genérico de [TSAP](#) ([Punto de Acceso al Servicio de Transporte](#)).

Los puntos terminales análogos de la capa de red se llaman [NSAP](#) ([Punto de Acceso al Servicio de Red](#)). Las direcciones [IP](#) son ejemplos de NSAPs.

Tomar la decisión correcta entre los protocolos TCP y UDP al diseñar o implementar un servicio o aplicación de red es esencial para garantizar un rendimiento óptimo.

Cada uno de ellos admite el flujo de datos en redes informáticas e Internet. Sin embargo, aunque los dos protocolos están en la [capa de transporte del modelo OSI](#), tienen algunas diferencias en la forma en que se conectan y transmiten datos.

La principal diferencia entre el TCP (protocolo de control de transmisiones) y el UDP (protocolo de datagramas de usuario) es que el TCP es un protocolo basado en conexiones y el UDP es sin conexiones. Aunque el TCP es más fiable, transfiere los datos más despacio. El UDP es menos fiable pero funciona más rápido. Esto hace que cada protocolo sea adecuado para distintos tipos de transferencia de datos.

En esta capa se manejan usualmente dos protocolos, los cuales son el **TCP (Protocolo de Control de Transmisión)** y **UDP (Protocolo de Datagrama de Usuario)**, sus características principales son las siguientes:

TCP	UDP
<ul style="list-style-type: none"> Servicio con conexión, se establece una sesión entre ambos host. Garantiza la fidelidad de los datos, creando una secuencia de datos para su entrega. Es mucho mas lento, ya que necesita comprobar que el paquete se haya recibido completo, ya que si no, se deberá reenviar el tramo faltante para completar el servicio. Permite una comunicación de uno a uno. 	<ul style="list-style-type: none"> Servicio sin conexión, no se necesita establecer una sesión entre host. No garantiza la fidelidad de los datos y no crea ninguna secuencia al entregar. Es mas rápido, solo se encarga de entregar los paquetes en el menor tiempo posible, por lo que no le interesa si llegan completos o no. Permite comunicaciones de uno a uno y de uno a muchos.

DIFERENCIAS ENTRE TCP Y UDP

Factor	TCP	UDP
Tipo de conexión	Requiere una conexión establecida antes de transmitir datos	No se necesita conexión para iniciar y finalizar una transferencia de datos
Secuencia de datos	Puede secuenciar datos (enviar en un orden específico)	No puede secuenciar u ordenar datos
Retransmisión de datos	Puede retransmitir datos si no llegan los paquetes	Sin retransmisión de datos. Los datos perdidos no se pueden recuperar
Entrega	La entrega está garantizada	La entrega no está garantizada
Comprobar si hay errores	Una exhaustiva comprobación de errores garantiza que los datos lleguen en buen estado	La comprobación de errores cubre los aspectos básicos, pero puede que no evite todos los errores
Emisiones	No es compatible	Sí es compatible
Velocidad	Lenta, pero entrega los datos completos	Rápida, pero existe el riesgo de que los datos se entreguen incompletos

Qué protocolo es mejor: ¿TCP o UDP?

Depende de lo que haga en línea y el tipo de datos que se transfieren. El UDP es mejor para jugar en línea, porque su rápida transferencia de datos permite jugar casi sin retrasos. El TCP es mejor si transfiere archivos, como fotos familiares, porque garantiza que los datos llegarán exactamente como se enviaron.

El TCP es mejor para:

1. Mandar correos electrónicos o mensajes de texto
2. Transferir archivos
3. Navegar en la web

El UDP es mejor para:

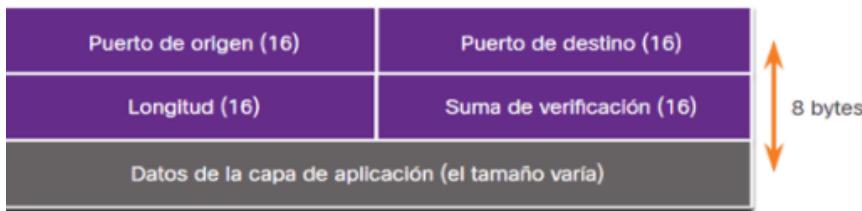
1. Hacer streaming en directo

2. Jugar en línea
3. Hacer videochats

ENCABEZADO TCP

Puerto de origen		Puerto de destino													
Número de secuencia															
número de acuse de recibo (ACK) o número de acuse de recibo negativo (NACK) dependiendo de la bandera activada															
Longitud de cabecera	Reservado	NACK	URG	ACK	PSH	RST	SYN	FIN							
Suma de verificación (CRC)		Puntero urgente													
Opciones															
Datos															

ENCABEZADO UDP



En el encabezado de cada segmento o datagrama hay

- **Un puerto origen:** El número de puerto de origen es el número para la comunicación, asociado con la aplicación que origina la comunicación en el host local
- **Un puerto destino:** El número de puerto de destino es el número para esta comunicación asociado con la aplicación de destino en el host remoto.

¿Cómo se asignan los números de puertos?

- De varias maneras
- Si el mensaje es una solicitud o una respuesta
- Los procesos en el servidor poseen números de puertos estáticos asignados a ellos
- Los clientes eligen un número de puerto en forma dinámica para cada conversación

Aplicación de cliente → Aplicación de servidor

Envía solicitud

QUITEZ

- El **puerto de destino** contenido en el encabezado es el número de puerto que se asigna al **daemon (programa residente)** de servicio que se ejecuta en el host remoto
- El **software del cliente** debe conocer el número de puerto asociado con el proceso del servidor en el host remoto

- Este número de puerto de destino se puede configurar, ya sea de forma **predeterminada o manual**
- **Ejemplo:** cuando una aplicación de explorador **Web** realiza una solicitud a un servidor **Web**, el explorador utiliza **TCP** y el número de puerto **80** a menos que se especifique otro valor, porque el puerto **TCP 80** es el puerto predeterminado asignado a aplicaciones de servidores **Web**
- El **puerto de origen** del encabezado de un segmento o datagrama de un cliente se genera de manera **aleatoria**
- Si no hay conflicto con otros puertos en uso en el sistema, el cliente puede elegir cualquier número de puerto
- El número de puerto actúa como **dirección de retorno** para la aplicación que realiza la solicitud
- La **capa de Transporte** mantiene un seguimiento de este puerto y de la aplicación que generó la solicitud, de manera que cuando se devuelva una respuesta, pueda ser enviada a la aplicación correcta
- El **número de puerto** de la aplicación que realiza la solicitud se utiliza como **número de puerto de destino** en la respuesta que vuelve del servidor

IANA

Autoridad de Números Asignados de Internet : asigna números de puerto. IANA es un **organismo de estándares** responsable de la asignación de varias normas de direccionamiento.

