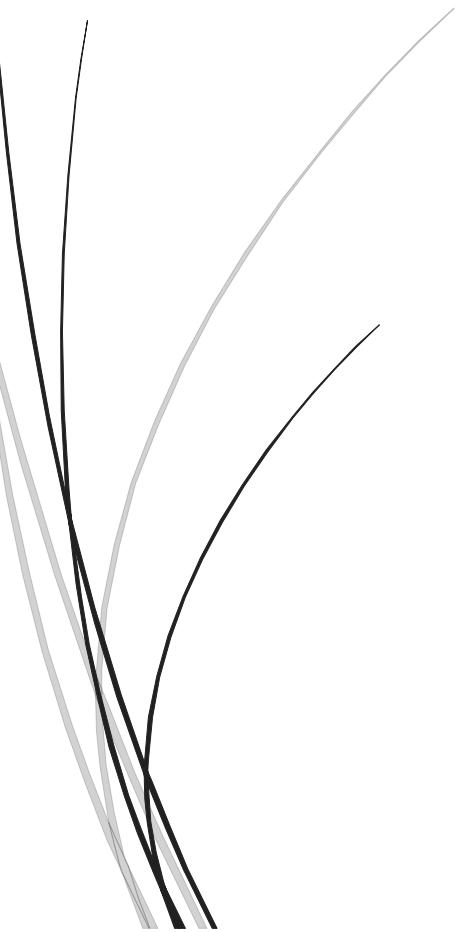


# Redes de computadoras

Introducción, modelos de  
referencia y capas



Julián Camargo  
TDIII

Introducción a redes .....	2
Hardware de red.....	2
Software de red.....	2
Modelos de referencia .....	4
Comparación OSI Y TCP/IP.....	6
Capa física .....	7
Medios de transmisión.....	7
Capa de enlace .....	8
Cuestiones de diseño.....	8
Servicios proporcionados a la capa de red .....	8
Códigos de corrección y detección de errores.....	11
Control de flujo .....	12
Protocolos de enlaces de datos.....	12
Subcapa MAC .....	13
Problema de asignación de canal.....	14
Ethernet .....	20
Capa de red .....	23
Aspectos de diseño de la capa de red .....	23
Algoritmos de enrutamiento.....	26
Ruteo estático .....	30
Ruteo dinámico.....	31
Interconexión de redes.....	32
La capa de red de internet.....	35
Capa de transporte.....	46
El servicio de transporte .....	46
Elementos de los protocolos de transporte.....	48
Los protocolos de transporte de internet: UDP .....	55
Los protocolos de transporte de internet: TCP .....	55
Capa de aplicación.....	58
DNS: sistema de nombres de dominio.....	58
WWW: World Wide Web .....	60
HTTP: protocolo de transferencia de hipertexto .....	64
Modelo cliente/servidor .....	65
Socket.....	66
Llamadas a sistemas de socket .....	66
Socket Stream vs. Socket Datagram .....	69
Apéndice: estructuras de direcciones en sockets .....	69

# INTRODUCCIÓN A REDES

## Hardware de red

Pueden diferenciarse dos **tecnologías de transmisión**, los enlaces de difusión, en los cuales las máquinas comparten el canal de comunicación y los enlace punto a punto, donde se conectan pares individuales de máquinas y donde los paquetes posiblemente tengan que visitar máquinas intermedias para llegar a destino.

Un ejemplo común de enlace de difusión son las redes inalámbricas, donde la comunicación se comparte a través de una región de cobertura y sin embargo, aunque muchas máquinas hayan escuchado el paquete, solo el receptor lo procesará. Estos sistemas también dan la posibilidad de enviar un paquete a todos los destino, mediante un código especial en el campo de dirección, esto se conoce como **broadcasting**. Algunos además dan la posibilidad de transmitir a un subconjunto, lo cual se denomina **multidifusión**.

Otra clasificación posible se basa en la escala de las redes. Podemos diferenciar redes PAN, LAN, MAN y WAN.

Por último, a la conexión de dos o más redes se le conoce como **interred**. El internet es el mejor ejemplo de una interred.

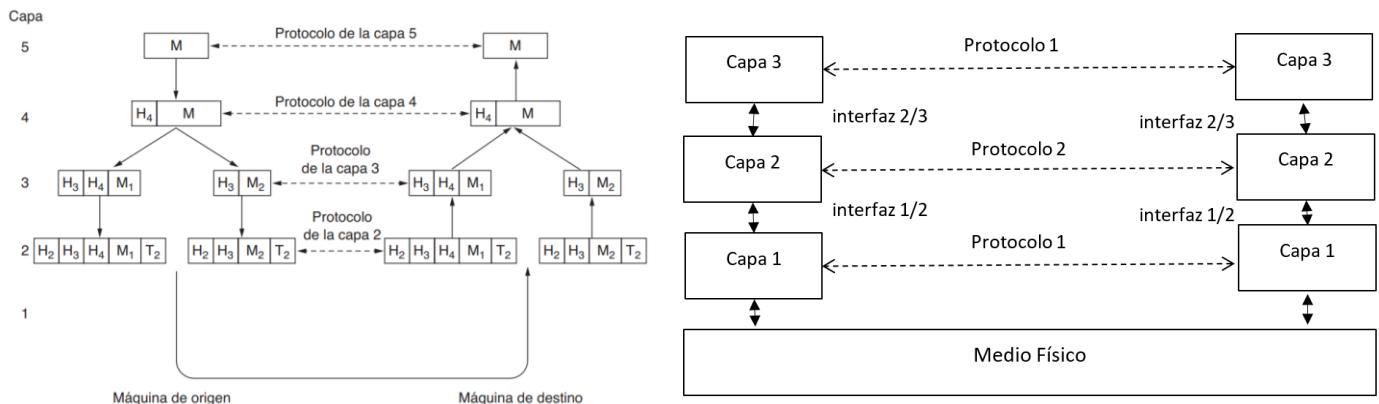
## Software de red

Las primeras redes de computadoras se diseñaron teniendo en cuenta el hardware como punto principal y el software como secundario, sin embargo, esta estrategia ya no funciona. Actualmente el software de red está muy estructurado.

## JERARQUÍA DE PROTOCOLOS

Las redes se organizan en pilas de **capas**, cada una ofrece **servicios** a sus superiores. Capas en la misma posición en diferentes máquinas se denominan **peers**, las cuales se comunican mediante **protocolos**. Entre capas adyacentes, una **interfaz** define las operaciones y servicios primitivos que pone la capa más baja a disposición de la superior. El conjunto de capas y protocolos se denomina **arquitectura de red**.

En la figura siguiente se observa el procedimiento genérico para el envío de mensajes:



1. La capa 5 genera un mensaje y lo pasa a la 4.
2. La 4 coloca un encabezado y lo pasa a capa 3.
3. La 3 lo descompone en paquetes y coloca un encabezado a cada uno.

4. La 2 agrega un encabezado y terminador.
5. La 1 transmite físicamente.

En el receptor el mensaje sube de capa en capa eliminando encabezados y terminadores, hasta llegar a la capa superior igual a como salió de la superior en el emisor.

## ASPECTOS DE DISEÑO PARA LAS CAPAS

- **Confiabilidad:** enfocado en verificar que una red opere correctamente.
  - Códigos de detección de errores: retransmisión de información errónea.
  - Códigos de corrección de errores: recuperación de mensajes erróneos.
    - Ambos usados de capas **inferiores** para protección de paquetes y en **superiores** para verificación de recepción.
  - Enrutamiento: normalmente existen múltiples rutas entre origen y destino. La red debe tomar decisiones de que ruta tomar automáticamente.
- **Direccionamiento:** cada capa debe identificar emisor y receptor en capas altas y bajas.
- **Evolución de la red:** con el tiempo, las redes aumentan su tamaño y emergen nuevos diseños. Para dividir el problema general y ocultar los detalles de la implementación se realiza una distribución de protocolos en capas. Esto también provocó la creación de mecanismos para desensamblar, transmitir y volver a ensamblar mensajes, tema conocido como interconexión de redes.
- **Asignación de recursos:** control de flujo y congestión.
- **Calidad de servicio:** entrega real y/o confirmada, según necesidad.
- **Seguridad:** confidencialidad, autentificación e integridad.

## SERVICIO ORIENTADO A CONEXIÓN VS SIN CONEXIÓN

El servicio **orientado a conexión** se inspira en el sistema telefónico. El usuario establece conexión, la utiliza y la libera. Al establecer la conexión se negocian los parámetros de la misma.

- Conexión no confiable.
- Secuencia de mensajes confiable: mensajes con límites.
- Flujo de bytes confiables: mensajes sin límites.

El servicio **sin conexión** se modela a partir del sistema postal. Los mensajes llevan dirección de destino completa y son enrutados independientemente.

- No confiable sin conexión: red de datagramas.
- Datagrama con confirmación.
- Solicitud – respuesta (modelo cliente – servidor).

## PRIMITIVAS DE SERVICIO

Un servicio es un **conjunto de primitivas**. Son diferentes para servicios con y sin conexión.

Primitiva	Significado
LISTEN	Bloquea en espera de conexión entrante
CONNECT	Establece conexión con peer en espera
ACCEPT	Acepta conexión de peer entrante
RECEIVE	Bloquea en espera de mensaje
SEND	Envía mensaje a peer
DISCONNECT	Termina conexión

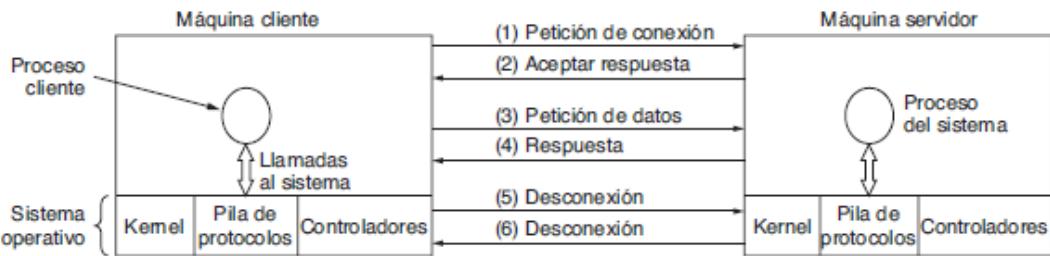


Figura 1-18. Una interacción cliente-servidor simple mediante el uso de datagramas con confirmación de recepción.

## RELACIÓN ENTRE SERVICIOS Y PROTOCOLOS

Un servicio es un conjunto de primitivas que una capa proporciona a la capa superior a ella. Define qué operaciones puede realizar la capa en beneficio de sus usuarios, pero no dice nada sobre cómo se implementan esas operaciones. Un servicio se relaciona con una interfaz entre 2 capas, donde la inferior es el proveedor de servicio y la superior es el usuario.

Un protocolo es un conjunto de reglas que rigen el formato y significado de los paquetes que intercambian peers. Las entidades los utilizan para implementar sus definiciones de servicios. Pueden cambiar sus protocolos a voluntad, siempre y cuando no cambien el servicio para sus usuarios.

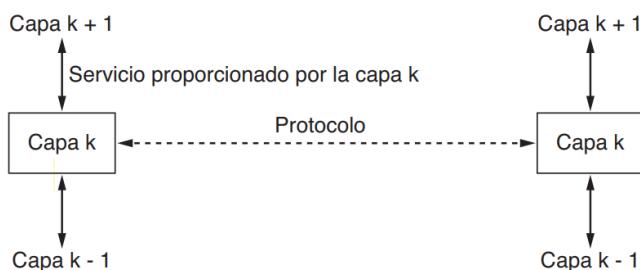


Figura 1-19. La relación entre un servicio y un protocolo.

## Modelos de referencia

El modelo OSI es bastante general y sigue siendo válido, sin embargo sus protocolos no se utilizan. El modelo TCP/IP por otro lado no se utiliza mucho, pero sus protocolos son ampliamente usados.

### MODELO DE REFERENCIA OSI

Posee 7 capas. Los principios del modelo son abstracción, funciones bien definidas, protocolos estandarizados, límites para menor flujo de información y cantidad suficiente de capas para tener pocas funciones por capa.

En capas inferiores (1 a 3), los protocolos se encuentran entre una máquina y su vecino inmediato, no entre máquina origen y máquina destino. Las capas 4 a 7 son extremo a extremo.

Estas capas son:

- **Capa física:** transmisión de bits, interfaces mecánica, eléctrica, etc.
- **Capa de enlace de datos:** tramas, subcapa LLC y MAC.
- **Capa de red:** paquetes, conexión de redes heterogéneas, manejo de congestión, etc.
- **Capa de transporte:** determina tipo de servicio.
- **Capa de sesión:** establecimiento de sesiones entre usuarios.
- **Capa de presentación:** sintaxis y semántica de información transmitida.
- **Capa de aplicación:** protocolos como HTTP y SMTP.

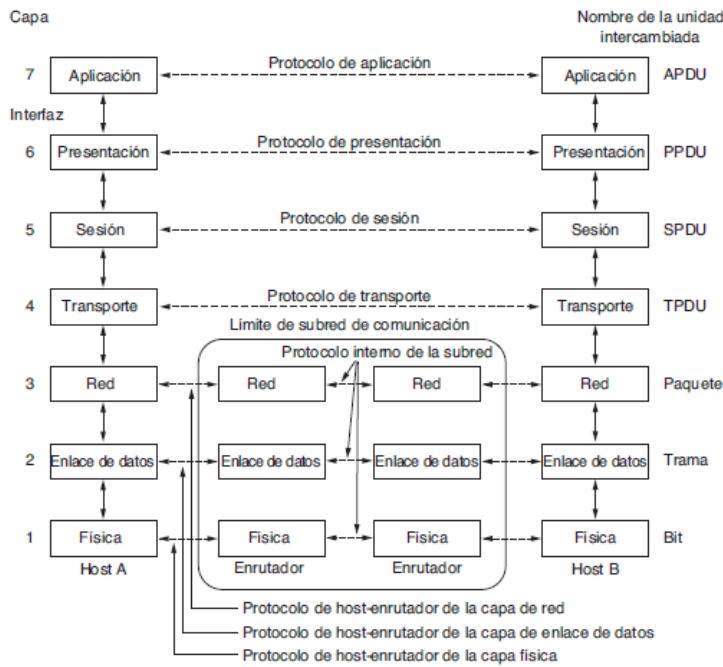


Figura 1-20. El modelo de referencia OSI.

Ubicación de dispositivos LAN dentro de este modelo:

Capa	Dispositivo
Aplicación	
Sesión	
Transporte	
Red	Routers
Enlace de datos	Switch, Bridge, NIC
Física	Repetidor, HUB

Un **HUB** es un repetidor multipuerto. Recibe datos de una máquina y los retransmite a todas las demás. Actúa solo en capa física.

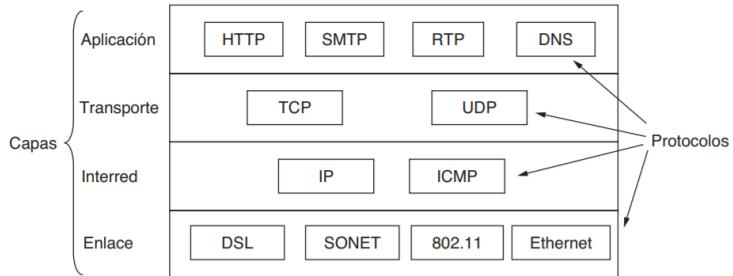
Un **Switch** es un puente (bridge) multipuerto. Interconecta 2 o más segmentos de red. Realiza una transmisión más eficiente, tomando decisiones según la MAC.

## MODELO TCP/IP

Describe los protocolos de red. Se usa en internet y se creó con los objetivos de conectar varias redes, sobrevivir a la pérdida de hardware de la subred y tener una arquitectura flexible.

Posee las siguientes capas:

- **Capa de enlace**: conmutación de paquetes basada en una capa sin conexión. Describe que los enlaces llevar a cabo, es una interfaz entre hosts y enlaces de transmisión.
- **Capa de internet**: permite que los paquetes viajen de forma independiente. Define formato de paquete y protocolo IP. Su principal tarea es entregar paquetes IP, considerando ruteo y congestión.
- **Capa de transporte**: permite a peers en origen y destino tener una conversación. Se definieron dos protocolos extremo a extremo:
  - **TCP**: confiable orientado a conexión. Permite flujo de bytes sin errores. Segmenta el flujo en mensajes discretos. Maneja el control de flujo para no saturar receptores.
  - **UDP**: sin conexión, usado en cliente – servidor.
- **Capa de aplicación**: protocolos de alto nivel.



Modelo OSI	Modelo TCP/IP
Aplicación	
Presentación	
Sesión	
Transporte	Transporte
Red	Internet
Enlace de datos	Enlace de datos
Física	

## MODELO DEL LIBRO

La fortaleza del modelo OSI es el modelo, mientras que la del TCP/IP son los protocolos. Se utiliza entonces el siguiente modelo híbrido:

- **Capa física:** transmisión de bits.
- **Capa de enlace:** envío de mensaje de longitud finita.
- **Capa de red:** combina redes de redes en interredes. Búsqueda de rutas (IP).
- **Capa de transporte:** garantías de entrega de capa de red. Abstracciones de entrega (TCP).
- **Capa de aplicación:** programas que hacen uso de la red (HTTP, SMTP, RTP, etc.).

## Comparación OSI Y TCP/IP

	OSI	TCP/IP
Descripción	<ul style="list-style-type: none"> <li>• Distinción clara entre servicios, interfaces y protocolos. La capa debe decidir los protocolos a usar.</li> <li>• Protocolos ocultos de mejor forma y reemplazables por avances tecnológicos (modelo creado antes que los protocolos).</li> <li>• Soporta comunicación con y sin conexión en capa de red pero solo con conexión en capa de transporte (donde importa).</li> </ul>	<ul style="list-style-type: none"> <li>• Describe los protocolos existentes (llegó después que los protocolos).</li> <li>• No encaja en otra pila de protocolos.</li> <li>• Soporta solo sin conexión en capa de red, pero ambos en capa de transporte, lo cual importa en protocolos simples de petición – respuesta.</li> </ul>
Crítica	<ul style="list-style-type: none"> <li>• Mal momento para establecer un estándar.</li> <li>• Dos capas casi vacías (sesión y presentación) y dos casi llenas (enlace y red).</li> <li>• Protocolos complejos. Direccionamiento, control de flujo y de errores recurrentes en cada capa.</li> <li>• Malas implementaciones, pesadas y lentas.</li> </ul>	<ul style="list-style-type: none"> <li>• No sirven para redes con nuevas tecnologías porque no diferencia con claridad los conceptos de servicios, interfaces y protocolos.</li> <li>• No es genérico ni apropiado para describir una pila diferente.</li> <li>• <b>La capa de enlace es solo una interfaz entre la de red y la de enlace de datos.</b></li> <li>• No distingue la capa física de al de enlace de datos.</li> </ul>

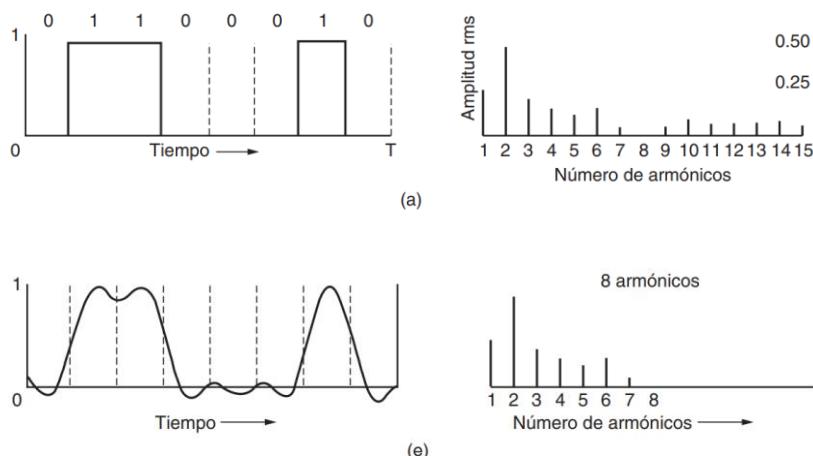
# CAPA FÍSICA

Es la base de la red, define la interfaz eléctrica, de temporización, etc. Cuando contamos con señales de BW limitado tenemos componentes de la señal que disminuyen, provocando distorsión. El rango de frecuencias que se transmite sin una atenuación considerable es el ancho de banda. La frecuencia de corte es aquella a la cual la potencia disminuye a la mitad.