TIPOS DE NÚMEROS DE PUERTOS

- **Puertos bien conocidos** (Números del 0 al 1023): estos números se reservan para servicios y aplicaciones. Por lo general, se utilizan para aplicaciones como HTTP (servidor Web), POP3/SMTP (servidor de e-mail) y Telnet. Al definir estos puertos conocidos para las aplicaciones del servidor, las aplicaciones del cliente pueden ser programadas para solicitar una conexión a un puerto específico y su servicio asociado.
- **Puertos registrados** (Números 1024 al 49151): estos números de puertos están asignados a procesos o aplicaciones del usuario. Estos procesos son principalmente aplicaciones individuales que el usuario elige instalar en lugar de aplicaciones comunes que recibiría un puerto bien conocido. Cuando no se utilizan para un recurso del servidor, estos puertos también pueden utilizarse si un usuario los selecciona de manera dinámica como puerto de origen.
- **Puertos dinámicos o privados** (Números del 49 152 al 65 535): también conocidos como puertos efímeros, suelen asignarse de manera dinámica a aplicaciones de cliente cuando se inicia una conexión. No es muy común que un cliente se conecte a un servicio utilizando un puerto dinámico o privado (aunque algunos programas que comparten archivos punto a punto lo hacen).

Rango de números de puerto	Grupo de puertos
De 0 a 1023	Puertos bien conocidos (Contacto)
De 1024 a 49151	Puertos registrados
De 49152 a 65535	Puertos privados y/o dinámicos

Puertos TCP registrados:
1863 MSN Messenger
8008 HTTP alternativo
8080 HTTP alternativo

Puertos TCP bien conocidos:
21 FTP
23 Telnet
25 SMTP
80 HTTP
110 POP3
194 Internet Relay Chat (IRC)
443 HTTP seguro (HTTPS)

Restablecer

Puertos TCP

Puertos UDP

Puertos TCP/UDP comunes

Rango de números de puerto	Grupo de puertos
De 0 a 1023	Puertos bien conocidos (Contacto)
De 1024 a 49151	Puertos registrados
De 49152 a 65535	Puertos privados y/o dinámicos

Puertos UDP registrados:
1812 Protocolo de autenticación RADIUS
2000 Cisco SCCP (VoIP)
5004 RTP (Voice and Video Transport Protocol)
5060 SIP (VoIP)

Puertos UDP bien conocidos:
69 TFTP
520 RIP

Restablecer

Puertos TCP

Puertos UDP

Puertos TCP/UDP comunes

Rango de números de puerto	Grupo de puertos
De 0 a 1023	Puertos bien conocidos (Contacto)
De 1024 a 49151	Puertos registrados
De 49152 a 65535	Puertos privados y/o dinámicos

Puertos TCP/UDP registrados comunes:
1433 MS SQL
2948 WAP (MMS)

Puertos comunes TCP/UDP bien conocidos:
53 DNS
161 SNMP
531 Mensajería instantánea de AOL, IRC

Restablecer

Puertos TCP

Puertos UDP

Puertos TCP/UDP comunes

Elementos: ESTABLECIMIENTO DE UNA CONEXIÓN

Cuando dos hosts se comunican utilizando **TCP**:

- Se establece una conexión antes de que puedan intercambiarse los datos
- El host rastrea cada segmento de datos dentro de una sesión e intercambia información sobre los datos recibidos
- Luego de que se completa la comunicación, se cierran las sesiones y la conexión finaliza

Para establecer la conexión los hosts realizan un intercambio de señales de **tres vías**. Los bits de control en el encabezado TCP indican el progreso y estado de la conexión.

Enlace de tres vías:

- Establece que el dispositivo de destino esté presente en la red
- Verifica que el **dispositivo de destino** tenga un servicio activo y esté aceptando las peticiones en el número de **puerto de destino** que el cliente que lo inicia intente usar para la sesión
- Informa al **dispositivo de destino** que el cliente de origen intenta establecer una sesión de comunicación en ese **número de puerto**

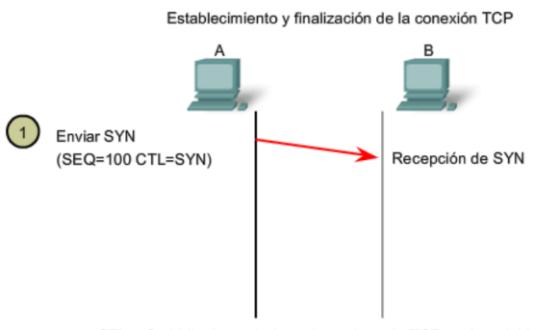
Los **tres pasos** para el establecimiento de una conexión **TCP** son:

1. El **cliente** que inicia la conexión envía un segmento que contiene un valor de secuencia inicial, que actúa como **solicitud** para **el servidor** para comenzar una sesión de comunicación
2. El **servidor** responde con un segmento que contiene un valor de reconocimiento igual al valor de secuencia recibido **más 1**, además de su propio valor de secuencia de sincronización. El valor es uno mayor que el número de secuencia porque el **ACK** es siempre el próximo Byte u Octeto esperado. Este valor de reconocimiento permite al **cliente** unir la respuesta al segmento original que fue enviado al servidor
3. El **cliente** que inicia la conexión responde con un valor de reconocimiento igual al valor de secuencia que recibió más uno. Esto completa el proceso de **establecimiento de la conexión**

Dentro del **encabezado** del segmento **TCP**, existen seis campos de 1 bit que contienen información de control utilizada para gestionar los procesos de **TCP**

- **URG:** Urgente campo de señalizador significativo
- **ACK: Campo significativo de acuse de recibo (parcial)**
- **PSH:** Función de empuje
- **RST:** Reconfiguración de la conexión
- **SYN:** Sincronizar números de secuencia
- **FIN: No hay más datos desde el emisor**

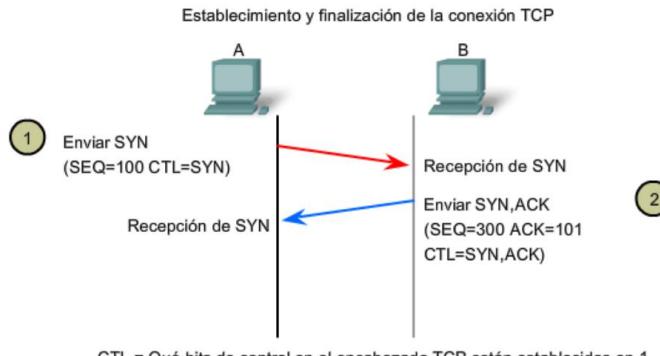
A estos campos se los denomina **señaladores** porque el valor de uno de estos campos es sólo de 1 bit, entonces tiene sólo dos valores: 1 ó 0. Si el valor del bit se establece en 1, indica la información de control que contiene el segmento



CTL = Qué bits de control en el encabezado TCP están establecidos en 1

A envía la solicitud de SYN a B.

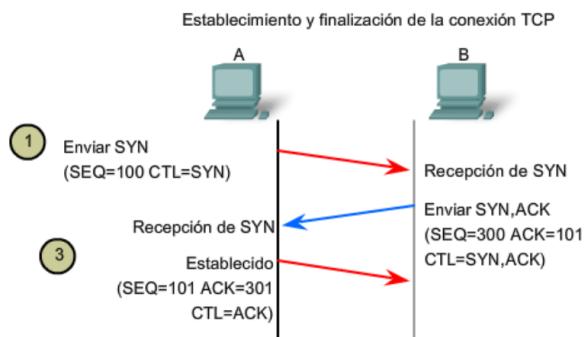
Restablecer	SYN ACK	1	2	3
-------------	---------	---	---	---



CTL = Qué bits de control en el encabezado TCP están establecidos en 1

B envía la respuesta de ACK y la solicitud de SYN a A.

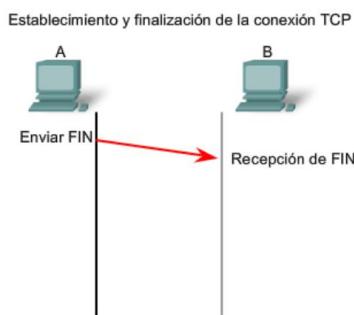
Restablecer	SYN ACK	1	2	3
-------------	---------	---	---	---



CTL = Qué bits de control en el encabezado TCP están establecidos en 1

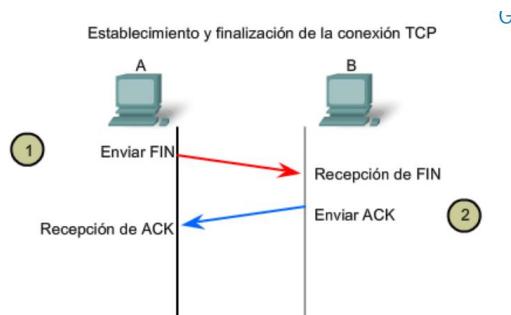
A envía la respuesta de ACK a B.

Restablecer	SYN ACK	1	2	3
-------------	---------	---	---	---



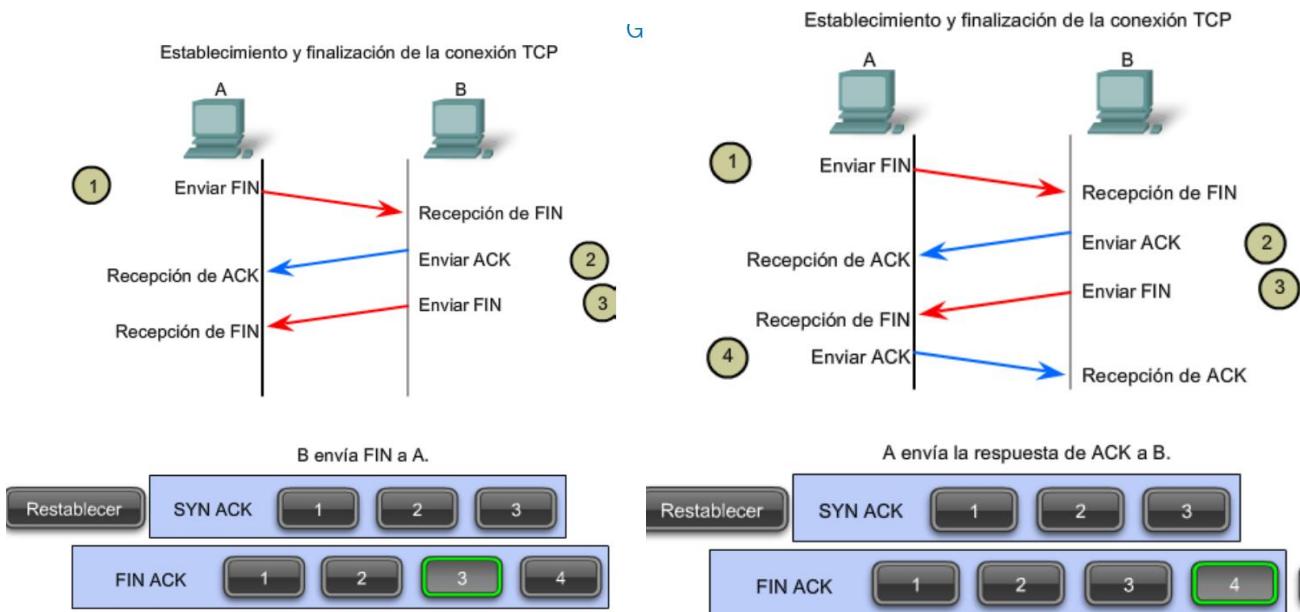
A envía la solicitud de FIN a B.

Restablecer	SYN ACK	1	2	3
FIN ACK	1	2	3	4



B envía la respuesta de ACK a A.

Restablecer	SYN ACK	1	2	3
FIN ACK	1	2	3	4



Elementos: FINALIZACIÓN DE UNA CONEXIÓN

- **Estilos:**

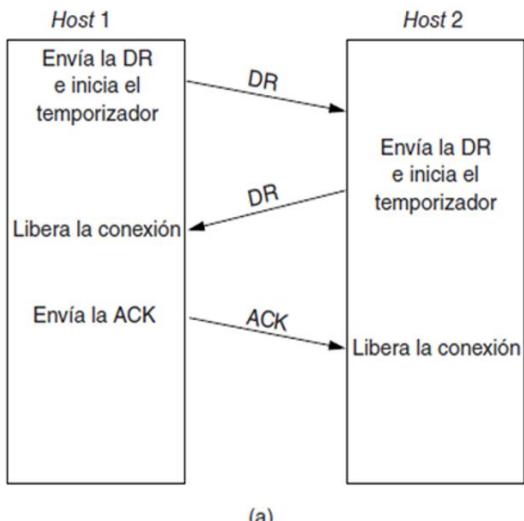
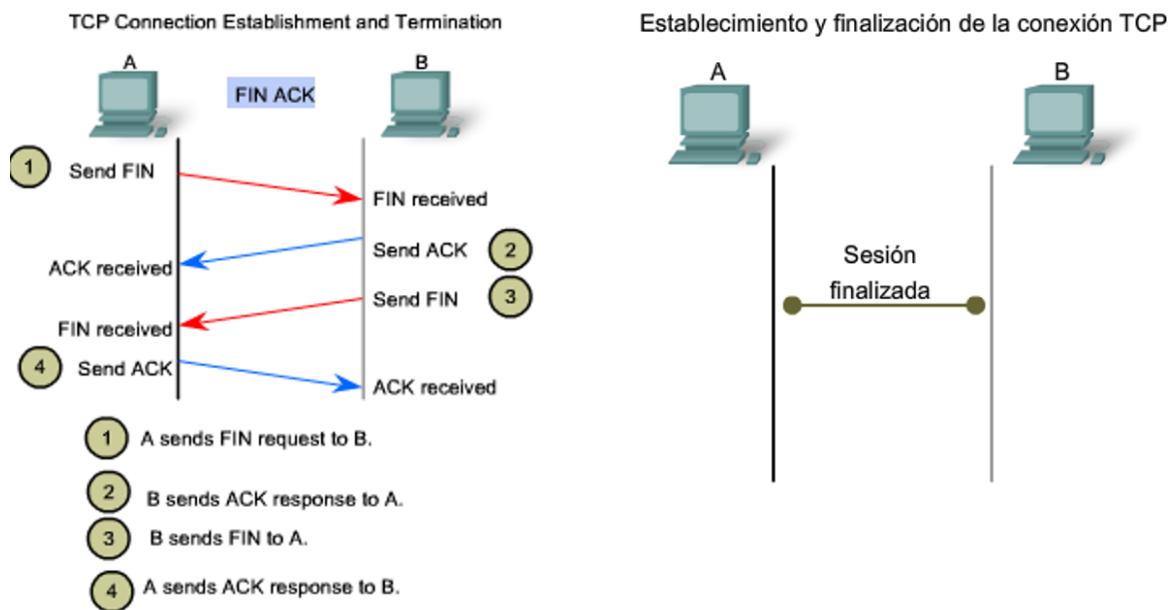
- **Liberación Asimétrica:** Como una llamada telefónica; un lado cuelga y termina la conexión. Es abrupta y puede provocar pérdida de datos.
- **Liberación Simétrica:** Cada dirección se cierra de forma independiente.