El BW es una propiedad física del medio. La información que puede transportarse solo depende de él.

Todas las señales se atenúan al ser transmitidas, sin embargo lo grave es perder potencia en armónicos en distinto grado, lo cual produce distorsión.

En transmisión digital el objetivo es transmitir una señal de la cual pueda reconstruirse la secuencia de bits, por lo cual no se necesitan todos los armónicos.



Un canal con una tasa de  $b \frac{[bits]}{[seg]}$  requiere de  $\frac{n}{b} [seg]$  para enviar  $n [bits]$  de a 1 bit a la vez. La frecuencia fundamental de esta señal entonces es  $\frac{b}{n} [Hz]$ .

El ancho de banda para el área de computación es la **tasa de datos máxima de un canal** en  $\frac{bits}{seg}$ . Esta puede deducirse según se tome o no en cuenta el ruido presente en el canal (Shannon o Nyquist respectivamente).

- **Nyquist:** al pasar una señal por un pasabajos de ancho de banda  $B$ , la señal puede reconstruirse con  $2B$  muestras por segundo. Si la señal cuenta con  $V$  niveles discretos:  $Tasa = 2B \cdot \log_2(V \frac{bits}{seg})$
- **Shannon:** la tasa de datos máxima de un canal ruidoso de ancho de banda  $B$  con relación señal a ruido  $SNR$  es:  $Tasa = B \cdot \log_2(1 + SNR)$

## Medios de transmisión

A grandes rasgos, los medios se agrupan en medios de transmisión guiados y en medios no guiados.

- **Medios de transmisión guiados:**

- Medios magnéticos.
- Par trenzado.
- Cable coaxial.
- Líneas eléctricas.
- Fibra óptica.

- **Medios de transmisión inalámbricos:**

- Radiodifusión.
- Microondas.
- Infrarrojo.
- Transmisión por luz laser.

## CAPA DE ENLACE

Los principios de diseño de esta capa se enfocan en algoritmos para lograr una comunicación confiable y eficiente de unidades completas de información llamadas **tramas** entre dos máquinas adyacentes. Esta capa toma un medio y lo transforma en una línea libre de errores.

### Cuestiones de diseño

Utiliza los servicios de la capa física para enviar y recibir bits a través de los canales de comunicación. Tiene las siguientes funciones:

- Proporciona a la capa de red una interfaz de servicio bien definida.
- Maneja errores de transmisión.
- Regula el flujo de datos para evitar saturaciones.

Para cumplir estas funciones, toma paquetes de la capa de red y los encapsula en tramas. El manejo de las tramas es la tarea más importante de esta capa.

La IEEE divide la capa de enlace OSI en las subcapas MAC (control de acceso al medio) y LLC (control de enlace lógico), donde:

- LLC participa del encapsulamiento.
- MAC selecciona cual host transmitirá.

### Servicios proporcionados a la capa de red

El servicio principal es la transferencia de datos de la capa de red en la máquina origen a la capa de red de la máquina de destino.

La capa de enlace de datos puede diseñarse para ofrecer varios servicios. Los servicios varían de un protocolo a otro. Tres posibilidades son:

- **Sin conexión ni confirmación de recepción:** apropiado si la tasa de error es baja y el retraso muy influyente.
- **Sin conexión con confirmación de recepción:** se confirma individualmente la recepción de cada trama. Útil en canales no confiables como WiFi. La confirmación es una optimización, no un requerimiento. La desventaja es el tiempo de espera del acuse de recibo.
- **Orientado a conexión con confirmación de recepción:** evita la espera del acuse de recibo estableciendo una conexión antes de transferir datos. Cada trama es enumerada y la capa garantiza la llegada de cada trama a destino en orden y sin repetirse. Ofrece un flujo de bits confiable. Es apropiado para enlaces largos y no confiables. La transferencia pasa por 3 fases:
  - Establecimiento de conexión.
  - Transmisión de tramas.
  - Liberación de conexión.

## ENTRAMADO

Para proveer servicio a la capa de red, la capa de enlace de datos debe usar el servicio que la capa física le provee. La capa física acepta un flujo de bits putos y trata de entregarlo a destino. En canales ruidosos (mayoría de los casos), la capa física agrega cierta redundancia a sus señales para reducir la tasa de error de bits a un nivel tolerable. No se garantiza que el flujo de bits recibido por la capa de enlace de datos esté libre de errores, por lo que es responsabilidad de esta última capa detectar y, de ser posible, corregir los errores.

El método común es que la capa de enlace de datos divida el flujo de bits en tramas discretas, calcule la suma de verificación para cada una y la incluya en la trama al momento de transmitirla. Al llegar a destino, se recalcula la suma de verificación, si la nueva suma es distinta de la contenida en la trama, la capa de enlace sabe que ha ocurrido un error.

Un buen diseño de división de flujo debe facilitar al receptor el proceso de encontrar el inicio de las nuevas tramas y utilizar una pequeña parte del ancho de banda del canal.

Se muestran 4 métodos de entrampado:

- **Conteo de bytes:** un campo en el encabezado indica el número de bytes de la trama. El problema es que si por error este valor cambia, se pierde la sincronía en la transmisión.

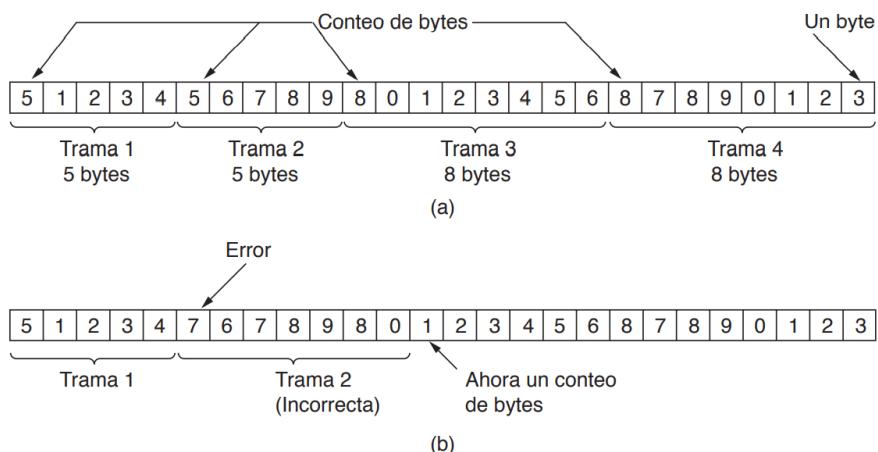
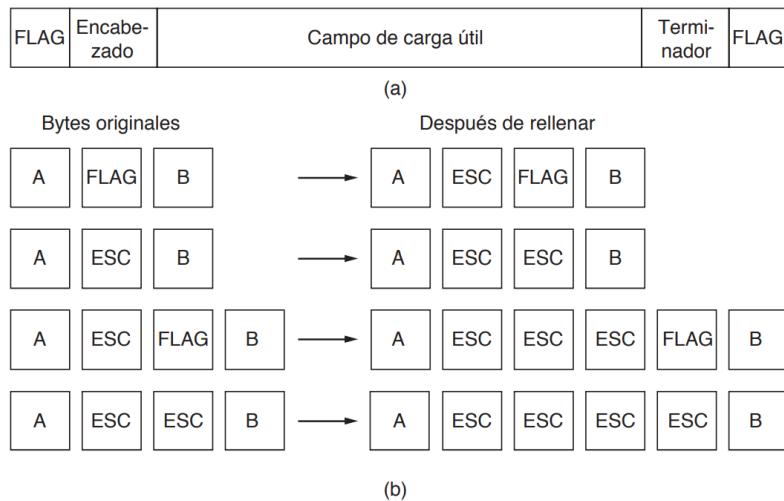


Figura 3-3. Un flujo de bytes. (a) Sin errores. (b) Con un error.

- **Bandera de bytes:** dos bytes de bandera consecutivos indican el final de una trama y el comienzo de otra. Si un byte de bandera aparece en los datos se coloca un carácter de escape (ESC). Si entre los datos hay un byte ESC, se coloca otro carácter de ESC. La capa de enlace en el receptor quita los bytes ESC antes de entregar los datos a la capa de red. Esto se denomina relleno de bytes.



**Figura 3-4.** (a) Una trama delimitada por bytes bandera. (b) Cuatro ejemplos de secuencias de bytes antes y después del relleno de bytes.

- **Bandera de bits:** resuelve la desventaja del anterior, de tener que ocupar siempre 8 bits para los bytes especiales. Es un entramado más barato, donde cada trama comienza y termina con un patrón de bits especial 01111110 o 0x7e. Esta secuencia nunca puede ocurrir en los datos ya que cada vez que la capa de enlace de datos del emisor encuentra 5 bits en 1 en los datos, inserta un 0 como relleno. Este cero se denomina bit de escape y es eliminado en el receptor. La desventaja radica en el trabajo que implica manejar bits.

```
(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
      ↑           ↑           ↑           ↑           ↑
      0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0
      ↑           ↑           ↑           ↑           ↑           ↑
      Bits de relleno
(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
```

**Figura 3-5.** Relleno de bits. (a) Los datos originales. (b) Los datos, según aparecen en la línea. (c) Los datos, como se almacenan en la memoria del receptor después de quitar el relleno.

Un efecto secundario del relleno de bits y bytes es que la longitud de la trama depende ahora del contenido de los datos que transmite.

- **Violación de codificación de la capa física:** la codificación de bits como señales incluye a menudo redundancia para ayudar al receptor. Esta redundancia significa que algunas señales no ocurrirán en los datos regulares, por lo que podemos usarlas como delimitadores (violaciones de código). Así no se requiere de relleno en los datos.

## CONTROL DE ERRORES

Debe asegurarse que las tramas se entreguen en el orden apropiado a la capa de red destino. La forma normal es proporcionar retroalimentación al emisor. Tenemos las siguientes posibilidades:

- **Servicio sin conexión ni confirmación:** no garantizan nada.
- **Servicio sin conexión con confirmación:** puede suceder que se pierdan la trama de ida o la de confirmación, en cuyo caso se tendrán tramas duplicadas.
- **Servicio orientado a conexión con confirmación:** no hay problemas, se enumeran las tramas.

## CONTROL DE FLUJO

Debe tenerse en cuenta la posibilidad de que un emisor rápido sature a un receptor lento. Se utilizan dos métodos para evitarlo:

- **Control de flujo basado en retroalimentación:** el receptor regresa información al emisor para autorizarlo a enviar más datos o indicarle su estado.
- **Control de flujo basado en tasa:** el protocolo tiene un mecanismo integrado que limita la tasa a la que el emisor puede transmitir los datos.

## Códigos de corrección y detección de errores

Para la detección de  $d$  errores se necesita un código de distancia  $d + 1$ .

Para la corrección de  $d$  errores se necesita un código de distancia  $2d + 1$ .

## CÓDIGOS DE CORRECCIÓN DE ERRORES

Debe incluirse información adicional redundante para que el receptor pueda deducir cuales deberían ser los datos transmitidos. Una trama consiste de  $m$  bits de datos y  $r$  bits redundantes. En un **código de bloque** los bits  $r$  se calculan en función únicamente de los bits  $m$  con los que se asocian. En un **código sistemático** los  $m$  se envían con los  $r$ , en lugar de codificarse antes. En un **código lineal** los  $r$  bits se calculan como una función lineal de los  $m$  bits de datos.

Poseen las siguientes características:

- **Longitud de boque n:**  $n = m + r$ .
- **Palabra codificada:** unidad de  $n$  bits que contiene los datos y la verificación.
- **Tasa de código:** fracción de la palabra codificada que lleva información no redundante ( $\frac{m}{n}$ ).
- **Distancia de Hamming:** cantidad de posiciones de bits en las que difieren dos palabras codificadas. Se calcula haciendo una XOR y contando la cantidad de bits 1. La menor distancia entre 2 palabras es la de todo el código.

Los siguientes códigos de corrección de errores son bloques lineales sistemáticos:

- **Hamming:** agrega bits de chequeo de paridad en las potencias pares (los bits potencia de 2 son de verificación), los cuales obligan a que el grupo de bits sea de paridad par o impar. La desventaja es que se realiza en software.
- **Convolucional:** al no ser un código de bloque no tiene límite de codificación. La salida depende de los bits de entrada actuales y pasados. El número de bits pasados de los que depende se llama restricción de código. Por cada bit entrante salen 2 que son la suma XOR de la entrada y los estados internos. Se duplica la información.
- **Reed – Solomon:** operan sobre símbolos de  $m$  bits y se basan en una ecuación polinómica de grado  $n$ .
- **Chequeo de paridad de baja densidad (LPDC):** códigos de bloque lineales. Cada bit de salida se forma solo a partir de una fracción de los bits de entrada. Conduce a una representación matricial del código. Es para grandes volúmenes de información.

## CÓDIGOS DE DETECCIÓN DE ERRORES

Deben permitir que el receptor detecte el error y solicite retransmisión:

- **Paridad:** códigos con 1 bit de paridad tienen una distancia de 2, por lo que pueden detectar 1 solo bit de error. Si existe una ráfaga de errores, la probabilidad de detectar el error es del 50%. Se puede mejorar esto si cada bloque a enviar se trata como una matriz de  $n \times k$ . Calculando y enviando un bit de paridad por fila podemos detectar hasta  $k$  bits siempre que haya 1 bit de error por fila. También puede calcularse la paridad en un orden distinto al transmitido, esto se denomina **intercalado**.
- **Checksum:** indica un grupo de bits de verificación asociados con un mensaje. Se coloca generalmente al final de mensaje, como complemento de la función suma. Los errores se pueden detectar al sumar toda la palabra, si el resultado es cero, no se ha detectado error.
- **Comprobación de redundancia cíclica (CRC):** emisor y receptor acuerda un polinomio generador  $G(x)$ , de longitud menor a la trama.

## Control de flujo

Hay dos formas de solucionar los casos donde un emisor rápido satura a un receptor lento.

- Control de flujo basado en realimentación: el receptor da autorización al emisor a enviar.
- Control de flujo basado en tasa: se limita la tasa a la que el emisor puede transmitir.

Ejemplos de los esquemas de control de flujo basado en realimentación son los mecanismos de ventana deslizante. En estos se aprovecha la bidireccionalidad de la transmisión para que el emisor conozca el estado del receptor (y sepa si enviar o no tramas) a la vez que almacena las tramas enviadas por si se requiere alguna retransmisión.

## Protocolos de enlaces de datos

La mayoría de las infraestructuras de redes WAN se basa en líneas punto a punto.

- **Enlace SONET:** envían paquetes de enlace de fibra en redes WAN.
- **Enlace ADSL:** operan en el lazo local de la red telefónica en un extremo de internet.

## PAQUETES SOBRE SONET

El protocolo estándar punto a punto **PPP** envía paquetes a través de estos enlaces. PPP se ejecuta en enrutadores IP para lograr el mecanismo de entrampado para diferenciar los paquetes ocasionales del flujo de bits continuo en el que se transportan. PPP se utiliza para manejar la configuración de detección de errores en los enlaces, soporta múltiples protocolos, permite autenticación y tiene otras funciones. Provee 3 características principales:

- Un método de entrampado que delinea final e inicio de las tramas. El formato de trama maneja la detección de errores.
- Un protocolo de control de enlace LCP que activa líneas, prueba, negocia opciones y desactiva líneas no necesarias.
- Un mecanismo para negociar opciones de la capa de red con independencia del protocolo de red a utilizar. El método elegido debe tener un Protocolo de control de red NCP distinto para cada capa de red soportada.

Además permite autenticación y asignación dinámica de IP.

El formato de una trama PPP es:

Bytes	1	1	1	1 a 2	Variable	2 a 4	1
	Bandera 01111110	Dirección 11111111	Control 00000011	Protocolo	Carga útil	Suma de verificación	Bandera 01111110

Figura 3-24. El formato de trama completa de PPP para la operación en modo no numerado.

- **Bandera:** 0111 1110 o 0x7E.
- **Dirección:** 1111 1111 para indicar que todas las estaciones deben aceptar la trama, evitando asignar direcciones en capa de enlace.
- **Control:** predeterminado 0000 0011 que indica trama no numerada.
- **Protocolo:** clase de paquete de campo de carga útil. Comenzando en 0 para IPv4, IPv6, etc. y con 1 para configuración.
- **Carga útil:** 1500 bytes de longitud por defecto. Se suele negociar mediante LCP al establecer la línea.
- **Checksum.**

PPP es un mecanismo de entramado que puede transportar paquetes de varios protocolos a muchos tipos de capas físicas. Es sin conexión ni confirmación.

La carga útil de PPP se mezcla aleatoriamente antes de insertarse en la carga útil de SONET, así la probabilidad de que un usuario ocasione problemas enviando largas secuencias de ceros es baja.

Antes de transportar tramas PPP en líneas SONET debe establecerse y configurarse el enlace PPP.

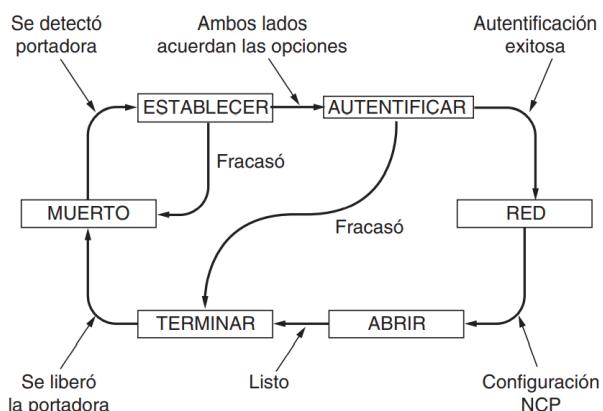


Figura 3-25. Diagrama de estado para activar y desactivar un enlace PPP.

El enlace inicia en MUERTO (sin conexión a capa física).

Al establecer conexión pasa a ESTABLECER, donde los peers de PPP intercambian paquetes LCP para seleccionar las opciones PPP para el enlace.

Si la negociación es exitosa, llega a AUTENTIFICAR, las partes pueden verificar identidades si lo desean.

Si la autenticación es exitosa, el enlace llega a estado RED y se envían paquetes NCP para configurar la capa de red.

Cuando el enlace llega a ABRIR se transportan los paquetes IP en tramas PPP mediante la línea SONET.

Al terminar la transmisión el enlace pasa a TERMINAR y de ahí a MUERTO al desactivar la conexión de la capa física.

## SUBCAPA MAC

En una red de difusión debe determinarse quien puede utilizar el canal cuando hay competencia por él. Los protocolos para determinar quién toma el canal pertenecen a esta subcapa. La misma determina las formas en que las estaciones van a compartir un medio.

# Problema de asignación de canal

Debe establecerse una forma de asignar un canal entre usuarios competidores, esto puede hacerse mediante dos tipos de asignaciones, estática y dinámica.

## ASIGNACIÓN ESTÁTICA

Divide la capacidad del canal mediante un esquema de multiplexación como los siguientes:

- **FDM (Multiplexación por división de frecuencia):** se divide el BW entre los N usuarios, dando una parte fija de igual tamaño a cada uno.
  - Ventajas:
    - No hay interferencia entre canales.
    - Mecanismo es sencillo y eficiente para pocos usuarios.
  - Desventajas:
    - Sí varía el número de usuarios se desperdicia BW o se niega permiso por falta del mismo.
    - En ráfagas intensas los canales estarán inactivos casi todo el tiempo.
    - Pobre desempeño debido al retardo promedio igual al número de divisiones.
- **TDM (Multiplexación por división de tiempo):** a cada usuario se le da una ranura de tiempo. En caso de no usarla, se desperdicia.
- **División física de la red:** aumenta el retardo promedio al dividir la capacidad de la red.