Para finalizar todas las sesiones TCP de una vía, se utiliza un enlace de dos vías, que consta de un segmento **FIN** y un **segmento ACK**

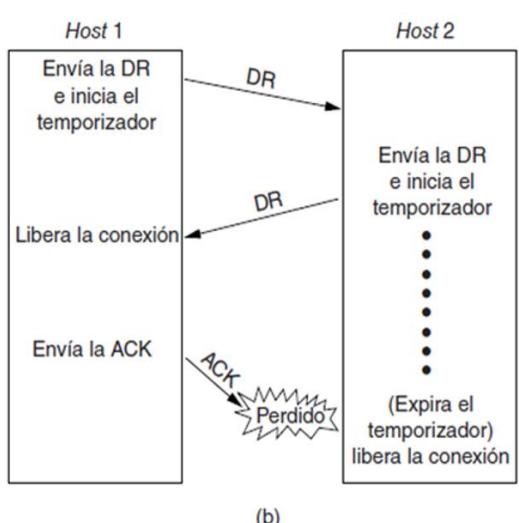
Por lo tanto, para terminar una conversación simple admitida por TCP, **se requieren cuatro intercambios para finalizar ambas sesiones :**

1. Cuando **el cliente** no tiene más datos para enviar al servidor, envía un segmento con el señalizador **FIN** establecido
2. El **servidor** envía un **ACK** para acusar recibo de **FIN** y terminar la sesión del **cliente al servidor**
3. El **servidor** envía un **FIN** al **cliente** para finalizar la sesión del **servidor al cliente**

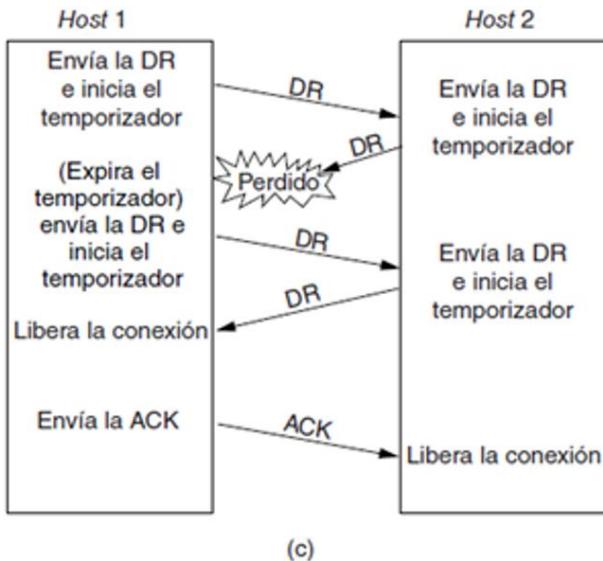
4. El cliente responde con un ACK para dar acuse de recibo de FIN desde el servidor



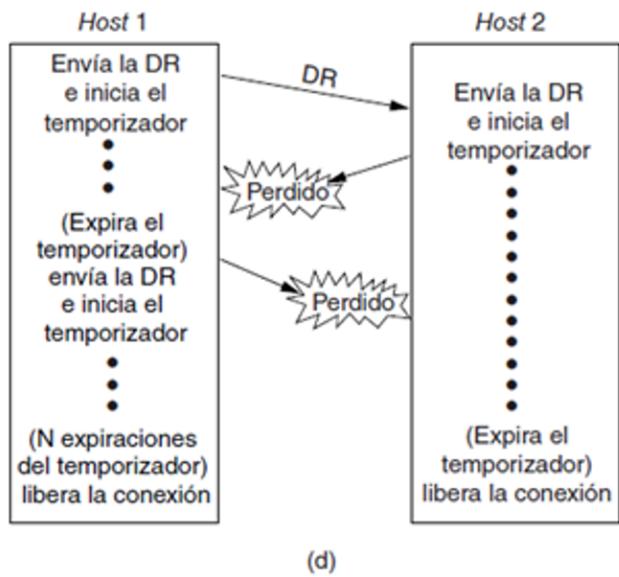
1. Caso normal



2. Se pierde la ultima TPDU ACK



3. Se pierde la 2º TPDU DR



4. Se pierden todos los intentos TPDU DR

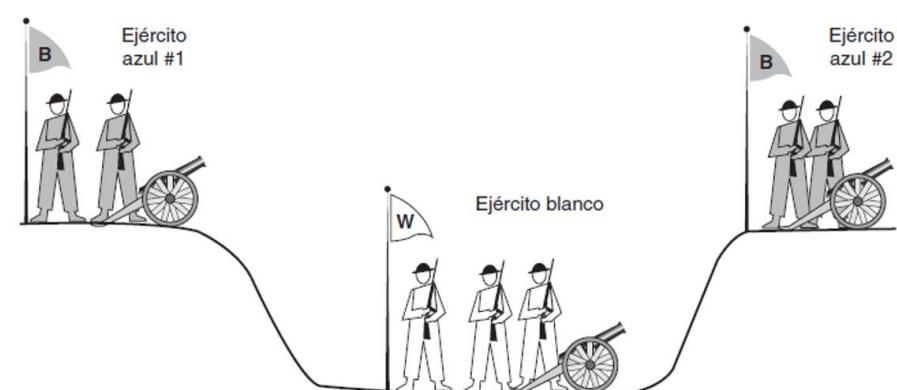


Figura 6-13. El problema de los dos ejércitos.

Elementos: CONTROL DE FLUJO

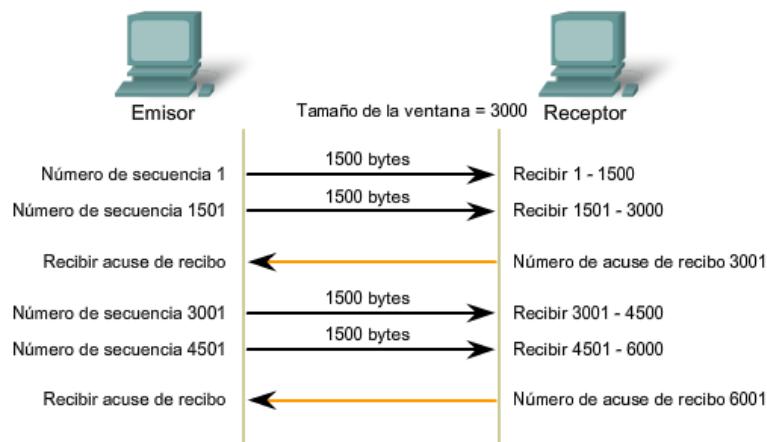
El control del flujo contribuye con la confiabilidad de la transmisión **TCP** ajustando la tasa efectiva de flujo de datos entre los dos servicios de la sesión

Cuando el **origen** advierte que se recibió la cantidad de datos especificados en los segmentos, puede continuar enviando más datos para esta sesión

El campo **Tamaño** de la ventana en el encabezado **TCP** especifica la cantidad de datos que puede transmitirse antes de que se reciba el acuse de recibo

El tamaño de la ventana inicial se determina durante el comienzo de la sesión a través del enlace de **tres vías**

Acuse de recibo de segmentos TCP y tamaño de la ventana



El **tamaño de la ventana** determina la cantidad de bytes enviados antes de esperar un acuse de recibo.

El número de **acuse de recibo** es el número del próximo byte esperado.

REDUCCIÓN DEL TAMAÑO DE LA VENTANA

Otra forma de controlar el flujo de datos es utilizar tamaños dinámicos de ventana.

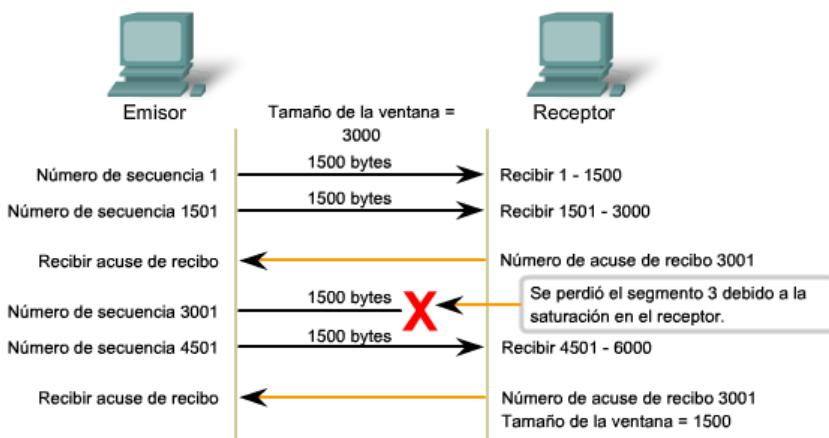
Cuando los recursos de la red son limitados, **TCP** puede **reducir** el tamaño de la ventana para lograr que los segmentos recibidos sean reconocidos con mayor frecuencia.

Esto disminuye de manera efectiva la tasa de transmisión, ya que el origen espera que los datos sean recibidos con más frecuencia.

El host **receptor TCP** envía el valor del tamaño de la ventana al **TCP emisor** para indicar el número de bytes que está preparado para recibir como parte de la sesión

Si el **destino** necesita disminuir la tasa de comunicación debido a limitaciones de **memoria del búfer**, puede enviar un valor de tamaño de la ventana menor al **origen** como parte de un acuse de recibo

Saturación de TCP y control del flujo



Si se pierden segmentos debido a la saturación, el receptor acusará recibo del último segmento secuencial recibido y responderá con un tamaño de ventana reducido.

Otros elementos que no puso: Control de Errores y Almacenamiento en Búfer

- **Mecanismos:** Son similares a los de la capa de enlace de datos, pero operan de extremo a extremo:
 - **Detección de Errores:** Sumas de comprobación (checksum) para detectar datos corruptos.
 - **Fiabilidad:** Números de secuencia y retransmisiones (ARQ).
 - **Control de Flujo:** Un protocolo de ventana deslizante para evitar que un emisor rápido saturé a un receptor lento.
 - **Diferencias con la Capa de Enlace de Datos:**
 - **Función:** Las sumas de comprobación de la capa de transporte son de **extremo a extremo**, protegiendo los datos incluso de corrupciones dentro de los enrutadores. Esto es crucial para la corrección (el *argumento de extremo a extremo*).
 - **Grado:** Las conexiones de transporte suelen tener un **producto ancho de banda-retardo** grande, lo que requiere ventanas deslizantes grandes para un buen rendimiento. Los protocolos "parada y espera" son ineficientes aquí.
 - **Almacenamiento en Búfer:**
 - **¿Dónde?** Se necesitan búferes tanto en el emisor (para retransmisiones) como en el receptor.
 - **Estrategia:** Depende del tráfico.
 - **Tráfico de bajo ancho de banda o en ráfagas (ej. telnet):** Adquirir búferes dinámicamente.
 - **Tráfico de alto ancho de banda (ej. transferencia de archivos):** El receptor debe dedicar una ventana completa de búferes.

- **Gestión de Búferes:** Se pueden usar búferes de tamaño fijo, de tamaño variable o un gran búfer circular por conexión.
- **Asignación Dinámica de Búferes y Control de Flujo:**
 - El receptor indica al emisor cuánto espacio tiene en búfer usando un campo de **tamaño de ventana** (como en TCP).
 - Esto proporciona control de flujo. El emisor ajusta su ventana basándose en el espacio que anuncia el receptor.
 - El límite último de la velocidad suele ser la **capacidad de transporte de la red**, no los búferes del receptor. Si el emisor envía demasiado rápido, causará congestión en la red. Los ajustes dinámicos de la ventana también se pueden usar para el **control de congestión**.

Multiplexión

- **Multiplexión:** Múltiples conexiones de transporte que utilizan la misma dirección de red (ej. Múltiples conexiones TCP comparten una dirección IP). La capa de transporte demultiplexa los segmentos entrantes hacia el proceso correcto basándose en el número de puerto.
- **Multiplexión Inversa:** Dividir una conexión de transporte en múltiples rutas de red para aumentar el ancho de banda o la fiabilidad. Un ejemplo es el protocolo **SCTP**.