## ASIGNACIÓN DINÁMICA

Se consideran los siguientes supuestos:

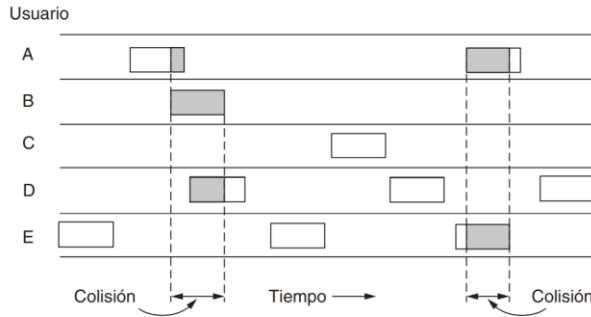
- Tráfico independiente: N estaciones independientes, transmiten, se bloquean y no hacen nada hasta que la trama se transmite con éxito.
- Canal único: un solo canal disponible y cada estación con la misma capacidad para transmitir.
- Colisiones observables: dos tramas transmitidas simultáneamente resultan en una alterada que debe repetirse.
- Tiempo continuo o ranurado: en el primero la trama se transmite en cualquier momento. En el otro se da un quantum de tiempo para que comience.
- Con o sin detección de portadora: con detección las estaciones pueden saber si el canal está ocupado antes de transmitir.

Los protocolos de asignación dinámica son:

### ALOHA

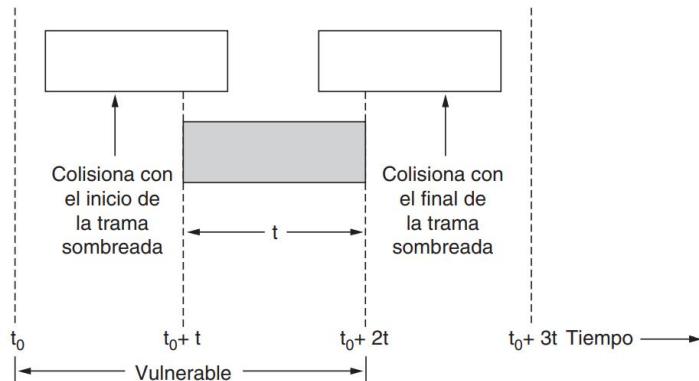
Existen dos versiones, puro y ranurado.

- **ALOHA puro:** se transmite apenas se tengan datos para enviar. La estación central, al recibir la trama, la difunde a todas las emisoras, si pueden escucharla significa que pasó, su fue destruida el emisor espera un tiempo aleatorio y vuelve a transmitir. El **tiempo de trama** es el tiempo necesario para transmitir cada trama estándar de longitud fija. El máximo uso posible del canal es del 18%.



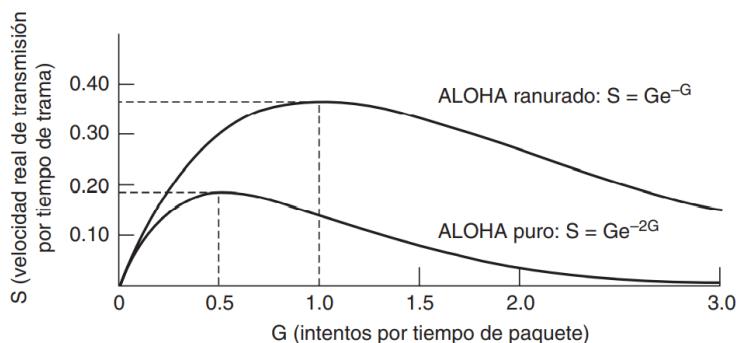
**Figura 4-1.** En ALOHA puro, las tramas se transmiten en tiempos completamente arbitrarios.

Con carga baja se tendrán pocas colisiones y por ende, pocas retransmisiones. Con carga alta tendremos muchas colisiones y retransmisiones. Con todas las cargas, la velocidad real de transmisión dependerá de la probabilidad de que una transmisión tenga éxito.



**Figura 4-2.** Período vulnerable para la trama sombreada.

- **ALOHA ranurado:** divide el tiempo en ranuras discretas, cada cual corresponde a una trama. Requiere que los usuarios acuerden los límites de ranura. Una forma de sincronizar es mediante una estación especial que emita una señal de comienzo para cada intervalo (AP), donde si una estación está transmitiendo la otra tiene que esperar el tiempo de ranura. Reduce el tiempo vulnerable a la mitad y si opera con mayores valores de media de tramas nuevas y reenviadas, reduce el número de ranuras vacías, pero aumenta exponencialmente las colisiones.



**Figura 4-3.** Velocidad real de transmisión contra tráfico ofrecido en los sistemas ALOHA.

La tasa de salida de tramas es menor en ALOHA puro ya que pueden ocurrir colisiones en cualquier momento.

## Protocolos de acceso múltiple con detección de portadora

En ALOHA puro el mejor aprovechamiento posible del canal es del 36%, debido a que las estaciones transmiten sin importar lo que hagan las demás. Los protocolos donde las estaciones escuchan una portadora antes de transmitir, se denominan **protocolos de detección de portadora** y logran un mayor aprovechamiento del canal.

- **CSMA persistente -1:** se escucha el canal antes de enviar una trama. Si está inactivo se transmite y si está ocupado se espera a que se desocupe. En caso de colisión, la estación emisora espera un tiempo aleatorio y comienza de nuevo. Se denomina **persistente** porque la estación lista para transmitir escucha el canal todo el tiempo y **-1** porque transmite con una probabilidad de 1 si el canal está inactivo (siempre).

Esto trae el problema de potenciales colisiones si más de una estación se encuentra lista para transmitir y detecta el canal inactivo.

Otro problema es el retardo entre estaciones, ya que una estación puede comenzar a transmitir y otra estar lista antes de que los datos de la primera le lleguen, por lo que detectaría un canal inactivo. Esto depende del número de tramas que quepan en el canal (producto ancho de banda – retardo).

- **CSMA no persistente:** la estación lista para transmitir escucha el canal, si ya está en uso vuelve a escucharlo luego de un tiempo aleatorio. Posibilita un mejor uso del canal pero incrementa el retardo.
- **CSMA persistente -p:** se aplica a canales ranurados. La estación lista escucha el canal. Si está inactivo transmite con una probabilidad de  $p$  o se posterga hasta la siguiente ranura con una probabilidad de  $q = (1 - p)$ . Si posterga, en la siguiente ranura vuelve a transmitir o postergar con las mismas probabilidades. Esto se repite hasta que transmite o que otra estación toma el canal, en cuyo caso la estación actúa como si hubiera ocurrido una colisión.

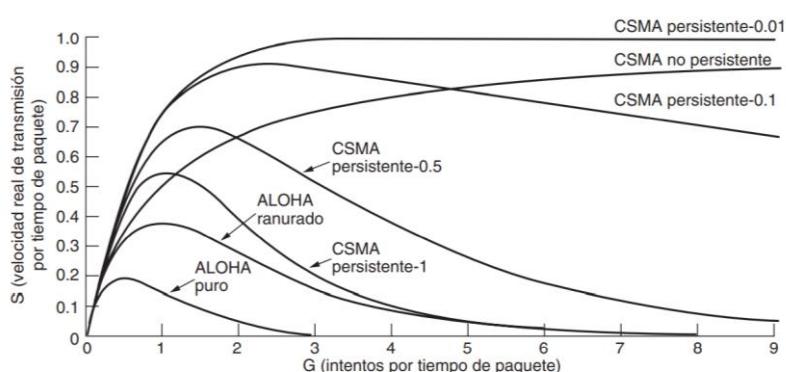


Figura 4-4. Comparación de la utilización del canal contra la carga para varios protocolos de acceso aleatorio.

- **CSMA con detección de colisiones (CSMA/CD):** los protocolos anteriores siguen teniendo colisiones cuando las estaciones transmiten a la vez. CSMA/CD es la base de LAN Ethernet. La detección de colisiones es un proceso analógico, la estación debe escuchar al canal mientras transmite. Si la señal que recibe es distinta a la que envía hay una colisión.

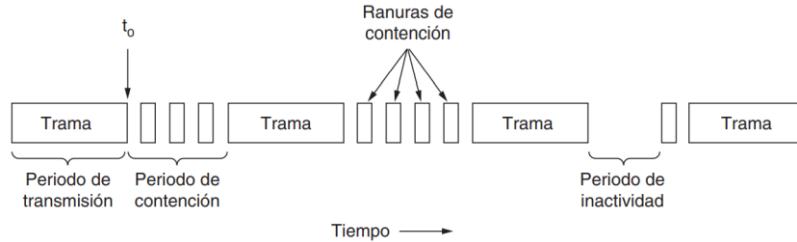


Figura 4-5. CSMA/CD puede estar en estado de contención, de transmisión o inactivo.

En  $t_0$  una estación terminó de transmitir. Si más de una estación deciden transmitir habrá colisión. Si una estación detecta una colisión aborta la transmisión, espera un tiempo aleatorio e intenta de nuevo. El modelo consiste en períodos alternantes de contención y transmisión con períodos de inactividad.

Para asegurarse de haber tomado el canal sin colisiones, una estación debe poder transmitir por un tiempo de  $2\tau$ , siendo  $\tau$  el tiempo de propagación a la estación más lejana. Puede pensarse entonces en CSMA/CD como un ALOHA ranurado con ranuras de ancho igual a  $2\tau$ .

Solo funciona para conexión alámbrica debido a la imposibilidad de detectar portadora en medios inalámbricos.

### Protocolos libres de colisiones

Resuelven las colisiones incluso en el período de contención.

- **Protocolos de mapa de bits (reservación):** cada período de contención consta de N ranuras. Se coloca un bit en 1 en la ranura correspondiente a la estación que quiere transmitir. Se denomina de reservación porque primero se difunde el interés por transmitir y luego la transmisión en sí.

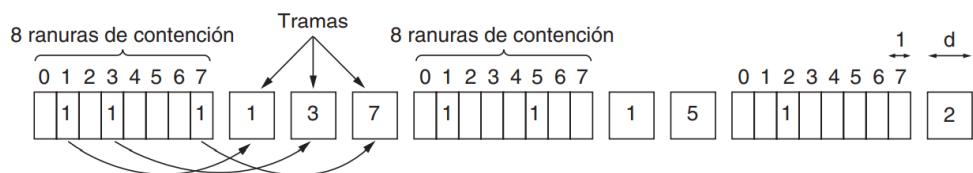
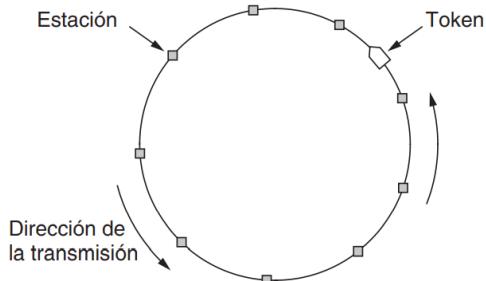


Figura 4-6. El protocolo básico de mapa de bits.

No es ideal su uso para muchas estaciones ya que la trama de contención podría resultar más grande que el dato mismo. A diferencia de CSMA/CD, la trama de contención no es una trama colisionada sino una trama con información.

- **Paso de token:** pasaje de un mensaje (token) de una estación a otra en un orden predefinido. Este token representa el permiso para enviar. Una estación lista para enviar al recibir el token envía la trama y luego pasa el token a la siguiente estación. Si no está lista solo pasa el token. En el protocolo **token ring** las estaciones se conectan en un solo anillo. Las estaciones listas al recibir el token transmiten en su misma dirección y luego lo pasan. Para que una trama no circule indefinidamente en el anillo debe ser quitada, lo cual puede hacerlo la misma estación emisora o la receptora.

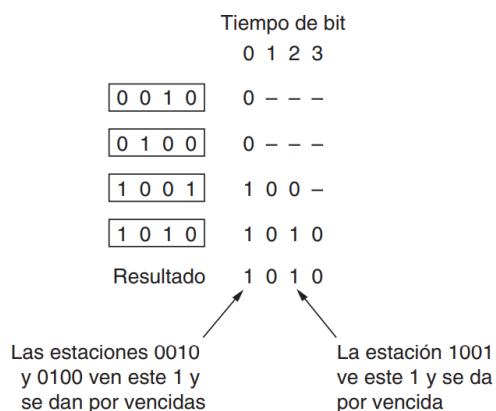
En este protocolo las ranuras de contención y las tramas de un ciclo están entremezcladas. Todas las posiciones del anillo son equivalentes, no hay preferencias por estaciones de mayor o menor numeración.



**Figura 4-7.** El protocolo *token ring*.

Si el canal que conecta las estaciones es un solo bus externo, se trata de la variante conocida como **token bus**.

- **Conteo descendente binario:** los anteriores sobrecargan con 1 bit por estación, por lo que no escalan bien. Este protocolo utiliza direcciones de estación binarias con un canal que combine las transmisiones. La estación lista difunde su dirección, comenzando por el bit más significativo. A los bits de todas las estaciones se les aplica una OR por el canal cuando se envían al mismo tiempo. Si una estación ve que una posición bit de orden alto de su dirección que era 0 ha sido sobrescrita con 1 se da por vencida, en caso contrario verifica el siguiente bit. Esto se repite con todos los bits. La estación vencedora puede transmitir.



**Figura 4-8.** El protocolo de conteo descendente binario. Un guión indica un silencio.

### Protocolos de contención limitada

En condiciones de carga ligera, la contención (ALOHA) posee un bajo retardo y menos colisiones. Al aumentar la carga, el arbitraje del canal se vuelve mayor, siendo menos conveniente.

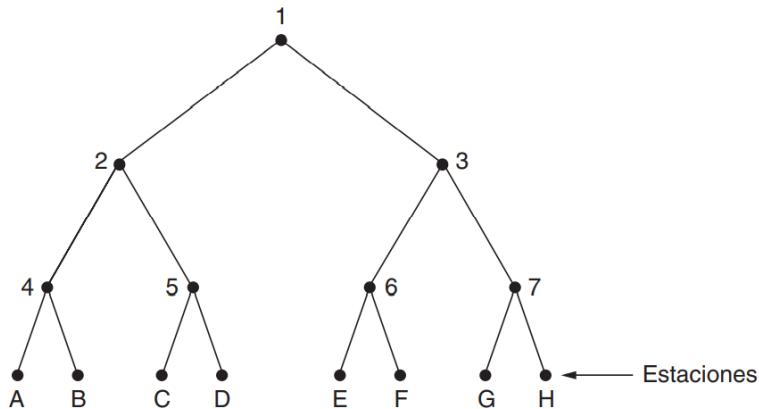
En protocolos libres de colisiones con carga baja, el retardo es alto, pero si aumenta la carga mejora la eficiencia, ya que las sobrecargas son fijas.

Los protocolos de contención limitada usan contención cuando la carga es baja y una técnica libre de colisiones cuando es alta. Disminuyen la competencia entre estaciones por un mismo canal dividiendo en grupos de estaciones. Cada grupo compite por una ranura disminuyendo la contención para cada ranura.

A medida que se asignan más estaciones a la misma ranura (grupo) aumenta la probabilidad de colisiones pero disminuye la longitud del escaneo del mapa de bits. Se necesita una forma de asignar

dinámicamente las estaciones a las ranuras, con muchas estaciones por ranura cuando la carga es baja (1 sola ranura = ALOHA puro) y pocas cuando es alta.

- **Protocolos de recorrido de árbol adaptable:** se considera a las estaciones como hojas de un árbol binario.



**Figura 4-10.** El árbol para ocho estaciones.

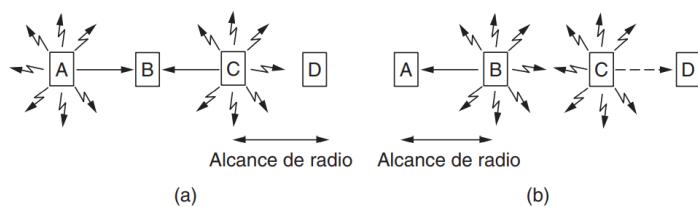
En la primer ranura de contención se permite a todas las estaciones adquirir el canal (nodo 1). Si hay colisión, continúa la búsqueda recursivamente con el hijo izquierdo (nodo 2) y luego el derecho (nodo 3) del canal. Si una ranura está inactiva o si solo una estación transmite en ella, se detiene la búsqueda de su nodo y se pasa a otro. Si en el nodo sigue detectándose colisión se baja a los inferiores. Cada ranura pertenece a un nodo.

Por ejemplo, si las estaciones G y H están listas:

- Colisión en nodo 1.
- Se intenta en nodo 2, se encuentra inactivo.
- Se sabe entonces que en el 3 habrá colisión, por lo que se pasa al 6.
- Se encuentra inactivo el nodo 6, por lo que se sabe que la colisión está en el 7.
- Como el 7 tiene dos estaciones, se intenta con la estación G.

### Protocolos de LAN inalámbrica

no pueden detectar colisiones debido a que por la atenuación de la señal puede que no se note la superposición de tramas. Por ello se usan **confirmaciones de recepción**. Como no pueden transmitir ni recibir tramas de todas las estaciones debido al rango de radio limitado, lo que importa ahora es la interferencia en el receptor, no en el emisor.



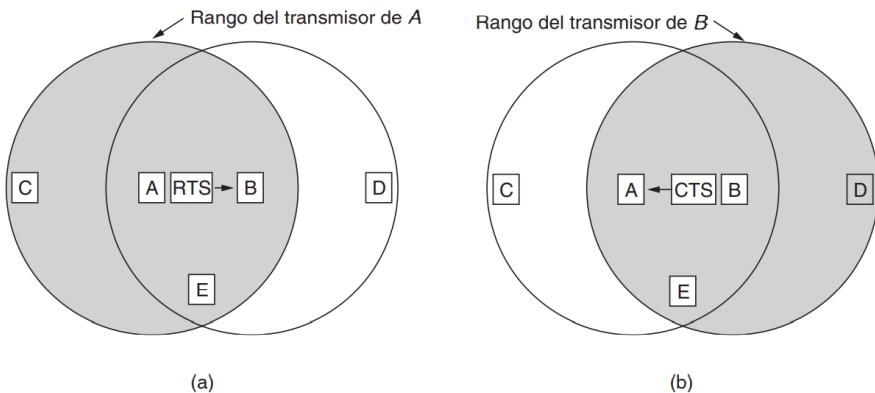
**Figura 4-11.** Una LAN inalámbrica. (a) A y C son terminales ocultas cuando transmiten a B. (b) B y C son terminales expuestas cuando transmiten a A y D.

En la figura izquierda, A y C quieren transmitir a B. Como se encuentran alejadas no se escuchan entre sí, detectando erróneamente un canal libre. Transmiten las dos y producen interferencia en C. Este problema se denomina **terminal oculta**.

En la figura derecha, *B* desea transmitir a *A* y *C* a *D*. La estación *C* detecta que *B* está transmitiendo, por lo que no transmite, sin embargo *B* no interferiría con *C* porque transmite a una estación diferente. Este problema se denomina **terminal expuesta**.

Un protocolo que aborda estos problemas es el protocolo MACA.

- **MACA (acceso múltiple con detección de colisiones):** el emisor estimula al receptor a enviar una trama corta, así las estaciones cercanas la detectan y ellas mismas evitan transmitir durante la siguiente trama de datos. Esto reemplaza a la detección de portadora.



**Figura 4-12.** El protocolo MACA. (a) *A* envía un RTS a *B*. (b) *B* responde con un CTS a *A*.

En la figura la estación *A* envía una trama **RTS** (solicitud de envío) a *B*. Esta trama de RTS de 30 bytes contiene la longitud de la trama de datos que se enviará después. La estación *B* contesta con una trama **CTS** (libre para envío) que contiene la longitud de datos copiada de la RTS. Al recibir la CTS, *A* comienza a transmitir.

Cualquier estación que escuche la RTS debe permanecer en silencio un tiempo suficiente para que el CTS se transmita. Cualquier estación que escuche el CTS está cerca de *B* por lo que debe permanecer en silencio durante la siguiente transmisión de datos, cuya longitud puede examinar del mismo CTS.

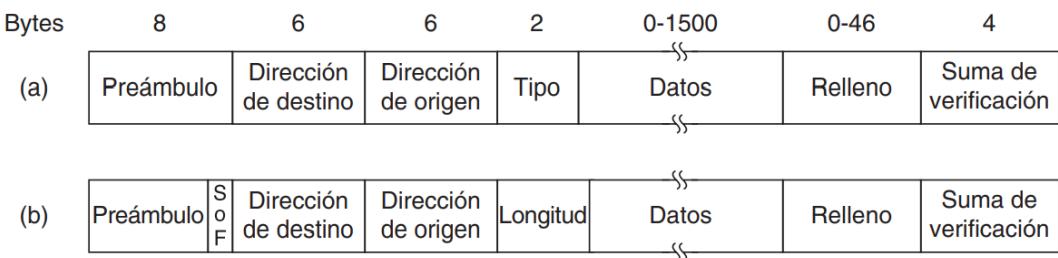
La estación *C* escucha el RTS de *A* pero no el CTS de *B*, por lo que mientras que no interfiera con el CTS es libre de transmitir mientras se envía la trama de datos (arregla terminal expuesta). La estación *D* no escucha el RTS de *A*, pero si el CTS de *B*, por lo que permanece en silencio durante la transmisión de la trama de datos (arregla terminal oculta).

Todavía pueden ocurrir colisiones en la transmisión de las tramas RTS. Al no obtener la trama de respuesta CTS, la estación que envió el RTS espera un tiempo aleatorio y vuelve a intentar (CSMA no persistente con el paquete RTS).

## Ethernet

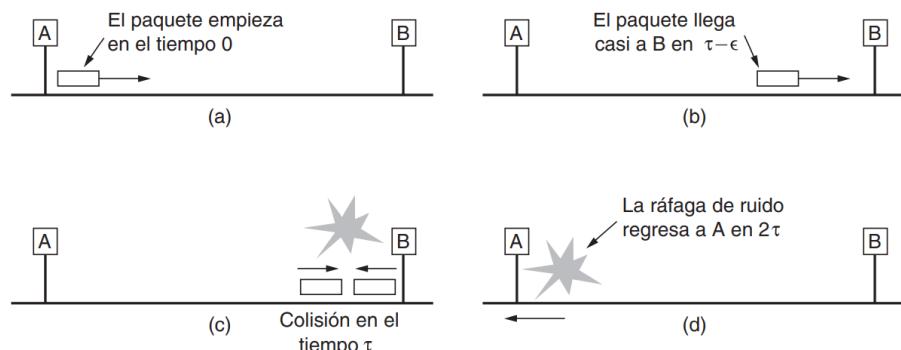
Red de difusión basada en bus con control descentralizado. Existen 2 tipos de Ethernet, la clásica que resuelve el acceso múltiple con la técnicas vistas y opera a tasa de transmisión de 3 a 10 Mbps, y la conmutada que usa switches para conectar distintas computadoras y opera a 100, 1000 y 10000 Mbps (Fast Ethernet, Gigabit Ethernet y 10 Gigabit Ethernet).

## PROTOCOLOS DE SUBCAPA MAC DE ETHERNET CLÁSICA



**Figura 4-14.** Formatos de trama. (a) Ethernet (DIX). (b) IEEE 802.3.

- **Preámbulo:** patrón de bits 10101010 excepto para la última trama (delimitador) que posee el ultimo bit en 1 (10101011). La codificación Manchester de este patrón permite que se sincronice el reloj del receptor con el del emisor.
- **Dirección de destino y origen:** las direcciones ordinarias comienzan con 0 y las de grupo con 1 (multidifusión). La que consta de todos 1 es reservada para broadcasting (difusión). El fabricante asigna los últimos 3 bytes de la dirección y programa la dirección completa en la NIC antes de venderla.
- **Tipo o longitud (trama Ethernet o IEEE 802.3):** cómo es posible utilizar múltiples protocolos de capa de red al mismo tiempo en la misma máquina, tipo indica a qué proceso darle la trama. El IEEE 802.3 decidió que este campo transporte la longitud de trama, ya que para obtenerla había que ver dentro de los datos, violando el uso de capa. Para resolverlo se agregó otro encabezado para el protocolo LLC que utiliza 8 bytes para transportar los 2 bytes de información del tipo de protocolo. IEEE establece que cualquier número menor o igual a 0x600 (1536) es valor de longitud, cualquier valor mayor es tipo.
- **Datos:** como la trama debe tener al menos 64 bytes, en caso de no llegar a esta longitud, se usa el campo de relleno. La longitud mínima es necesaria para:
  - Facilitar la distinción entre tramas validas de inservibles.
  - Evitar que una estación complete una transmisión antes de que el primer bit llegue al extremo más alejado del cable, donde podría colisionar con otra trama. Si una estación comienza a transmitir luego de que otra ya lo hizo detecta la colisión y genera una ráfaga de ruido para avisar. Si la trama del primer emisor es corta, puede que su transmisión termine antes de que le llegue la ráfaga de ruido indicando colisión, por lo que incorrectamente supondrá que la transmisión tuvo éxito. Para evitarlo, todas las tramas deben tardar más de  $2\tau$  en enviarse (siendo  $\tau$  el tiempo que tarda en llegar al extremo más alejado), así la transmisión sigue en proceso cuando llegue la ráfaga de ruido.



**Figura 4-15.** La detección de una colisión puede tardar hasta  $2\tau$ .

- **Checksum:** CRC de 32 bits que solo realiza detección de errores.

### CSMA/CD con retroceso exponencial binario

Ethernet clásica usa el algoritmo CSMA/CD persistente -1. Tras una colisión el tiempo se divide en ranuras discretas de longitud  $2\tau$ . Luego de la primera colisión cada estación espera 0 o 1 tiempos de ranura, al azar, antes de volver a intentar. Si más de una estación elige el mismo número, habrá otra colisión, luego de la cual se elegirá un numero entre 0 y 3. Luego de  $i$  colisiones se elige un numero entre 0 y  $2^i - 1$ . Al llegar a 10 colisiones el intervalo se congela en 1023 ranuras. Luego de 16 colisiones se informa a la computadora que falló.

Este algoritmo asegura un pequeño retardo cuando solo unas cuantas estaciones colisionan pero asegura que la colisión se resuelva en un intervalo razonable cuando muchas colisionan. Ni CSMA/CD ni Ethernet proveen confirmaciones de recepción.

## DESEMPEÑO DE ETHERNET

Con  $k$  estaciones listas, si cada estación transmite durante una ranura de contención con una probabilidad  $p$ , la probabilidad  $A$  de que una estación adquiera el canal durante esa ranura es:

$$A = k \cdot p \cdot (1 - p)^{k-1}$$

Si la trama promedio tarda  $P$  segundos en transmitirse:

$$\text{Eficiencia canal} = \frac{P}{P + \frac{2\tau}{A}}$$

Donde  $\frac{2\tau}{A}$  es el intervalo promedio de contención  $\omega$ .

En términos de la longitud de trama  $F$ , en ancho de banda  $B$ , la longitud del cable  $L$  y la velocidad de propagación de la señal  $c$  para  $e$  ranuras de contención por trama, con  $P = \frac{F}{B}$  es:

$$\text{Eficiencia canal} = \frac{1}{1 + \frac{2BLe}{cF}}$$

Un aumento en el ancho de banda o en la distancia de la red reduce la eficiencia para una trama dada. La eficiencia es menor cuanto más chica es la trama, ya que es relativamente mayor el encabezado y terminador.

## ETHERNET CONMUTADA

Utiliza switches que envían las tramas a los puertos para los cuales están destinadas. El switch recibe una trama Ethernet y verifica las direcciones Ethernet para ver cuál es el puerto de destino de la trama. Cada puerto es su propio dominio de colisión independiente. Las colisiones son imposibles y no se necesita CSMA/CD.

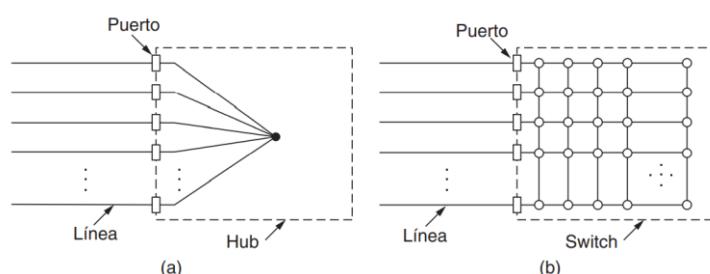


Figura 4-17. (a) Hub. (b) Switch.

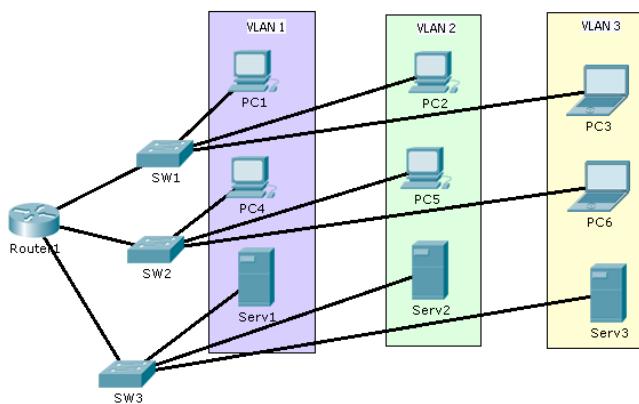
En comparación con un HUB, el switch mejora el desempeño de la red de dos maneras:

- Como no hay colisiones, la capacidad se usa con más eficiencia.
- Puede enviar varias tramas al mismo tiempo por distintas estaciones. Debe tener un buffer ya que pueden llegar dos tramas para la misma estación al mismo tiempo, así puede poner temporalmente una trama en cola.

## VLAN – LAN VIRTUAL

Método de creación de redes lógicas e independientes dentro de una misma red física. Es una red de hosts que se comportan como si estuviesen conectados al mismo switch.

- Varias VLANs pueden coexistir en un único switch físico.
- Ayudan a la administración de la red.
- Las VLANs se configuran mediante software, por lo que son flexibles.



## CAPA DE RED

Se encarga de llevar los paquetes todo el camino, de origen a destino. Puede que se necesiten saltos por enruteadores intermedios. Es la capa más baja que maneja la transmisión de extremo a extremo.

Debe conocer la topología de red para elegir rutas apropiadas y solucionar los problemas cuando origen y destino se encuentran en redes diferentes.

IP es el protocolo de capa de red de internet. Los paquetes se denominan **PDU**.

## Aspectos de diseño de la capa de red

Problemas a los que deben enfrentarse los diseñadores de la capa de red.

## CONMUTACIÓN DE PAQUETES DE ALMACENAMIENTO Y REENVÍO

Los componentes principales de la red son el equipo del ISP y el de los clientes.

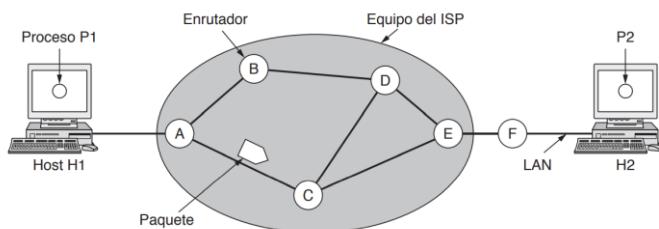


Figura 5-1. El entorno de los protocolos de la capa de red.

H1 está conectado al enrutador A del ISP, H2 está en una LAN con un enrutador F del cliente. Los routers locales de los clientes se consideran parte del ISP porque ejecutan los mismos algoritmos.

Un host que desea enviar un paquete lo transmite al router más cercano, se almacena en él hasta llegar por completo y se comprueba el checksum. Luego reenvía al siguiente enrutador y así hasta el host destino. Este mecanismo se denomina **conmutación de almacenamiento y reenvío**.

## SERVICIOS PROPORCIONADOS A LA CAPA DE TRANSPORTE

Deben diseñarse los servicios con los siguientes objetivos en mente:

- Servicios independientes de la tecnología del router.
- Capa de transporte aislada de la cantidad, tipo y topología de los routers presentes.
- Direcciones de red disponibles para capa de transporte con plan de numeración uniforme.

Dadas estas metas, los diseñadores tienen mucha libertad para detallar los servicios, lo que degenera en una discusión que se centra en determinar si la capa de red debe dar un servicio orientado o no a conexión. Un bando (comunidad **internet**) sostiene que la tarea de los routers solo es mover paquetes, la red es de naturaleza no confiable y los hosts deben realizar los controles de error y flujo, concluyendo que el servicio debe ser **sin conexión**. El otro bando (compañías **telefónicas**) sostiene que la red debe dar un servicio confiable, **orientado a conexión**, donde la calidad de servicio es predominante.

Esta controversia sigue vigente. Las primeras redes de datos eran orientadas a conexión. Sin embargo, desde ARPANET y los inicios de internet, la popularidad de las capas de red sin conexión aumentó. Igualmente internet desarrolla características orientadas a conexión, como en MPLS y redes VLAN.

## IMPLEMENTACIÓN DEL SERVICIO SIN CONEXIÓN

Los paquetes se transmiten por separado en la red y se enrutan de manera independiente. No necesita configuración previa. Los paquetes se conocen como datagramas y la red como red de datagramas.

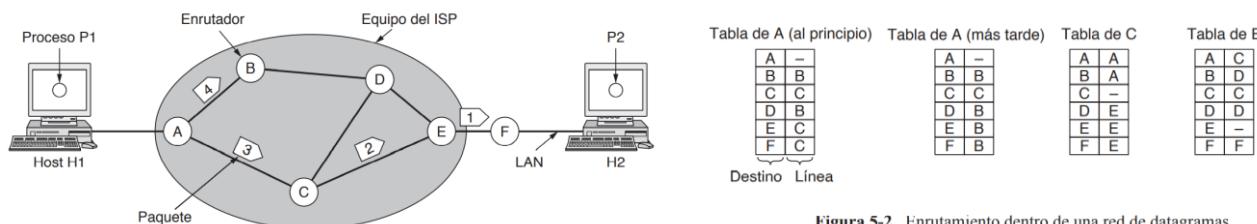


Figura 5-2. Enrutamiento dentro de una red de datagramas.

Si el proceso P1 tiene un mensaje para P2, este lo entrega a la capa de transporte indicando que lo envíe al proceso P2 en H2. El código de capa de transporte se ejecuta en H1, agregando un encabezado y entregándolo a la capa de red.

Cada router tiene una tabla interna que indica donde enviar paquetes para cada uno de los destinos. Las entradas de la tabla son un par compuesto por un destino y la línea de salida para dicho destino.

En el router A, los paquetes 1, 2 y 3 se almacenan hasta llegar por completo y comprobar su checksum. Luego cada uno se envía según la tabla de A, en este caso por el enlace de salida a C.

En caso de congestión, A puede decidir enviar por una ruta diferente actualizando su tabla. El algoritmo que maneja las tablas y realiza estas decisiones es el **algoritmo de enrutamiento**.

## IMPLEMENTACIÓN DEL SERVICIO ORIENTADO A CONEXIÓN

Utiliza circuitos virtuales (CV) para evitar la necesidad de elegir una nueva ruta para cada paquete enviado. Al establecer una conexión se elige una ruta desde la máquina origen a destino como parte de la configuración de conexión y se almacena en tablas dentro de enruteadores. Cada paquete lleva un identificador que indica a cuál circuito virtual pertenece.

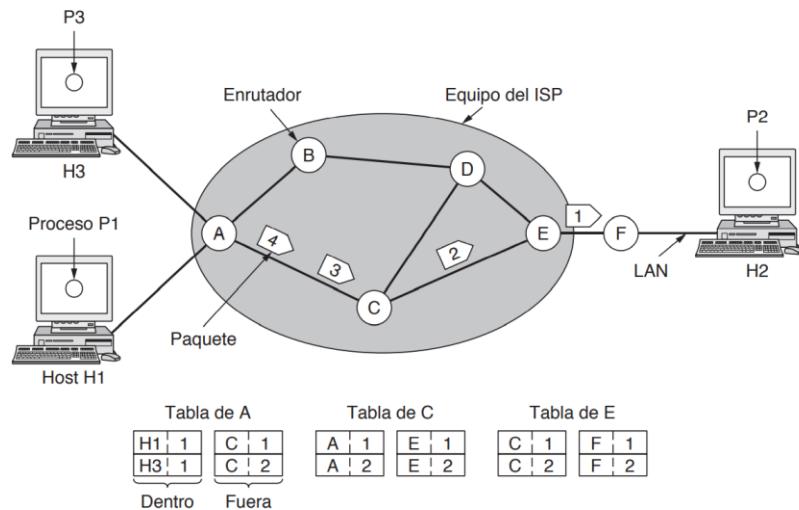


Figura 5-3. Enrutamiento dentro de una red de circuitos virtuales.

En la figura anterior *H1* establece la conexión 1 con *H2*. Esta conexión es la primer entrada en cada una de las tablas de enrutamiento. La primera línea de la tabla *A* indica que si un paquete con identificador de conexión 1 llega desde *H1*, se enviará a *C* con el identificador 1.

Si *H3* desea establecer conexión con *H2*, elige también el identificador de conexión 1. Aunque luego *A* puede saber cuáles paquetes vienen de *H1* y cuáles de *H3*, *C* no. Por ello, *A* asigna un identificador de conexión diferente al tráfico de salida para la segunda conexión.

A este proceso se lo denomina **conmutación multiprotocolo mediante etiquetas** (MPLS). Se utiliza dentro de las redes del ISP en internet.

## COMPARACION REDES DE CV Y DE DATAGRAMAS

El uso de CV requiere una configuración previa que necesita tiempo y recursos. En la red de datagramas no, pero se requiere un procedimiento más complejo para localizar la entrada correspondiente al destino.

Las direcciones de destino en redes de datagramas son más largas que los números de conexión en las redes de CV. Si los paquetes son cortos, la dirección destino completa representa una sobrecarga.

Los CV tienen ventajas en garantizar la calidad de servicio y evitar congestiones ya que los recursos se reservan al establecer la conexión.

Los CV tienen un problema de vulnerabilidad, si falla un router todos los CV por él deberán abortarse.

Asunto	Red de datagramas	Red de CV
Configuración de circuito	No requerida	Requerida
Direccionamiento	Cada paquete con dirección origen y destino completas	Cada paquete con número de CV
Información de estado	Los routers no tienen información de estado de las conexiones	Cada CV requiere espacio de tabla del router por cada conexión

Enrutamiento	Independiente	Ruta seleccionada al establecer el CV
Efecto de fallas en router	Ninguno, excepto para paquetes perdidos durante la falla	Terminan todos los CV que pasan por el router defectuoso
Calidad de servicio y control de congestión	Difícil	Fácil si se pueden asignar suficientes recursos por adelantado.

## Algoritmos de enruteamiento

Parte del software de capa de red que decide por cuál línea de salida se transmitirá un paquete. Es útil distinguir dos conceptos:

- **Enrutamiento:** proceso de tomar la decisión de cuales rutas utilizar. El algoritmo de enruteamiento llena y actualiza las tablas.
- **Reenvío:** acción realizada al llegar un paquete. Maneja cada paquete conforme llega y luego busca en las tablas la salida.