Recuperación de Fallas

- **Problema:** Cómo recuperarse de un fallo y reinicio de un host sin perder o duplicar datos.
- **El Dilema Fundamental:** El servidor debe elegir entre *confirmar primero* o *escribir en la aplicación primero*. El cliente debe elegir si *retransmitir* o no después de un fallo.
 - **Confirmar Primero, Luego Escribir:** Si el servidor falla después del ACK pero antes de escribir, el cliente cree que los datos se recibieron, pero en realidad se perdieron.
 - **Escribir Primero, Luego Confirmar:** Si el servidor falla después de escribir pero antes del ACK, el cliente retransmitirá, causando un duplicado.
- **Conclusión Fundamental:** Es **imposible** que la capa de transporte se recupere de forma transparente de un fallo de un host. La recuperación debe ser manejada por una capa superior (la aplicación) que retenga suficiente estado.
- **El Argumento de "Extremo a Extremo" Revisitado:** Una verdadera confirmación de extremo a extremo (que signifique que el trabajo está *definitivamente hecho*) es imposible solo en la capa de transporte. La aplicación debe estar involucrada para lograr este nivel de garantía.

DESEMPEÑO

Cuando hay muchos equipos conectados entre si, es muy común que existan **interacciones complejas** entre equipos y que **provocan consecuencias** que no se pueden prever y que **empobrece** el desempeño de esa red.

Importancia

- El desempeño depende del **trabajo conjunto** de las capas de **enlace, red y transporte**.
- Entenderlo no es una ciencia exacta, sino **un arte basado en la experiencia y observación práctica**.

ASPECTOS DEL DESEMPEÑO DE LAS REDES (final)

1. Problemas de desempeño

1. Surgen por **congestión** (exceso de tráfico), **desequilibrios de recursos** (por ejemplo, una PC lenta conectada a una red muy rápida) o **errores sincronizados** que generan sobrecargas.
 1. **Congestión:** Ocurre cuando existe **sobrecarga de información en la red**, es decir, cuando el volumen de datos transmitidos supera la capacidad que la red puede manejar. Esto produce **retrasos, pérdidas de paquetes** y necesidad de retransmisión, afectando el desempeño global.
 2. **Desequilibrio estructural de los recursos:** Sucede cuando un recurso se utiliza en exceso hasta el punto de degradar el rendimiento general de la red.

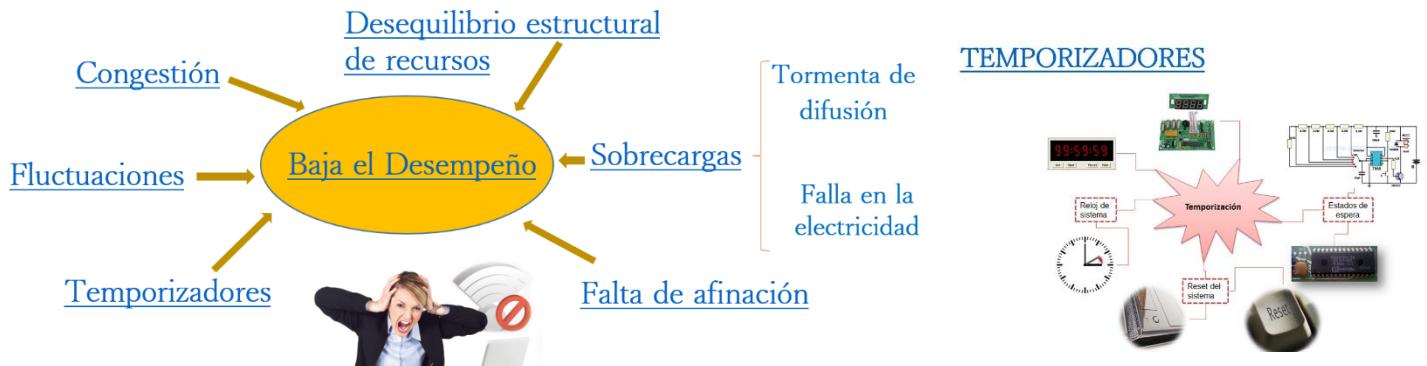
Ejemplos:

1. Saturación del ancho de banda.
 2. Routers desconfigurados o con mala distribución de tráfico.
 3. Rotura de cables (por ejemplo, en cables de par trenzado) que afectan la estabilidad de la transmisión.
3. **Sobrecarga:** Se manifiesta principalmente en dos situaciones:
 1. **Error en la retransmisión:** Si muchas máquinas tienen problemas al recibir un paquete, este puede retransmitirse miles de veces, generando una sobrecarga masiva en la red.
 2. **Corte de suministro eléctrico:** Cuando se restablece la energía, todos los sistemas y dispositivos se reinician al mismo tiempo, provocando una sobrecarga temporal de hardware y tráfico en la red.
 4. **Falta de afinación del sistema:** Se refiere a una configuración inadecuada de los parámetros del sistema o la red, lo que puede generar ineficiencia o bloqueos parciales en la transmisión de datos.
 5. **Temporizadores:** Los tiempos de espera (timeouts) deben definirse con precisión.
 1. Si el tiempo es demasiado corto, los paquetes pueden no alcanzar a enviarse o recibirse, generando congestión por retransmisiones innecesarias.
 2. Si el tiempo es demasiado largo, se desperdicia tiempo de procesamiento y se reduce el rendimiento general.

6. **Fluctuaciones:** Son variaciones o inestabilidades en la transmisión de datos dentro de la red.

Por ejemplo, si fluctúa el número de puerto o la secuencia de un paquete, el resto de la comunicación se ve afectado, ya que se deben verificar y reenviar todos los paquetes involucrados.

2. Estos problemas provocan **retrasos, pérdida de paquetes, retransmisiones** y, en consecuencia, una **reducción del rendimiento general**.



2. Medición del desempeño de una red

1. Implica evaluar métricas como **ancho de banda, retardo, pérdida de paquetes, eficiencia de transmisión y tiempo de respuesta**.
2. Sirve para detectar cuellos de botella y optimizar el flujo de datos.