Además de sencillez y exactitud, todo algoritmo de enruteamiento debe poseer:

- **Robustez:** ante fallas, ser capaz de manejar cambios de topología y tráficos sin abortar todas las tareas de todos los hosts.
- **Estabilidad:** converger con rapidez a un conjunto de rutas fijo y conservarlo.
- **Equidad y eficiencia:** metas contradictorias, por lo que se requiere una solución de compromiso.

Existen 2 clases principales de algoritmos:

1. **No adaptativo:** la decisión de qué ruta usar se calcula por adelantado, fuera de línea y se descarga en los routers al arrancar la red. Se denomina **enrutamiento estático**. No responde a fallas, por lo que es útil cuando la elección de enruteamiento es clara.
2. **Adaptativo:** cambian las decisiones de enruteamiento ante cambios de topología y tráfico o cada cierto periodo de tiempo. Se denomina **enrutamiento dinámico**. La métrica usada para la optimización puede ser la distancia, el número de saltos o el tiempo estimado de tránsito.

## PRINCIPIO DE OPTIMIZACIÓN

Si el enruteador *J* se encuentra en la ruta óptima de *I* a *K*, entonces la ruta optima de *J* a *K* es la misma ruta. El grupo de rutas óptimas de todos los orígenes a un solo destino forman un árbol con raíz en el destino, conocido como **árbol sumidero**. El objetivo de los algoritmos de enruteamiento es descubrir y usar estos árboles. Proporciona además el punto de referencia para medir otros algoritmos.

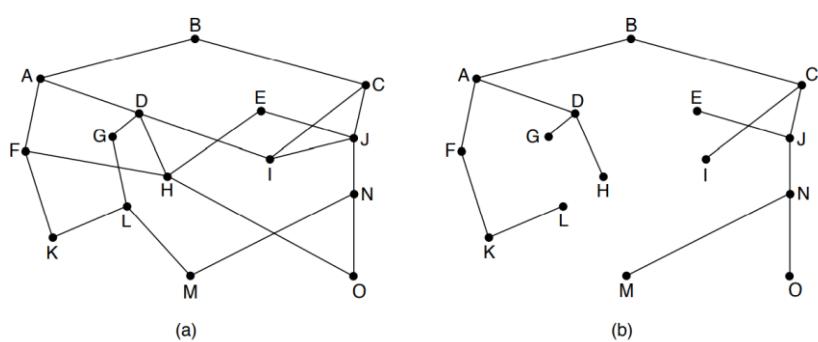


Figura 5-6. (a) Una red. (b) Un árbol sumidero para el enruteador *B*.

## ALGORITMO DE LA RUTA MÁS CORTA

Se construye un grafo de la red donde cada nodo es un router y cada arco un enlace. Cada nodo se etiqueta con su distancia desde el nodo origen a través de la mejor ruta conocida. Al principio las etiquetas son tentativas, una vez que se descubre que una etiqueta representa la ruta más corta, se vuelve permanente.

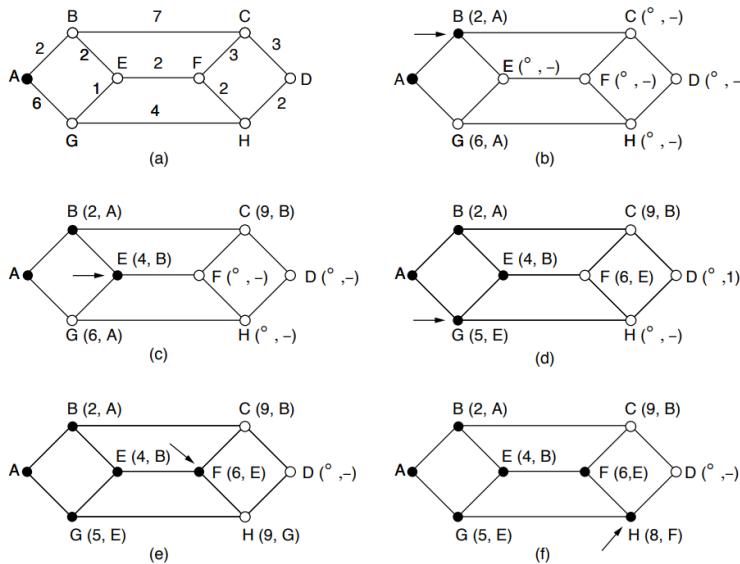


Figura 5-7. Los primeros seis pasos utilizados para calcular la ruta más corta de *A* a *D*. Las flechas indican el nodo de trabajo.

Por ejemplo, para encontrar la ruta más corta de *A* a *D*:

1. Marcados *A* como permanente.
2. Examinamos los nodos adyacentes y los reetiquetamos con la distancia a *A*.
3. Examinamos los nodos etiquetados y hacemos permanente el de la etiqueta más pequeña. Este se convierte en el nuevo nodo de trabajo.
4. Comenzamos por este nuevo nodo (*B* en la figura) y examinamos los nodos adyacentes a él.

## INUNDACIÓN

Cuando se implementa un algoritmo de enrutamiento, cada enrutador debe tomar decisiones con base en el conocimiento local. Una técnica local simple es la **inundación**, en la que cada paquete entrante se envía a todas las líneas de salida, excepto a la línea por la cual llegó.

Esto genera grandes cantidades de paquetes duplicados a menos que se tomen medidas para limitar el proceso. Una medida es integrar un contador de saltos en el encabezado de cada paquete, el cual disminuya con cada salto, y se descarte el paquete cuyo contador llegue a cero. Lo ideal sería inicializar este contador con la cantidad de saltos entre origen y destino, pero si el emisor desconoce este valor puede iniciarse con el peor caso, la longitud total de la red.

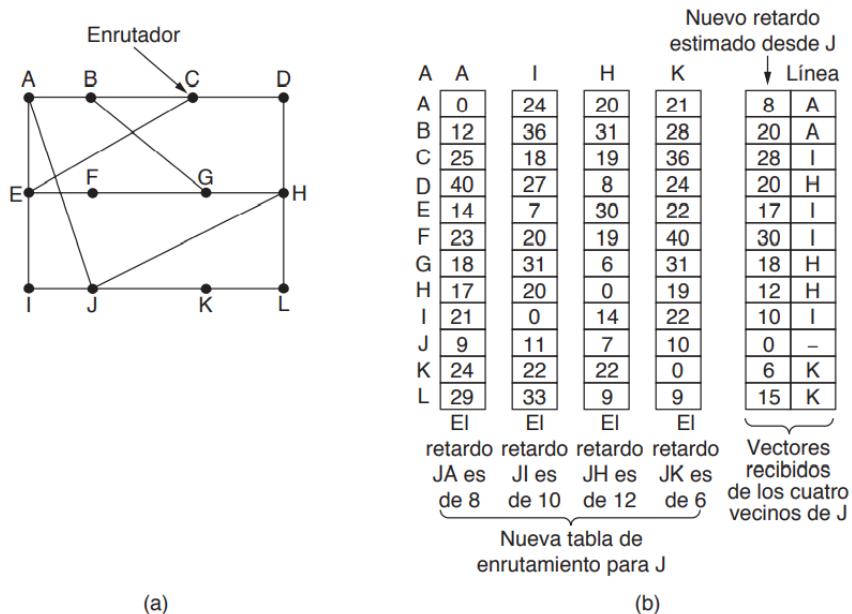
La inundación no es práctica para enviar los paquetes, pero tiene algunos usos y características importantes:

- Asegura que un paquete se entregue a todos los nodos de la red. Esto podría ser un desperdicio si solo hay un destino deseado, pero es efectivo para difusión.
- Es muy robusta, incluso si gran cantidad de routers dejan de funcionar encontrará una ruta, si es que existe, para transmitir un paquete a su destino. Además requiere poca configuración.
- Siempre selecciona la ruta más corta, ya que selecciona todas las rutas posibles en paralelo.
- Puede usarse como métrica de comparación.

## ENRUTAMIENTO POR VECTOR DISTANCIA

Cada router mantiene una tabla (vector) con la mejor distancia conocida a cada destino y el enlace para llegar allí. Cada entrada tiene la línea preferida de salida para ese destino y una estimación de tiempo o distancia al mismo. El router conoce la distancia a cada uno de sus vecinos.

Es un algoritmo distribuido ya que la información está distribuida en los routers. Este se corre en capa 7 de aplicación como un programa que modifica la tabla de ruta usada en la capa 3.



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

Por ejemplo,  $J$  calcula su nueva ruta a  $G$  de la siguiente manera:

- $J$  sabe que puede llegar a  $A$  en 8ms y  $A$  puede llegar a  $G$  en 18ms.
  - $J$  sabe ahora que si reenvía por  $A$ , cuenta con un retardo total a  $G$  de 26ms (18+8).
  - Del mismo modo calcula el retardo a través de  $I$  ( $31+10=41$ ms),  $H$  ( $6+12=18$ ms) y  $K$  ( $31+6=37$ ms).
  - El mejor de estos valores es a través de  $H$ , por lo que escribe una entrada en su tabla de enrutamiento para indicar que el retardo a  $G$  es de 18ms y se utilizará la ruta a través de  $H$ .

## **El problema del conteo al infinito**

Este algoritmo tiene una grave debilidad en la práctica ya que si bien reacciona rápido a buenas noticias, reacciona muy lento a malas.

Si en una red un router está desactivado y los otros routers de la red lo saben, al activarse, los routers lo sabrán de él gracias al intercambio de vectores. Esta noticia se difunde a razón de un salto por intercambio.

Si todos los routers de una red están activos y de pronto uno se desactiva, los demás nodos al no poder llegar a este nodo desconectado intentarán hacerlo por la ruta de otro nodo. No importa si la nueva ruta los incluye porque ellos no lo saben. Intentan por la ruta de todos los nodos adyacentes y los adyacentes de los adyacentes, tomando un tiempo muy alto en reconocer la desconexión del nodo destino. La razón por la que las malas noticias viajan con lentitud es que hay un enrutador que tenga en algún momento un valor mayor en más de una unidad que el mínimo de todos sus vecinos. Gradualmente todos los routers elevan su cuenta hacia el infinito, pero el número de intercambios requeridos depende del valor numérico usado para el infinito. Por ello es prudente hacer que el infinito sea igual a la ruta más larga + 1.

## ENRUTAMIENTO POR ESTADO DEL ENLACE

Reemplazó al enrutamiento por vector de distancia ya que tardaba demasiado en converger ante cambios de topología en la red.

Cada enrutador debe:

- Descubrir a sus vecinos y conocer sus direcciones de red.
- Establecer la métrica de distancia o de costo para cada uno.
- Construir un paquete que indique lo que acaba de aprender.
- Enviar este paquete a todos los enrutadores y recibir el de ellos.
- Calcular la ruta más corta a todos los demás enrutadores.

Se distribuye la topología completa a todos los routers. Despues se ejecuta Dijkstra en cada router para encontrar la ruta más corta. Los 5 pasos en detalle son:

1. **Aprender sobre los vecinos:** cuando un router se pone en funcionamiento envía un paquete especial HELLO en cada línea punto a punto. El router del otro extremo responde con su nombre.
2. **Establecimiento de los costos de los enlaces:** una elección común es hacer que el costo sea inversamente proporcional al ancho de banda del enlace. Si la red es muy dispersa puede considerarse al retardo en el costo. Para determinar el retardo se envía un paquete ECO a través de la línea que debe devolver el otro extremo, el tiempo que tarda en volver dividido 2 es el retardo.
3. **Construcción de los paquetes de estado de enlace:** cada router debe construir un paquete con todos los datos, identidad emisor, número de secuencia, edad y lista de vecinos con costos para cada uno.

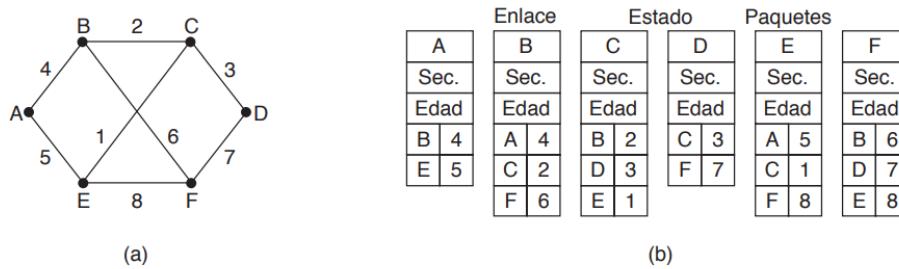


Figura 5-12. (a) Una red. (b) Los paquetes de estado del enlace para esta red.

Estos pueden construirse periódicamente, cuando ocurra un evento significativo o cuando las propiedades cambien considerablemente.

4. **Distribución de los paquetes de estado de enlace:** todos los routers deben recibir todos los paquetes de estado de enlace con rapidez y confiabilidad. Se utiliza **inundación** para distribuirlos. Para controlar la inundación cada paquete contiene un número de secuencia que se incrementa con cada nuevo paquete enviado. Cuando llega un nuevo paquete se verifica y compara con la lista de paquetes vistos. Si es nuevo se reenvía por todas las líneas excepto por la que llegó, si es duplicado se descarta. Si llega un paquete con numero de secuencia menor que el mayor visto hasta el momento, se rechaza como obsoleto. Este algoritmo tiene problemas cuando:

- Los números de secuencia vuelven a comenzar, la solución es usar un número de 32 bits.
- Falla un router (se pierde el registro de su número de secuencia).
- Se corrompe un numero de secuencia (se rechazan paquetes no duplicados).

La solución es incluir la edad de cada paquete después del número de secuencia y disminuirla una vez cada segundo. Si la edad llega a cero se descarta la información de ese router.

En la siguiente figura cada fila corresponde a un paquete de estado de enlace recién llegado que aún no se procesa por completo. La tabla registra dónde se originó el paquete, su número de secuencia y su edad. Hay banderas de envío (indican por donde debe enviarse el paquete) y confirmación de recepción (indican donde hay que confirmar recepción).

Origen	Sec.	Edad	Banderas de envío			Banderas de ACK			Datos
			A	C	F	A	C	F	
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Figura 5-13. El búfer de paquetes para el enrutador B de la figura 5-12(a).

Vemos que el paquete de A se debe enviar a C y F y confirmar recepción a A. El tercer paquete proveniente de E llega dos veces (EAB y EFB) por lo que debe enviarse a C y confirmar a A y F..

5. **Cálculo de las nuevas rutas:** una vez que el router acumuló paquetes de estado de enlace puede construir el grafo de la red. Puede ejecutar localmente Dijkstra. Los resultados le indican al router qué enlace usar para llegar a cada destino. Esta información se instala en las tablas de enrutamiento y se puede reanudar la operación normal.

## Ruteo estático

Método manual en el cual se indica explícitamente en cada equipo las redes que puede alcanzar y por qué camino hacerlo.

Ventajas:

- Simple de configurar.
- No supone sobrecarga adicional sobre los routers.

Desventajas:

- Trabajo engorroso.
- Aumenta la probabilidad de cometer errores.
- Si falla un enlace deben modificarse las rutas manualmente.

Se configura mediante los siguientes comandos:

```
router(config)# ip route [IP_Destino] [máscara] [IP_siguiente_salto o interfaz_salida]
```

Interfaz de salida se refiere a la interfaz del router local conectada a la red externa.

Para eliminar una ruta estática:

```
router(config)# no ip route [IP_Destino] [máscara] [IP_sig_salto o interfaz_salida]
```

Ruteo estático por defecto:

```
router(config)# ip route 0.0.0.0 0.0.0.0 [IP_siguiente_salto o interfaz_salida]
```

Envía a la interfaz de salida o IP de siguiente salto todos los paquetes cuya dirección de origen no están en la tabla de enrutamiento.

## Ruteo dinámico

Se configura mediante los siguientes pasos:

1. Indicar rango de direcciones IP

```
router(config)# ip dhcp excluded-address [IP Inicial] [IP Final]
```

2. Dar un nombre al conjunto de direcciones que serán asignadas.

```
router(config)# ip dhcp pool [Nombre]
```

3. Definir parámetros

```
router(config)# network [IP_red] [Máscara] // IP_red es la terminada en .0  
router(config)# default-router [IP Gateway]// Generalmente terminada en .1  
router(config)# dns-server [IP_server_DNS] // Generalmente 5.5.5.5 o 6.6.6.6
```

4. Ver tabla de asignación:

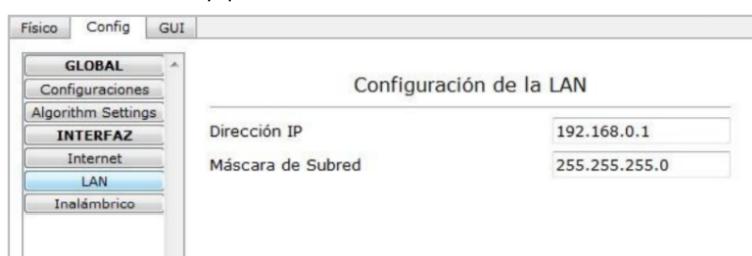
```
router# show ip dhcp binding
```

IP address	Client-ID/Hardware address	Lease expiration	Type
192.168.0.10	00C.CF55.8D7E	--	Automatic

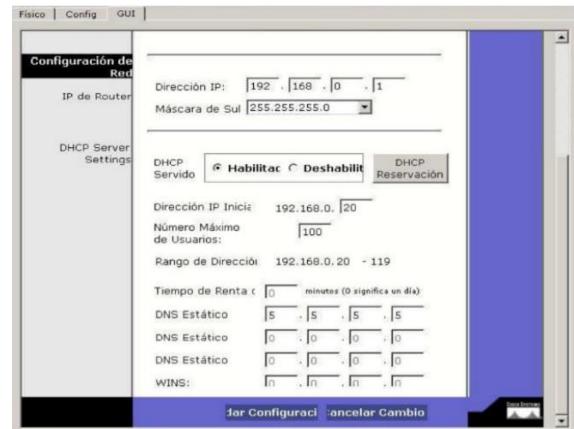
- **Configuración de puerto internet:** se configura el puerto internet del modem WiFi para que el tipo de conexión sea por el servidor DHCP.



- **Configuración del puerto LAN:** se configura el puerto LAN con una dirección IP. La IP de este puerto luego envía como Gateway predeterminado el servidor DHCP del router.



- **Configuración del servidor DHCP:**
  - Habilitar servidor DHCP.
  - Guardar configuración.
  - Verificar IP y máscara.
  - Colocar la primer IP a ser asignada.
  - Colocar el número máximo de usuarios.
  - Colocar el servidor DNS.
  - Guardar configuración.



## Interconexión de redes

Al conectar 2 o más redes debe lidiarse con problemas de heterogeneidad y escala.

### CÓMO DIFIEREN LAS REDES

Las redes pueden diferir en los siguientes aspectos:

- **Servicio ofrecido:** puede ser orientado o no a conexión.
- **Direccionamiento:** tamaños, plano o jerárquico.
- **Difusión:** puede o no estar presente.
- **Tamaño de paquetes:** cada red tiene su valor máximo (MTU).
- **Calidad de servicio:** puede o no estar presente.
- **Ordenamiento:** entrega ordenada o desordenada.
- **Confiabilidad:** distintos niveles de pérdida.
- **Seguridad:** reglas de privacidad, cifrado, etc.
- **Parámetros:** tiempos de expiración, especificaciones de flujo, etc.
- **Contabilidad:** tiempo de conexión, paquete, byte o ninguna.

### CÓMO PUEDEN CONECTARSE LAS REDES

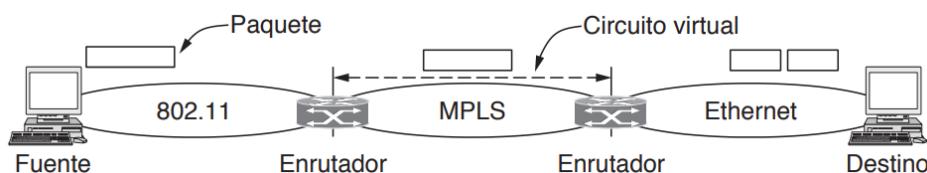
Existen 2 opciones básicas:

1. Construir dispositivos que traduzcan los paquetes de cada tipo de red.
2. Agregar una capa de indirección común por encima de las distintas redes.

Se optó por la capa común para ocultar diferencias. Esta capa se separó en TCP/IP, pero después se adoptó IP ya que proporciona un formato de paquete universal conocido por todos los routers.

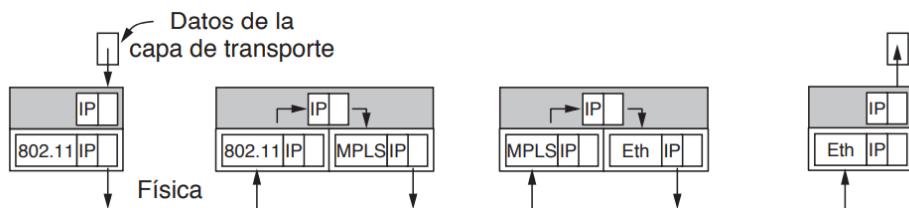
Los dispositivos para la interconexión que operan la capa de red son los routers.

En la siguiente figura se ve el procedimiento de interconexión de redes mediante la capa común IP.



La máquina en la red 802.11 desea enviar un paquete a la máquina en la red Ethernet. Este paquete transporta una dirección de capa de red. La red 802.11 provee un servicio sin conexión, mientras que MPLS es orientado a conexión por lo que se necesita establecer un circuito virtual para atravesar esta

red. Luego el paquete sigue y llega a Ethernet, donde puede que sea demasiado grande (802.11 puede trabajar con tramas más grandes que Ethernet), en cuyo caso se fragmenta y al llegar a destino se desfragmenta.



Desde el punto de vista de los protocolos, la capa de red recibe los datos de la capa de transporte y genera un encabezado con la capa de red común (IP). El paquete se encapsula en una trama 802.11 con destino al primer router. El paquete se extrae del campo de datos y se descarta el encabezado de la trama 802.11. El router examina la IP en el paquete y busca esta dirección en su tabla de enrutamiento. Con base en esta dirección decide enviar el paquete al segundo router. Al llegar al segundo router sucede lo mismo con el encabezado MPLS.

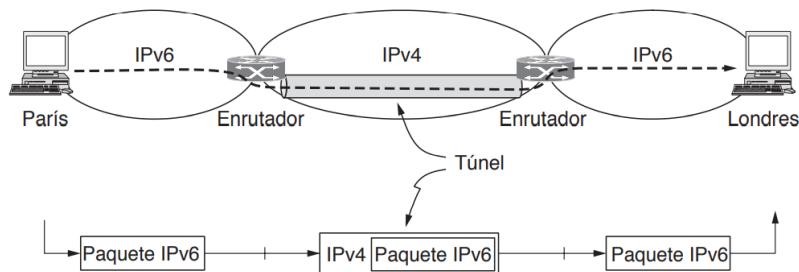
Podemos decir que el router extrae de la trama el paquete y decide según la dirección de red dónde enviarlo. También se encarga de conectar distintas redes en la capa de red, por lo que deben entender sus protocolos.

Un router que puede manejar múltiples protocolos de red se denomina **enrutador multiprotocolo**.

## TUNELIZACIÓN

Cuando origen y destino se encuentran en el mismo tipo de red, pero entre ellos hay una red diferente, puede manejarse la interconexión de redes distintas para distintos protocolos de red.

La solución es la técnica de **tunelización**. En ella un enrutador multiprotocolo encapsula un paquete de IPv6 con un encabezado IPv4, es decir, coloca un paquete IPv6 dentro del paquete IPv4.



**Figura 5-40.** Tunelización de un paquete de París a Londres.

Se usa con frecuencia para conectar hosts y redes aisladas mediante el uso de otras redes. La red resultante se denomina **red superpuesta** (overlay). La desventaja es que no se puede llegar a hosts de la red que se tuneliza, es decir, los paquetes no pueden escapar a mitad del túnel.

## ENRUTAMIENTO ENTRE REDES

Las redes pueden tener distintos algoritmos de enrutamiento, por lo que podría dificultarse encontrar las rutas más cortas. Esto conduce a un algoritmo de enrutamiento de dos niveles:

- **Dentro de cada red:** protocolo intradominio o de puerta de enlace interior.
- **Entre las redes de la interred:** protocolo interdominio o de puerta de enlace exterior.

Todas las redes pueden usar diferentes protocolos intradominio pero deben usar el mismo protocolo interdominio. En internet el protocolo interdominio se denomina BGP (protocolo de puerta de enlace de frontera).

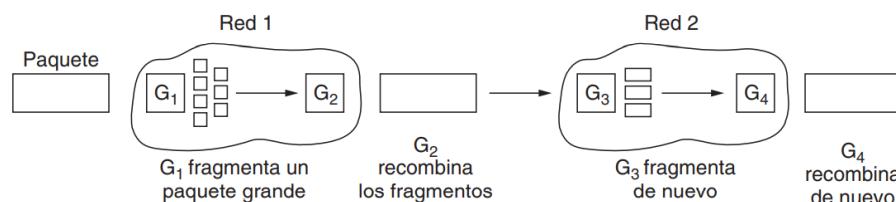
## FRAGMENTACIÓN DE PAQUETES

Cada red o enlace impone un tamaño máximo a sus paquetes. Esto tiene varias razones:

- Hardware.
- SO.
- Protocolos.
- Estándares.
- Deseo de minimizar retransmisiones por errores.
- Deseo de evitar que un paquete ocupe el canal demasiado tiempo.

Surgen problemas entonces si un paquete grande viaja a través de una red de tamaño máximo de paquete menor (**MTU**). Las soluciones son:

- Asegurar que no ocurra el problema. Sin embargo la fuente no conoce la ruta, por lo que no sabe que tan pequeños deben ser los paquetes. Incluso si la fuente conociera la MTU de la ruta, los paquetes se enrutan de manera independiente en una red sin conexión como internet, por lo que las rutas pueden cambiar de repente, lo que a su vez puede cambiar de manera inesperada el MTU de la ruta.
- Permitir que los routers dividan los paquetes en fragmentos y envíen cada uno como un paquete de red separado. La posterior fragmentación es menos sencilla y existen 2 estrategias para realizarla:
  - Fragmentación transparente: cada enrutador fragmenta en la entrada y desfragmenta en la salida. Esto trae los siguientes problemas:
    - El router necesita un campo de conteo o bit de fin de paquete.
    - Las demás rutas están restringidas que todas las piezas deben salir por el mismo router.
    - Puede que el router necesite de un buffer donde colocar los fragmentos porque no sabe si van a llegar todos o si tiene que descartarlos.



- Fragmentación no transparente: la recombinación ocurre solo en el host destino.
  - La ventaja es que los routers tienen menos trabajo. IP trabaja de esta manera, donde a cada fragmento se le asigna un número de paquete. Cuando llegan a destino usa el número de paquete y el desplazamiento de fragmento para colocar los datos en la posición correcta y la bandera de fin de paquete para determinar si el paquete está completo.
  - El problema es que si un fragmento se pierde, se pierde todo el paquete y además sobrecarga a los hosts ya que se agregan bits para control.

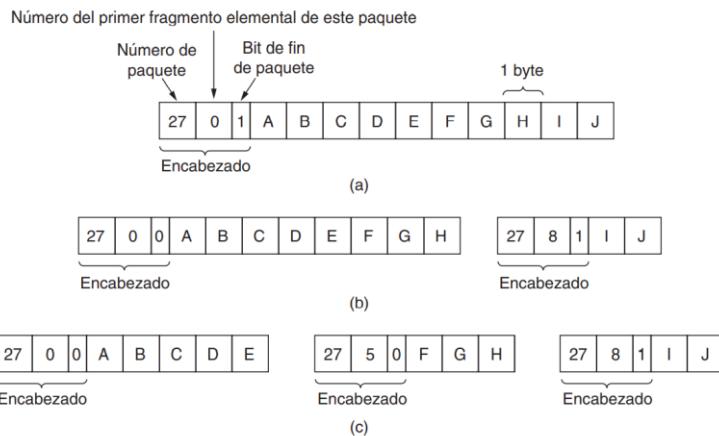
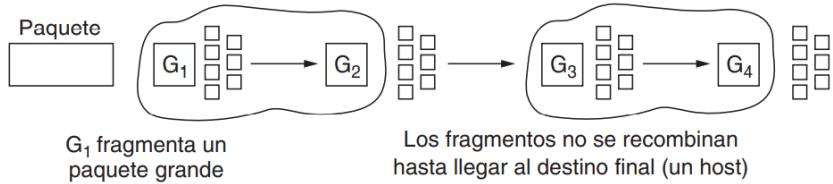


Figura 5-43. La fragmentación cuando el tamaño de datos elemental es de 1 byte. (a) El paquete original que contiene 10 bytes de datos. (b) Los fragmentos después de pasar por una red con un tamaño máximo de paquete de 8 bytes de carga útil más encabezado. (c) Fragmentos después de pasar a través de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

Los problemas generados por la fragmentación nos hacen deshacernos de ella. Se usa una estrategia llamada **descubrimiento del MTU de la ruta**. Cada paquete IP se envía con sus bits de encabezado establecidos para indicar que no puede realizarse fragmentación. Si un router recibe un paquete mayor a su MTU genera un paquete de error que envía a la fuente y descarta el paquete recibido. Cuando la fuente recibe el paquete de error fragmenta el paquete para que pueda ser manejado por el router. Si más adelante un router posee una MTU menor, se repite el proceso.

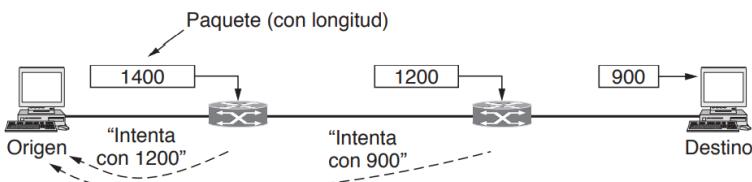


Figura 5-44. Descubrimiento de MTU de la ruta.

La ventaja es que la fuente ahora conoce la longitud del paquete que puede enviar. Igualmente la fragmentación sigue siendo necesaria entre la fuente y el destino, a menos que las capas superiores descubran la MTU de la ruta y pasen la cantidad correcta de datos al protocolo IP. Puede decirse que la fragmentación se sacó de la red y pasó a los hosts. La desventaja es que puede haber retardos iniciales.

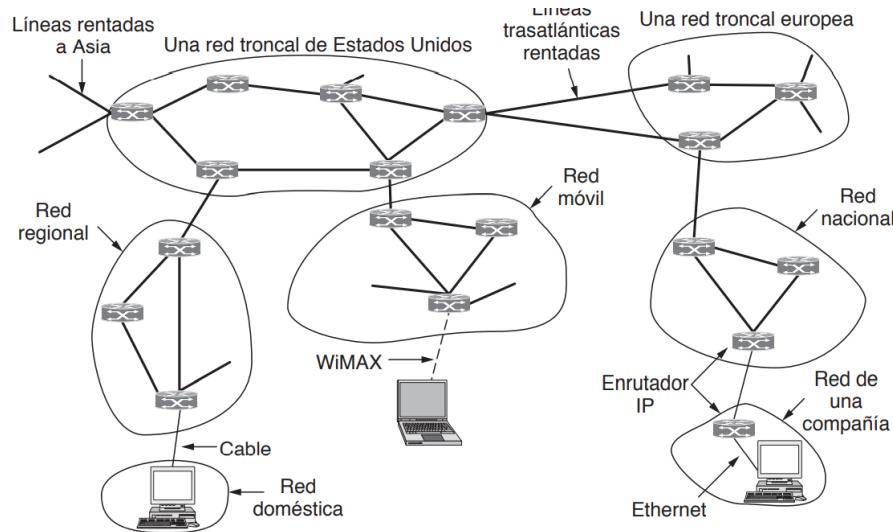
## La capa de red de internet

Los principios más importantes de su diseño son:

- Asegurar funcionamiento.
- Simplicidad.
- Opciones claras.
- Modularidad.
- Prevenir heterogeneidad.

- Evitar opciones y parámetros estáticos.
- Buscar buen diseño, no perfecto.
- Estricto al enviar y tolerante al recibir.
- Escalabilidad.
- Desempeño y costo.

En capa de red, internet puede verse como un conjunto de redes interconectadas. Las redes troncales principales son líneas de alto BW con enruteadores rápidos. Conectados a las redes troncales están los ISP. A las redes regionales se conectan más ISP. A las redes locales se conectan más redes regionales.



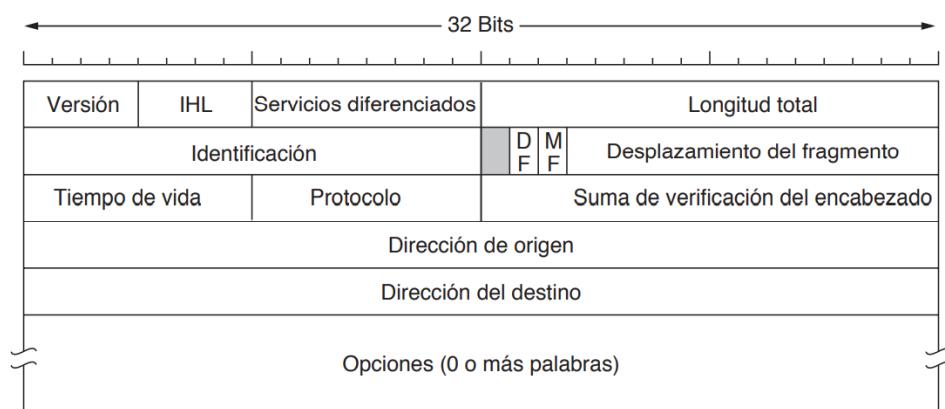
**Figura 5-45.** Internet es una colección interconectada de muchas redes.

Internet se mantiene unida por el protocolo de la capa de red IP, diseñado con el objetivo de la interconexión de redes. La comunicación en internet funciona de la siguiente manera:

- La capa de transporte toma flujos de datos y los divide para poder enviarlos como paquetes IP.
- Los routers IP reenvían cada paquete mediante internet de un router a otro hasta el destino.
- Al llegar todas las piezas a destino, la capa de red las ensambla para formar el datagrama original, el cual se entrega a la capa de transporte.

## PROTOCOLO IPV4

Consiste en 2 partes, encabezado y carga útil. El encabezado tiene una parte fija de 20 bytes y una opcional de longitud variable. Es serial, cada renglón de 32 bits se envía detrás del otro.



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

- **Versión:** IPv4 – IPv6.
- **IHL:** longitud del encabezado en palabras de 32 bits.
- **Servicios diferenciados:** distingue las clases de servicio. Son posibles varias combinaciones de confiabilidad y velocidad. Los 6 bits superiores marcan el paquete con la clase de servicio (expedito y asegurado). Los 2 inferiores transportan información de congestión.
- **Longitud total:** longitud de todo el datagrama (máximo de 65535 bytes).
- **Identificación:** para que host destino sepa a quién pertenece un paquete recién llegado.
- **DF (Don't Fragment):** si está en 1 indica que el paquete no debe fragmentarse.
- **MF (More Fragments):** si está en 1 indica que hay más fragmentos del mismo paquete. Solo el último fragmento del paquete tiene este bit en 0.
- **Desplazamiento del fragmento:** posición del fragmento dentro del paquete.
- **Tiempo de vida (TTL):** contador limitador de vida de un paquete. Cuenta los saltos, al llegar a cero se descarta y se avisa al host origen.
- **Protocolo:** indica el proceso de transporte (TCP o UDP) al cual debe entregar el paquete una vez ensamblado.
- **Suma de verificación del encabezado:** suma todas las medias palabras de 16 bits del encabezado conforme van llegando mediante aritmética de complemento a 1 y obtiene el complemento a 1 del resultado. Útil para detectar errores mientras el paquete viaja por la red. Debe recalcularse en cada salto ya que siempre cambia por lo menos el campo TTL.
- **Dirección origen y destino:** IP de fuente y destino.
- **Opciones:** recurso para que versiones subsiguientes incluyan información adicional.

Opción	Descripción
Seguridad	Indica qué tan secreto es el paquete
Enrutamiento estricto desde el origen	Ruta completa a seguir
Enrutamiento libre desde el origen	Lista de routers obligatorios
Registrar ruta	Cada router adjunta su IP
Estampa de tiempo	Cada router adjunta dirección y etiqueta de tiempo

Los pasos realizados por un router para enviar un paquete IPv4 son:

1. Almacena el paquete en buffer.
2. Inspecciona la versión, si no es compatible lo descarta.
3. Verifica TTL, si es 0 lo descarta.
4. Observa IHL.
5. Conociendo IHL calcula el checksum.
6. Con la dirección de destino observa en la tabla de enrutamiento por donde enviarlo.
7. Decrementa TTL y recalcula checksum.
8. Reenvía el paquete.

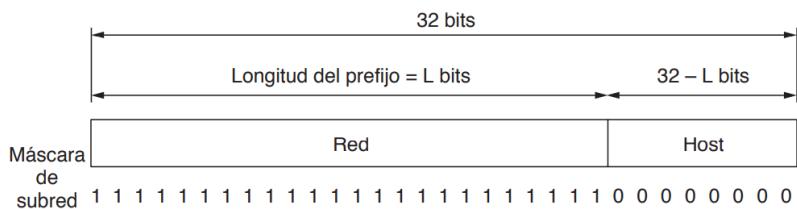
## DIRECCIONES IP

Una dirección IP se refiere a una interfaz de red. Un host en dos redes debe tener dos IP. Normalmente los hosts están en una sola red, en cambio los enrutadores tienen varias interfaces.

### Prefijos

Las direcciones IP son jerárquicas. Se componen de una porción de red de longitud variable en los bits superiores y una porción de host en los bits inferiores. La porción de red tiene el mismo valor para todos los hosts de una sola red (una red es un espacio contiguo de direcciones IP) y se denomina **prefijo**.

Los prefijos suelen describirse por su longitud ("/16") la cual corresponde a una máscara binaria de 1 en la porción de red denominada **máscara de subred**. Se aplica una AND a la máscara con la dirección IP para extraer solo la porción de red.

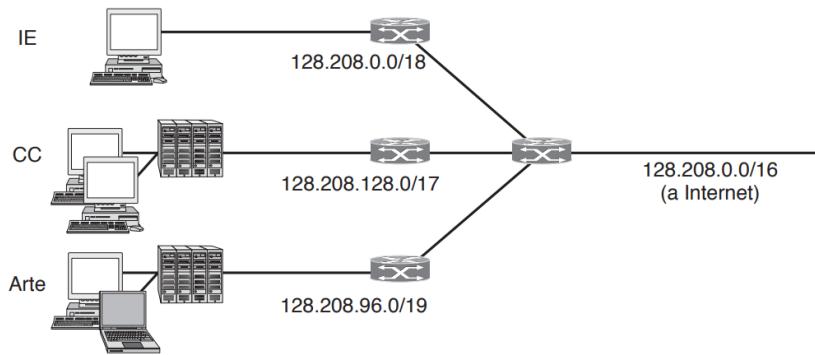


**Figura 5-48.** Un prefijo y una máscara de subred del protocolo IP.

## Subredes

El enruteamiento por prefijo requiere que todos los hosts en una red tengan el mismo número de red. Puede haber problemas cuando las redes aumentan su tamaño.

La solución es dividir el bloque de direcciones en varias partes para uso interno en múltiples redes, que actúen como una sola red para el mundo exterior.



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

Al llegar un paquete, el router analiza su dirección de destino y verifica a cuál subred pertenece. Aplica una AND a la dirección con la máscara de subred para cada subred y verifica que el resultado sea el prefijo correspondiente.