Ante un **bajo desempeño** las quejas son con los **administradores**, quienes deben determinar que ocurre, **efectuando mediciones**. La mejora del desempeño consta de un ciclo de tres pasos:

1. **Medir** parámetros y el desempeño de la red
2. **Tratar** de entender que ocurre
3. **Cambiar** un parámetro

Estos pasos se repiten hasta tener un **buen desempeño** o hasta que se hayan **agotado** todas las **posibles mejoras**

1. Asegurarse que el **tamaño de la muestra** sea lo bastante grande
2. Asegurarse de que las **muestras son representativas**
3. Cuidado de **usar relojes de intervalos grandes**
4. El cache puede **arruinar las mediciones**: Afectan porque el cache posee datos previos entonces cuando queramos medir exactamente en el momento actual y el cache tiene datos previos va a afectar la medición por lo cual debemos eliminar el cache antes de medir.
5. Entender **qué se está midiendo**



*** TRADUCIR - convertir de una forma a otra...

*** INTERPRETAR - explicar o resumir la información

*** EXTRAPOLAR - extender el significado
más allá de los datos...

6. Tener cuidado con la extrapolación de los resultados

EXTRAPOLAR:

- Se define como la habilidad de aplicar un concepto, contenido o acuerdo a una situación nueva.
- Genera relaciones íntimas con el proceso de conceptualizar, analogizar, inferir y todas aquellas habilidades que se encuentran a la base de estas.

3. Diseño de sistemas con mejor desempeño

1. Los equipos deben tener suficiente **capacidad de procesamiento y memoria** para manejar grandes volúmenes de tráfico.
2. Una máquina lenta puede convertirse en el **punto débil** de toda la red.

La **medición y los ajustes** con frecuencia mejoran el **desempeño**, pero no pueden reemplazar un **diseño original**. Una red mal diseñada se puede mejorar hasta **un límite**.

Algunas reglas empíricas, relacionadas con el diseño del sistema, no solo con el diseño de la red, se detallan a continuación:

Regla #1: La velocidad del CPU es más importante que la velocidad de la red.

Regla #2: Reducir el número de paquetes para reducir la sobrecarga de software.

Regla #3: Reducir al mínimo las commutaciones de contexto.

Regla #4: Reducir al mínimo las copias.

Regla #5: Es posible comprar más ancho de banda, pero no un retardo menor.

Regla #6: Evitar la congestión es mejor que recuperarse de ella.

Regla #7: Evitar expiraciones del temporizador.

4. Procesamiento rápido de segmentos TPDU (Unidad de Datos del Protocolo de Transporte)
 1. Se busca optimizar la **gestión interna de los paquetes TCP/UDP**, reduciendo el tiempo que tarda el sistema operativo en procesarlos.
 2. Esto mejora la eficiencia de transmisión y reduce la latencia.
5. **Compresión de encabezados:**
 1. En conexiones lentas o repetitivas, comprimir los encabezados TCP/IP puede **reducir el tamaño de los datos enviados y aumentar la velocidad efectiva**.
6. Protocolos para redes futuras de alto desempeño
 1. Se diseñan **versiones mejoradas de TCP** y otros protocolos para adaptarse a **redes modernas de alta velocidad**, minimizando la congestión y maximizando el rendimiento.

Control de congestión: Cubeta por goteo y cubeta por tokens

El **control de congestión** busca evitar la sobrecarga en la red, regulando la cantidad de datos que pueden transmitirse en un momento determinado.

Para ello, existen mecanismos como **la cubeta por goteo (leaky bucket)** y **la cubeta por tokens (token bucket)**.

Cubeta por goteo (Leaky Bucket)

Concepto:

Funciona como una cubeta que tiene un pequeño orificio por donde el agua (datos) gotea a una velocidad constante.

Explicación:

- Los paquetes llegan a la cubeta a una cierta velocidad.
- Si la velocidad de llegada **superá la velocidad de drenaje**, la cubeta se llena y **rebalsa**.
- En el contexto de redes, esto significa que **los paquetes excedentes se pierden** (se descartan).

Problema:

Cuando hay un exceso de tráfico, el sistema **pierde información**, ya que los paquetes que no pueden almacenarse se eliminan.

Cubeta por tokens (Token Bucket)

Concepto:

En lugar de controlar el flujo mediante un drenaje constante, este método utiliza **tokens (fichas)** generados por un **temporizador**.

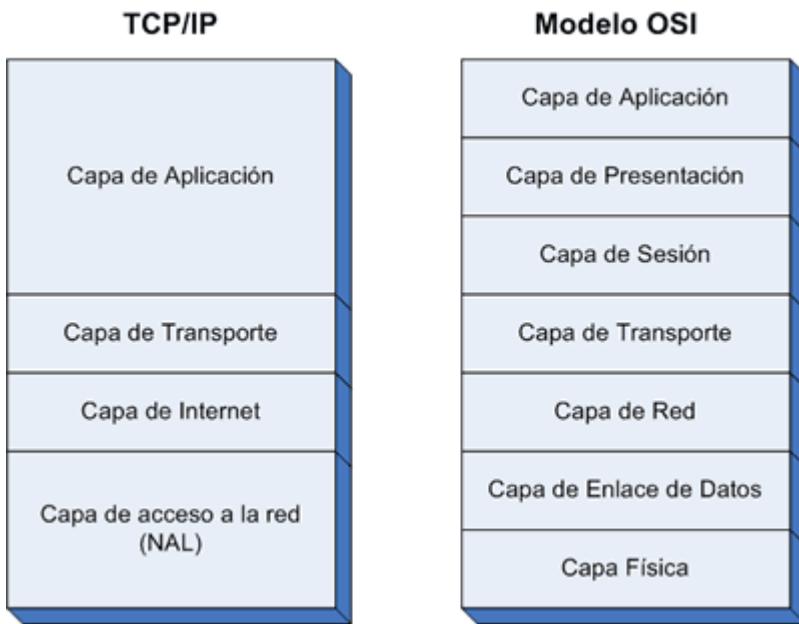
Explicación:

- Cada token representa **el permiso para enviar un paquete**.
- Los tokens se van acumulando en la cubeta hasta un límite máximo.
- Si la cubeta está llena y se generan nuevos tokens, **estos se descartan**, pero **no se pierden paquetes**, ya que solo se envían cuando hay tokens disponibles.

Resultado:

- Si hay tokens, se pueden transmitir paquetes inmediatamente.
- Si no hay tokens, el sistema debe esperar a que se generen nuevos.
- De esta forma, **no se pierden datos**, ya que el control se realiza sobre los **tokens** y no sobre los paquetes.

Capa de Sesión y Capa de Presentación



(Temas que suelen aparecer como contenido del ejercicio 3 en los finales, pero no son tema principal de examen.)

Capa de Sesión

La **capa de sesión** permite que usuarios en diferentes máquinas establezcan **sesiones de comunicación** entre sí. A través de estas sesiones se puede realizar el transporte de datos —como lo hace la capa de transporte—, pero **mejorando los servicios** y el control sobre la comunicación entre aplicaciones.

- La **capa de transporte** apunta principalmente a los **dispositivos**.
- La **capa de sesión** apunta a las **aplicaciones** que se comunican entre sí.

Definición general

Recibe los paquetes de las cuatro capas inferiores del modelo OSI y los transforma en **sesiones**. Para lograrlo, utiliza **mecanismos de control** que gestionan el flujo y la sincronización de la comunicación.