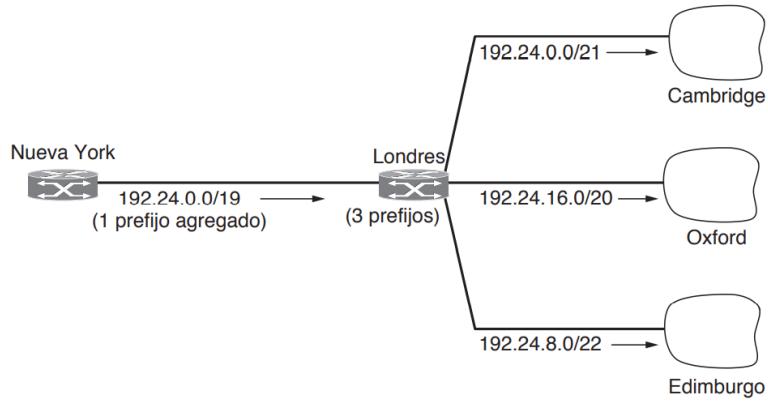
## Supernetting

Soluciona el problema de la explosión de las tablas de enruteamiento. Para reducir el tamaño de las mismas podemos aplicar la misma perspectiva que en las subredes pero en lugar de dividir un bloque de direcciones en subredes, aquí combinamos varios prefijos pequeños en un solo prefijo más grande. Este proceso se conoce como **agregación de rutas**. Al prefijo más grande se lo suele denominar **superred**.

Universidad	Primera dirección	Última dirección	Cuántas	Prefijo
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edimburgo	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Disponible)	194.24.12.0	194.24.15.255	1024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

**Figura 5-50.** Un conjunto de asignaciones de direcciones IP.

Ahora las direcciones IP están contenidas en prefijos de diversos tamaños, es responsabilidad del router tener la información del prefijo correspondiente. Este diseño funciona con las subredes y se denomina **CIDR**.



**Figura 5-51.** Agregación de prefijos IP.

Es posible además traslapar prefijos. La regla es que los paquetes se envíen a la dirección de ruta más específica, es decir, al prefijo más largo coincidente (menor cantidad de direcciones IP).

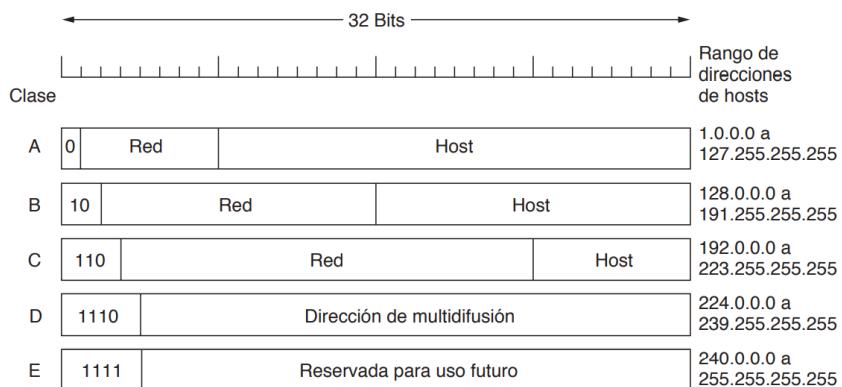
Al llegar un paquete se explora la tabla de enrutamiento para determinar si el destino está dentro del prefijo. Si coinciden varias entradas se utiliza la del prefijo más largo.

### Subnetting y Supernetting

En Subnetting se divide una red en varias subredes con prefijo mayor (máscara con más bits). En cambio en CIDR se agrupan varias redes bajo una superred de prefijo menor.

### Direccionamiento con clases y especial

Antes de 1993 las direcciones IP se dividían en las siguientes 5 clases:



**Figura 5-53.** Formatos de direcciones IP.

Esto se denominó **direccionamiento con clases**. Es un diseño jerárquico donde los tamaños de los bloques de dirección son fijos. Para la mayoría de las organizaciones una red clase A con 16 millones de direcciones es demasiado, una C con 256 no es suficiente y una B con 65536 también es demasiado para la mayoría. Por ello se introdujeron:

- Subredes para asignar de manera flexible bloques de direcciones dentro de una organización.
- Luego CIDR para reducir el tamaño de la tabla de enrutamiento global.

Actualmente, los bits que indican si una dirección IP pertenece a una red clase A, B o C ya no se usan. La clase D se sigue usando en internet para multidifusión.

**Figura 5-54.** Direcciones IP especiales.

Existen otras IP con significados especiales. La IP “0.0.0.0” es usada por los hosts al momento de encenderlos. Las IP con número de red 0 se refieren a la red actual, permitiendo que las máquinas hagan referencia a su propia red sin conocer su número. La dirección de todos unos (255.255.255.255) indica a todos los hosts en una red especificada.

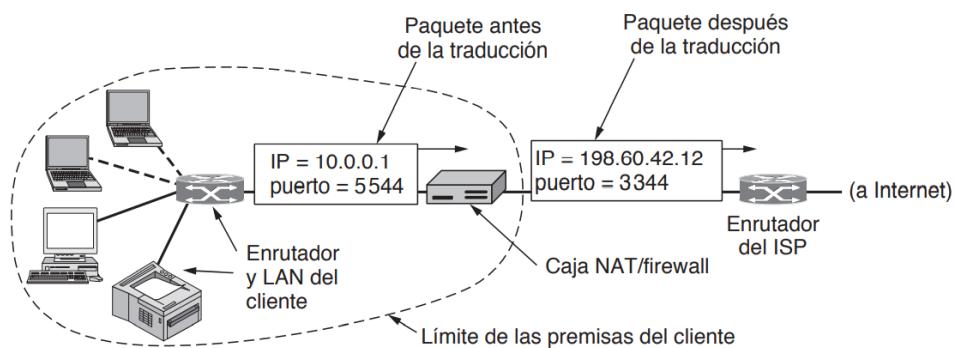
## NAT: Traducción de dirección de red

Las IP son escasas, una solución a largo plazo para cubrir el incremento de ellas es que internet migre a IPv6. La solución rápida utilizada actualmente es NAT.

La idea de NAT es que el ISP asigne a cada cliente una sola IP para el tráfico de internet, denominada **IP pública**. Dentro de la red del cliente, cada computadora obtiene una dirección IP única, denominada **IP privada**, para enrutar el tráfico interno. Antes de que un paquete salga de la red del cliente al ISP, la dirección IP privada se traduce a IP pública. Hay 3 rangos reservados a IP privada:

- 10.0.0.0 – 10.255.255.255 /8
  - 172.16.0.0 – 172.31.255.255 /12
  - 192.168.0.0 – 192.168.255.255 /16

En la siguiente figura, dentro de las premisas del cliente, cada máquina tiene una IP única de la forma  $10.x.y.z$ . Al salir un paquete, pasa a través de la capa NAT que convierte la dirección IP de origen interna a la dirección IP verdadera del cliente. Es posible integrar la caja NAT dentro de un router o modem ADSL.



**Figura 5-55.** Colocación y funcionamiento de una caja NAT.

Para que pueda enrutarse la respuesta de vuelta debe usarse la capa de transporte. La mayoría de los paquetes IP llevan cargas útiles de TCP o UDP, las cuales poseen encabezados con un puerto de origen y uno de destino. Estos puertos son enteros de 16 bits que indican donde empieza y termina la conexión TCP, por lo que proporcionan el campo requerido para hacer que NAT funcione.

Cuando un proceso desea establecer una conexión TCP con un proceso remoto, se conecta a un puerto TCP sin usar en su propia máquina (puerto origen) y le indica al código TCP donde enviar los

paquetes entrantes pertenecientes a esa conexión. El proceso también da un puerto de destino para indicar a quien dar los paquetes en el lado remoto.

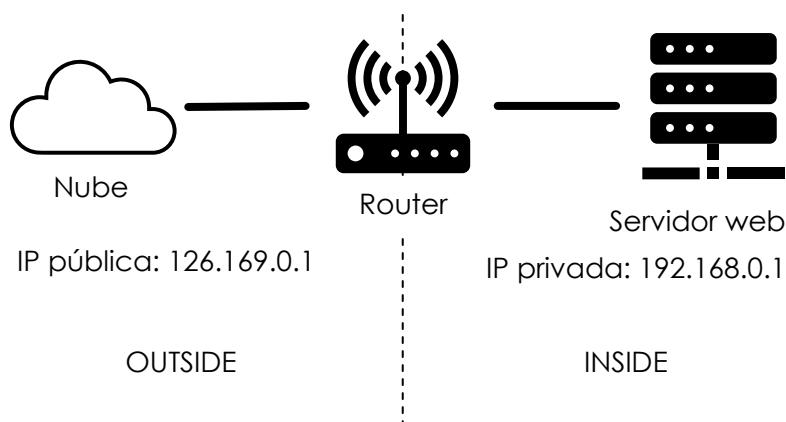
Cuando un paquete llega a la caja NAT desde el ISP, el puerto de origen del encabezado se extrae y utiliza como índice en la tabla de asignación de la caja NAT. La IP privada y el puerto de origen TCP se insertan en el paquete. Los checksum de IP y TCP se recalculan e insertan al paquete, el cual pasa entonces al enrutador del cliente para su entrega normal.

NAT presenta las siguientes objeciones:

1. Viola el modelo arquitectónico de IP (una IP identifica a una sola máquina).
2. Quebranta el modelo de conectividad extremo a extremo de internet. Como la asignación NAT se establece mediante los paquetes de salida, no se pueden aceptar paquetes hasta después de estos.
3. Cambia a internet de una red sin conexión a un tipo especial orientado a conexión. La caja NAT debe mantener la información para cada conexión a través de ella.
4. Viola la regla más fundamental de los protocolos distribuidos en capas: la capa  $k$  no puede hacer ninguna suposición sobre lo que la capa  $k + 1$  ha puesto en la carga útil.
5. En internet no se exige que los procesos usen TCP o UDP. Si un usuario usa un nuevo protocolo de transporte la caja NAT hará que falle la aplicación por no poder localizar el puerto origen TCP.
6. Algunas aplicaciones usan múltiples conexiones TCP/IP o puertos UDP. Como NAT no sabe de estos arreglos, no puede reescribir las direcciones IP ni justificarlas. Esta falta de entendimiento significa que las aplicaciones pueden fallar, a menos que se tomen precauciones especiales. A menudo es posible arreglar NAT para estos casos, pero no es una buena idea tener que arreglar el código en la caja NAT cada vez que surge una nueva aplicación.
7. Como el campo puerto de origen en TCP es de 16 bits, se limita a 65536 máquinas cada dirección IP. En realidad, los primeros 4096 puertos se reservan para usos especiales, por lo que cada IP puede manejar 61440 máquinas.

A pesar de todo esto, NAT se utiliza mucho en redes domésticas y de negocios pequeños. Se ha visto envuelta con los firewalls y la privacidad debido a que bloquea de manera predeterminada los paquetes entrantes no solicitados.

- **NAT estático (NAT 1:1):** una IP privada se traduce en una IP pública. Utilizado en servidores web donde estos pueden tener una IP de red privada y aun así ser visibles en internet.



Configuración del router:

1. Asociar IP privada a IP pública

```
router(config)# ip nat inside source static [IP privada] [IP pública]
```

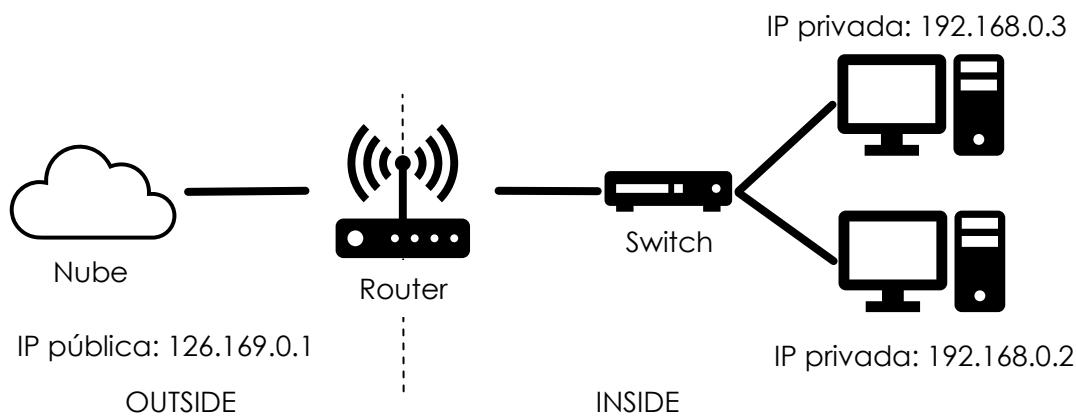
2. Indicar interfaz conectada a la red interna

```
router(config)# interface [Interfaz interna]
router(config)# ip nat inside
router(config)# exit
```

3. Indicar interfaz conectada a la red externa

```
router(config)# interface [Interfaz externa]
router(config)# ip nat outside
router(config)# exit
```

- **NAT dinámico:** la IP privada se mapea a una IP pública basándose en una tabla de direcciones IP registradas. El router NAT mantendrá una tabla de direcciones IP registradas y cuando la IP privada requiera acceso a internet, este colocará la IP en esta tabla.



Configuración del router:

1. Definir la lista de direcciones IP públicas

```
router(config)# ip nat pool [Nombre] [1ºIP pública] [Última] netmask [máscara]
```

El nombre se crea en este paso. La primer y última IP se obtienen al colocar la IP pública y su máscara en una calculadora de IP.

2. Configurar la lista de acceso para que conozca el rango de direcciones privadas a las que tiene que aplicar NAT.

```
router(config)# access-list [Número] permit [IP red privada] [Wildcard]
```

El número de lista se crea en este paso. La wildcard es la máscara negada. Si hay error de ACL es posiblemente porque está mal la IP red privada.

3. Indicar a NAT con qué lista de acceso va a controlar las IP que debe convertir.

```
router(config)# ip nat inside source list [Número] pool [Nombre] overload
```

4. Indicar interfaces conectadas a red interna y red externa.

```
router(config)# interface [interfaz interna]
router(config)# ip nat inside
router(config)# exit

router(config)# interface [interfaz externa]
router(config)# ip nat outside
router(config)# exit
```

## PROTOCOLOS DE CONTROL DE INTERNET

Además de IP, usado para transferencia de datos, internet posee varios protocolos de control.

### ICMP: Protocolo de mensajes de control en internet

Cuando algo inesperado ocurre durante el procesamiento de un paquete en un router, ICMP informa al emisor. Los mensajes ICMP se transportan encapsulados en un paquete IP.

Tipo de mensaje	Descripción
Time exceeded	TTL llegó a cero.
Parameter problem	Campo de encabezado inválido
Source quench	Paquete regulador
Redirect	Enseña a un router la geografía
Echo and echo reply	Verifica si una máquina está viva
Timestamp Request/reply	Igual a echo pero con marca de tiempo
Router advertisement/solicitation	Busca router cercano

### ARP: Protocolo de resolución de direcciones

Si bien en internet cada máquina tiene una o más IP, no son suficientes para enviar paquetes. Las NIC de la capa de enlace no entienden de direcciones de internet. En ethernet, cada NIC tiene una dirección ethernet única de 48 bits y envían y reciben tramas basadas en estas direcciones.

La forma en que se convierten las direcciones IP en direcciones de la capa de enlace de datos, como Ethernet, es la siguiente.

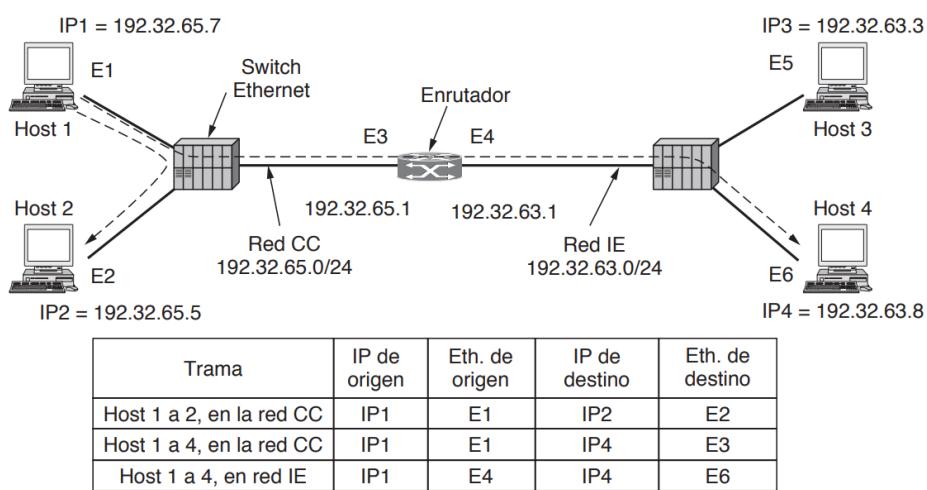


Figura 5-61. Dos redes LAN Ethernet conmutadas, unidas por un enrutador.

1. El host 1 envía un paquete a un usuario en host 2 de la red *CC*.
  - a. Primero encuentra la IP para el host 2 mediante una consulta por el DNS que le devuelve dicha dirección.
  - b. El software de la capa superior en el host 1 elabora ahora un paquete con 192.32.65.5 (IP de host 2) en el campo dirección de destino y lo entrega al software de IP para que lo transmita.
  - c. Para encontrar la dirección Ethernet de destino la mejor opción es que el host 1 envíe un paquete de difusión hacia Ethernet y pregunte quién posee la dirección IP 192.32.65.5.
  - d. Cada máquina en la red *CC* verificará su dirección IP. El host 2 responderá con su dirección de Ethernet *E2*. El protocolo para hacer esta pregunta y obtener la respuesta se denomina **ARP**.
  - e. El software IP en el host 1 crea una trama Ethernet dirigida a *E2*, coloca el paquete IP en el campo de carga útil y lo descarga hacia Ethernet.
2. El host 1 envía un paquete al host 4 de la red *IE*.
  - a. Verá que la dirección IP de destino no está en la red *CC*. Sabe enviar todo ese tráfico fuera de la red al router, el cual también se conoce como **puerta de enlace predeterminada**, que por convención es la dirección más baja de la red 198.31.65.1.
  - b. Para enviar una trama al router, el host 1 debe conocer la dirección Ethernet de la interfaz del router en la red *CC*. Para descubrirla envía una difusión ARP para 198.31.65.1 y aprende *E3*.
  - c. Envía la trama. Cuando la NIC ethernet del router recibe esta trama entrega el paquete al software IP. Gracias a la máscara de red, sabe que el paquete se debe enviar a la red *IE*, donde llegará a host 4. Si el router no conoce la dirección Ethernet para el host 4, usará ARP nuevamente.

### **DHCP: Protocolo de configuración dinámica de host**

ARP asume que los hosts están configurados con cierta información básica, como sus propias IP. Esta información la obtienen configurando de forma manual cada computadora o utilizando **DHCP**. Este protocolo de capa de aplicación tipo cliente – servidor, permite a los clientes de una red IP obtener sus parámetros de configuración automáticamente.

Generalmente, un servidor posee una lista de direcciones IP dinámicas y las asigna a los clientes conforme estas van estando libres, conociendo quién ha estado en posesión de esa IP, cuánto tiempo y a quién la pasó.

Al iniciar una máquina, esta tiene integrada una dirección de capa de enlace en la NIC pero no una IP. Similar a ARP, la máquina difunde una solicitud de dirección IP en su red utilizando un paquete **DHCP DISCOVER** que debe llegar al servidor DHCP. Cuando el servidor recibe la solicitud, asigna una IP libre y la envía al host en un paquete DHCP OFFER. El servidor identifica a un host mediante su dirección Ethernet transportada por el paquete DHCP DISCOVER.