Mecanismos de control

- **Control a nivel de conversación:** determina **quién habla y cuándo**, evitando colisiones o interferencias en la comunicación.
- **Coordinación de peticiones y respuestas:** regula las solicitudes de servicio y mantiene el control de las aplicaciones involucradas en una sesión.
- **Control y separación del diálogo:** establece, administra y finaliza las sesiones activas entre aplicaciones.

En síntesis, la capa de sesión **coordina quién se comunica, en qué momento y cómo se desarrolla la conversación** entre el host de origen y el de destino.

Servicios que brinda

- Establecer, mantener y finalizar sesiones entre aplicaciones.
- Controlar la sesión entre emisor y receptor.
- **Control de concurrencia:** administra el uso simultáneo de aplicaciones y controla la demanda (por ejemplo, ante ataques DDoS).
- **Puntos de verificación (checkpoints):** permiten reanudar una sesión desde un punto específico en caso de interrupción, evitando reiniciar desde el principio.

En conclusión, **mantiene el enlace lógico entre dos computadoras.**

Nota: las “colisiones” en esta capa **no** son físicas (como las de paquetes en la capa de enlace), sino que ocurren cuando **la demanda de una aplicación excede su capacidad de manejo.**

Problemas frecuentes en la sesión

1. Cruce de mensajes durante la conversación

○ **Solución:**

- *Half-duplex:* comunicación alternada de dos vías; transmite y recibe en ambas direcciones, pero solo una a la vez.
- *Full-duplex:* comunicación simultánea en ambas direcciones; ideal para sistemas en tiempo real (juegos online, streaming, SCADA, etc.).

2. Necesidad de pausas o interrupciones controladas durante el diálogo.

Control de diálogo

Evita colisiones estableciendo el **orden** en el flujo de mensajes entre usuarios finales.

La elección entre half-duplex y full-duplex forma parte del **control de diálogo**.

Además, controla las **mini-conversaciones** dentro de una sesión completa.

Si las colisiones se vuelven incontrolables, se pasa a modo **half-duplex**.

Recuperación / Resincronización

Permite **acordar puntos de recuperación** entre el origen y el destino para reanudar la sesión desde ese punto en caso de falla o interrupción.

Protocolos utilizados en la capa de sesión

- **RCP** (Remote Copy Protocol)
- **SCP** (Session Control Protocol)
- **ASP** (*obsoleto, usado en Apple*)
- **NFS** (Network File System): permite acceder y compartir archivos en red; puede actuar como cliente o servidor según sea necesario.
- **X Window System:** sistema gráfico de red que también puede comportarse como cliente o servidor.
- **SQL:** establece sesiones entre cliente y servidor para gestionar bases de datos.

Capa de Presentación

La **capa de presentación** se encarga de **formatear y traducir los datos** para que el sistema receptor pueda interpretarlos correctamente.

Actúa como un **traductor** entre las aplicaciones de distintos equipos.

Descripción general

Se ocupa de la **representación de los datos**, garantizando que, aunque los equipos usen diferentes métodos internos para manejar caracteres, números, sonidos o imágenes, la información se **presente de forma reconocible**.

Es la **primera capa que trabaja directamente con el contenido** de la comunicación (sintaxis y semántica de los datos).

Su función principal es **asegurar que los datos transmitidos sean comprensibles y coherentes entre sistemas distintos**.

Funciones principales

(Tema que entra seguro en la tercera pregunta del final)

- **Formateo de datos:** da formato a la información sin modificar su contenido.
- **Cifrado de datos:** codifica los datos para mantener la seguridad (ejemplo: código Morse, cifrado César, AES, etc.).
- **Compresión de datos:** reduce el tamaño de la información utilizando algoritmos específicos (por ejemplo, **JPEG → JPG**, cuando Windows limitaba las extensiones a tres caracteres).

Parcial: preguntas:

- La World Wide Web (www) es una red informática mundial accesible a través de Internet. Está formada por páginas web interconectadas que ofrecen diversos tipos de contenido textual y multimedia.
Verdadero.
- La necesidad de un conjunto de paquetes (flujo) se caracteriza por cuatro parámetros principales: confiabilidad, retardo, fluctuación y ancho de banda.
Verdadero
- HTTPS es un protocolo de la capa de aplicación que se utiliza para:
transferencia de hipertexto seguro
- Para la siguiente URL, señale cuales son las partes que contiene la misma (más de una respuesta) <https://www.salta.gov.ar>:
Protocolo, Sufijo o código de país, Dominio, Tipo de Organización o entidad
- La recuperación de caídas, el establecimiento y liberación de una conexión, forman parte de algunos de los elementos de los protocolos de transporte
Verdadero

- Cuando se establece una conexión entre dos Host por el acuerdo de tres vías , esta conexión se produce siempre de manera ideal:
'Falso'
- ¿Cuáles son los protocolos usados para correo electrónico? (más de una respuesta)
POP3, SMTP
- La conmutación de etiquetas multiprotocolo (MPLS) se agrega en un paquete IP adentro de:
El encabezado
- Marque dos funciones principales de la capa de presentación
Cifrado de datos, Compresión de datos
- La creación de tablas de rutas para encaminar la información a través de la red la realiza:
Capa de Red
- Con el algoritmo de enrutamiento por inundación selectiva el mensaje se repite e inunda toda la red
Falso
- La capa encargada de recuperar la red en caso de una falla es:
Capa de Sesión
- Cuando se solicita una desconexión en una comunicación entre dos Host, los datos dejan de entregarse en ese instante
Falso
- **TCP y UDP** determinan los segmentos y datagramas para la aplicación mediante:
Numero de puertos
- Cuando la capa de red no puede controlar la congestión con algunos de los métodos que emplea, entonces comienza a descartar paquetes
Verdadero
- El intercambio de paginas HTML entre navegador y servidor web lo realiza la Capa de Presentación
Falso
- La Capa de Sesión es la encargada de establecer la comunicación entre las aplicaciones y el usuario que quiera realizar una transferencia de datos
Falso
- El algoritmo de Cubeta con Goteo se diseño para evitar la perdida de paquetes por rebalse en la cubeta y perder tokensen lugar de los paquetes
Falso
- La capa de presentación del modelo OSI se encarga de
Representar los datos para que sean reconocibles
- las diferencias entre UDP y TCP (más de una respuesta)
TCP es más lento en comparación con UDP, UDP no espera una respuesta de datos recibidos, TCP está orientado a la conexión
- Para la medición de desempeño de una red, las muestras deben ser mínimas y se deben realizar en un horario donde existe mucho tráfico entre los equipos de la red a medir
Falso
- ***El algoritmo de ruta más corta***
Calcula la distancia más corta según varios criterios

- Los 3 servicios proporcionados por la capa de transporte son:
Multiplexación, Segmentación y Reensamble, Control de flujo de extremo a extremo