DHCP se usa para configurar todo tipo de parámetros en internet. Los ISP lo usan para establecer los parámetros de los dispositivos a través del enlace de acceso a internet, así los clientes no tienen que comunicarse por teléfono con sus ISP para obtener esa información. Esto incluye información de la máscara de red, IP de Gateway predeterminada y direcciones IP de los servidores DNS y de tiempo.

El protocolo DHCP incluye 3 métodos de asignación de direcciones IP:

1. **Asignación manual o estática:** asigna una IP a una máquina determinada.
2. **Asignación automática:** asigna una IP de forma permanente a una máquina la primera vez que hace solicitud al servidor DHCP y hasta que el cliente la libera. Se utiliza cuando el número de clientes no varía demasiado.
3. **Asignación dinámica:** permite la reutilización dinámica de las direcciones IP. El administrador de la red determina un rango de direcciones IP y cada dispositivo conectado a la red está configurado para solicitar su dirección IP al servidor cuando la tarjeta de interfaz de red se inicializa.

## PROTOCOLO IPV6

El protocolo IPv6 tiene direcciones más largas que IPv4 (128 bits). Esto resuelve el problema de proporcionar un suministro efectivamente ilimitado de direcciones de Internet.

Además posee una simplificación en el encabezado. Solo contiene 7 campos, lo cual permite a los enruteadores procesar los paquetes con más rapidez, mejorando la velocidad de transmisión real y el retardo.

Otra mejora es un soporte mejorado para las opciones. Los campos que antes eran requeridos ahora son opcionales, ya que no se usan con tanta frecuencia. La forma en que se representan las opciones es diferente y así es más fácil para los routers omitir porciones no destinadas a ellos. Esto agiliza el tiempo de procesamiento de los paquetes.

Otras dos mejoras son un gran avance en seguridad y calidad de servicio.

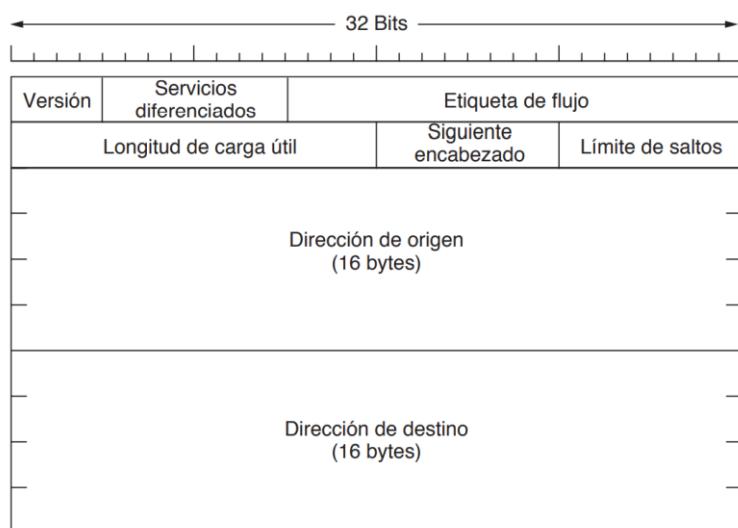


Figura 5-56. El encabezado fijo de IPv6 (requerido).

- **Versión:** es 6 para IPv6.
- **Servicios diferenciados:** distingue la clase de servicio para los paquetes con distintos requerimientos de entrega en tiempo real.
- **Etiqueta de flujo:** proporciona el medio para que origen y destino marquen grupos de paquetes que tengan los mismos requerimientos y que la red deba tratar de la misma forma, formando una pseudo conexión. Los flujos son un intento de tener la flexibilidad de una red de datagramas y las garantías de una red de CV.

- **Longitud de carga útil:** indica cuantos bytes van después del encabezado de 40 bytes. El encabezado no se cuenta como parte de la longitud.
- **Siguiente encabezado:** campo por la cual pudo simplificarse el encabezado ya que permite encabezados adicionales (opcionales) de extensión. Este indica cuál de ellos sigue, en caso de que haya. Si no hay más, este campo indica el protocolo de transporte al que se entregará el paquete.
- **Límite de saltos:** evita que los paquetes vivan eternamente. Idéntico al TTL de IPv4.
- **Dirección origen y destino:** dirección de 16 bytes fija. Se desarrolló una nueva notación para escribir direcciones de 16 bytes. Se escriben como 8 grupos de 4 dígitos hexadecimales, separando los grupos por 2 puntos (8000:0000:0000:0000:0123:4567:89AB:CDEF).

Se eliminaron los campos relacionados con la fragmentación, ya que IPv6 utiliza el descubrimiento de MTU de la ruta.

Además se eliminó el checksum ya que al calcularlo se reduce el desempeño. teniendo en cuenta que las capas de transporte y de enlace tienen su checksum, la ventaja de tener otro checksum en capa de red no valía el costo de desempeño que generaba.

### **Encabezados de extensión**

Ocasionalmente se requieren algunos de los campos faltantes de IPv4 en IPv6, por ello se introdujo el concepto de encabezados de extensión (opcionales). Se usan para proporcionar información adicional codificada de una manera más eficiente. Si hay más de uno presente deben aparecer justo después del encabezado fijo y de preferencia en el orden de la siguiente lista:

Encabezado de extensión	Descripción
Opciones por salto	Información diversa para los enrutadores
Opciones de destino	Información adicional para el destino
Enrutamiento	Lista informal de routers a visitar
Fragmentación	Manejo de fragmentos de datagramas
Autentificación	Verificación de identidad de emisor
Carga útil de seguridad cifrada	Información sobre el contenido cifrado

Algunos encabezados tienen un formato fijo, otros contienen un número variable de opciones de longitud variable. Cada elemento en ellos está codificado como una tupla (*Tipo, Longitud, Valor*).

## CAPA DE TRANSPORTE

Se basa en la capa de red para proveer transporte de datos de un proceso en una máquina origen a una máquina destino, con un nivel de confiabilidad independiente de las redes físicas que se utilizan. Ofrece las abstracciones que necesitan las aplicaciones para usar la red. Su objetivo es la multiplexión entre aplicaciones. Las direcciones de esta capa se denominan **puertos**.

### El servicio de transporte

#### SERVICIOS PROPORCIONADOS A LAS CAPAS SUPERIORES

La meta fundamental es proporcionar un servicio de transmisión de datos eficiente, confiable y económico, utilizando los servicios de la capa de red. El hardware o software que se encargado se denomina **entidad de transporte** y se ubica normalmente en kernel.

La relación lógica entre las capas de red, transporte y aplicación se muestra en la siguiente figura:

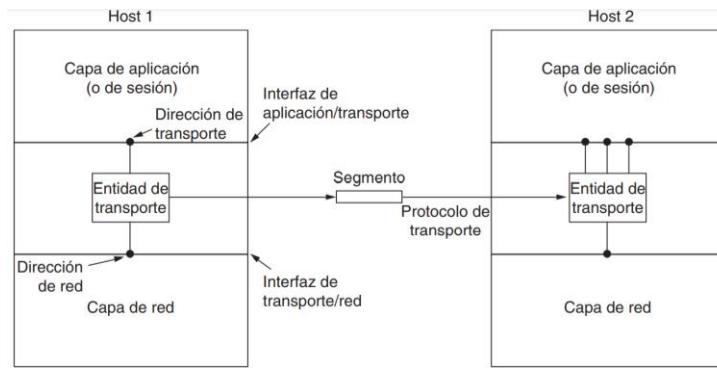


Figura 6-1. Las capas de red, transporte y aplicación.

Existen 2 tipos de servicios:

- **Orientado a conexión:** parecido al servicio de la capa de red.
- **Sin conexión:** parecido al de capa de red, pero puede ser difícil proveer un servicio de transporte sin conexión sobre uno de red con conexión, ya que es ineficiente establecer una conexión para enviar solo un paquete y deshacerla.

El código de transporte se ejecuta en máquinas de usuario. Si la capa de red ofrece un servicio inadecuado, los usuarios no tienen control real sobre ella, no pueden resolver problemas. La solución es colocar encima otra capa que mejore la calidad de servicio. La entidad de transporte puede detectar problemas y compensarlos.

La capa de transporte hace que el servicio de transporte sea más confiable. Las primitivas de transporte se pueden implementar como llamadas a procedimientos de biblioteca para que sean independientes de las primitivas de red.

Las capas 1 a 4 son capas del proveedor de servicio, las superiores son de usuario, siendo la de transporte el límite.

## PRIMITIVAS DEL SERVICIO DE TRANSPORTE

La capa de red debe proporcionar una interfaz del servicio de transporte. Cada servicio tiene su propia interfaz. Los servicios de red son usados únicamente por las entidades de transporte y las primitivas de transporte por muchos programas, por lo que deben ser convenientes y fáciles de usar.

Primitivas orientadas a conexión		
Primitiva	Paquete enviado	Significado
LISTEN	-	Se bloquea hasta que algún proceso intente conectarse
CONNECT	CONNECTION REQ.	Intenta establecer conexión
SEND	DATA	Envía información
RECEIVE	-	Se bloquea hasta que llega un paquete DATA
DISCONNECT	DISCONNECTION REQ.	Solicita liberación de conexión

Los **segmentos** son los mensajes enviados entre una entidad de transporte y otra, contenidos en paquetes (capa de red) que se encuentran en tramas (capa de enlace).

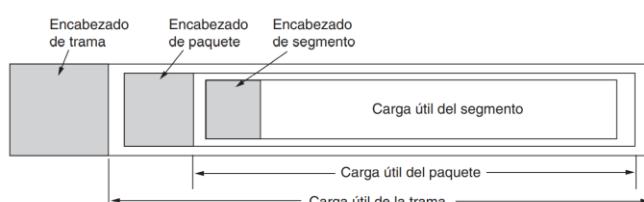
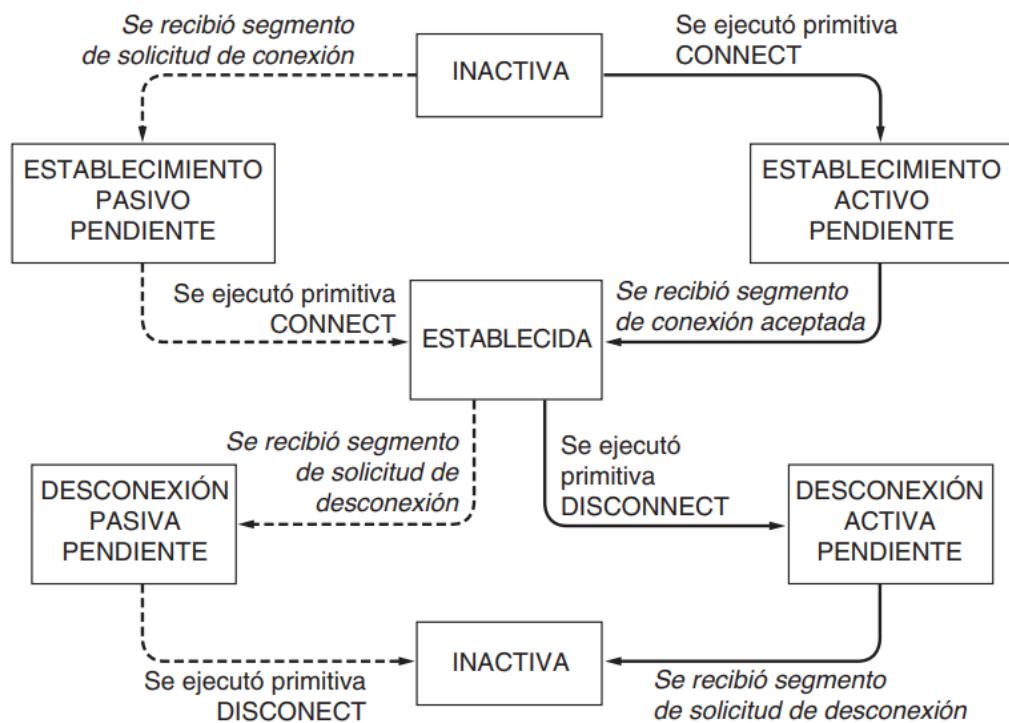


Figura 6-3. Anidamiento de segmentos, paquetes y tramas.

CONNECT envía un segmento CONNECTION REQ. al servidor. La entidad de transporte verifica que el servidor esté en LISTEN, lo desbloquea y envía un segmento CONNECTION ACCEPTED al cliente. Este se desbloquea y establece conexión. Luego intercambian datos mediante SEND y RECEIVE.

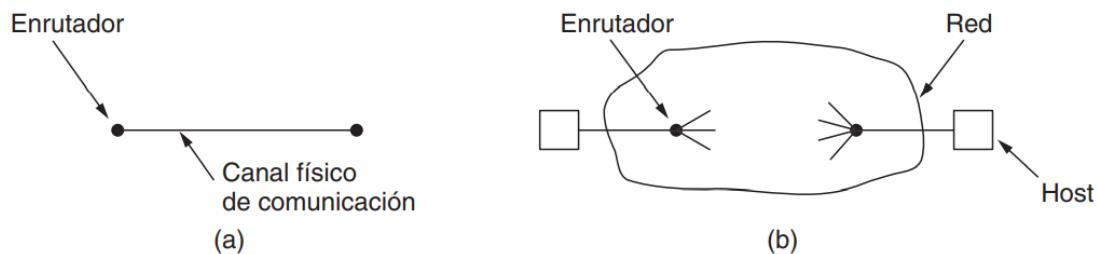
Diagrama de estado para establecer y liberar una conexión:



**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** entre las dos entidades de transporte. En ciertos aspectos, los protocolos de transporte se parecen a los de enlace de datos, sin embargo, en esta capa el canal física se sustituye por la red completa.



**Figura 6-7.** (a) Entorno de la capa de enlace de datos. (b) Entorno de la capa de transporte.

## DIRECCIONAMIENTO EXPLÍCITO DE LOS DESTINOS

Si un proceso de aplicación desea establecer conexión con otro debe especificar a cuál se conectará usando la dirección de transporte en la que el proceso escucha la solicitud de conexión. En internet estos puntos terminales se denominan **puertos** (TSAP).

1. Un proceso servidor de correo se enlace con el TSAP 1522 en host 2 y espera una llamada entrante.
2. Un proceso cliente en host 1 envía un mensaje de correo, se enlace con el TSAP 1208 y emite CONNECT, especificando a TSAP 1208 en host 1 como origen y TSAP 1522 en host 2 como destino. Se establece la conexión.
3. El proceso cliente envía un mensaje de correo.
4. El servidor de correo responde para decir que entregará el mensaje.
5. Se libera la conexión de transporte.

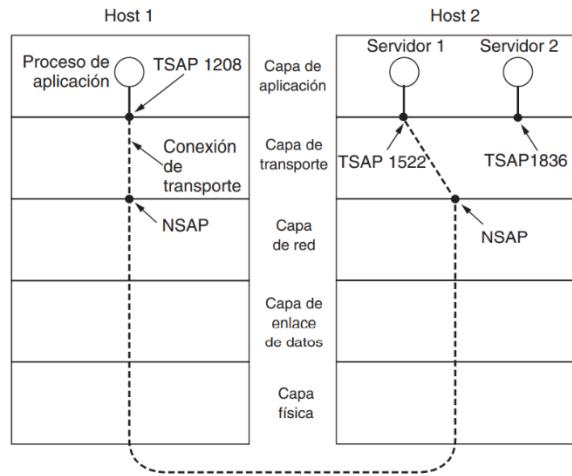


Figura 6-8. Los puntos TSAP y NSAP, junto con las conexiones de transporte.

Para comunicarse con procesos sin dirección TSAP bien conocida por adelantado puede usarse un proceso especial denominado **asignador de puertos** (portmapper) para encontrar la dirección TSAP correspondiente a un nombre de servicio específico.

1. El usuario establece conexión al asignador de puertos (que escucha en un TSAP conocido) y envía un mensaje especificando el nombre de servicio.
2. El asignador le regresa la dirección TSAP.
3. El usuario libera la conexión con el asignador y establece con el TSAP que le llegó.

Además, muchos procesos servidor se utilizan poco, por lo que es un desperdicio que estén escuchando en un TSAP permanentemente. En el **protocolo de conexión inicial**, en lugar de que cada uno de los servidores escuche en un TSAP conocido, cada máquina tiene un servidor de procesos especial, el cual actúa como proxy de los procesos servidores que se usan menos.

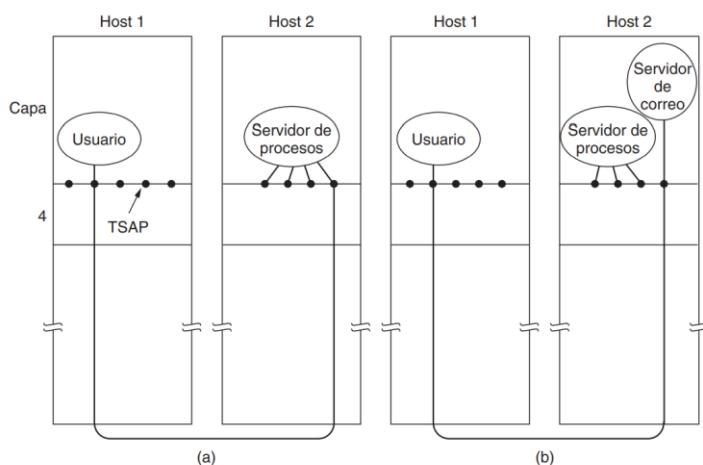


Figura 6-9. Cómo establece un proceso de usuario en el host 1 una conexión con un servidor de correo en el host 2, mediante un servidor de procesos.

## ESTABLECIMIENTO DE UNA CONEXIÓN

Surge un problema cuando el emisor envía paquetes a rincones lejanos de la red, donde los duplicados con retraso se consideran erróneamente paquetes nuevos. La solución es eliminar estos paquetes viejos que continúan vagando.

Para ello, el TTL de un paquete puede restringirse a un máximo conocido mediante:

- **Un diseño de red restringido:** esta técnica incluye cualquier método que evite que los paquetes hagan ciclos, combinado con una manera de limitar el retardo, incluyendo la congestión a través de la trayectoria más larga posible (conocida). Es difícil ya que el alcance de las redes puede variar.
- **Colocando un contador de saltos en cada paquete:** se inicializa el contador de saltos con un valor apropiado y se decrementa cada vez que se reenvía el paquete. El protocolo de red descarga cualquier paquete con contador cero.
- **Marcar el tiempo en cada paquete:** cada paquete lleva la hora de su creación y los routers descartan los que hayan pasado cierto tiempo. Requiere que los relojes de los routers estén sincronizados, lo cual es complicado.

En la práctica se necesita garantizar además que todas las confirmaciones de recepción estén eliminadas, por lo que se introduce un periodo  $T$ , múltiplo pequeño del tiempo de vida máximo verdadero del paquete el cual depende del protocolo. Si esperamos un tiempo  $T$  después de enviar un paquete, podemos asegurarnos de que todos sus rastros hayan desaparecido.

Para resolver el caso en que una máquina pierda la memoria de su estado tras una falla, se propuso equipar a cada host con un reloj. Al establecerse conexión, los  $k$  bits de menor orden del reloj son el número de secuencia inicial de  $k$  bits. Así cada conexión comienza a numerar sus segmentos con un número de secuencia distinto.

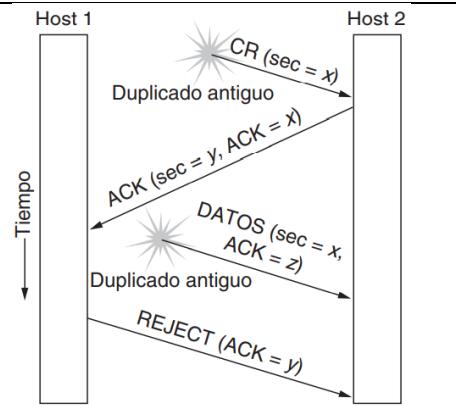
### Protocolo acuerdo de las 3 vías

Si bien el reloj resuelve el problema de diferenciar segmentos duplicados con retardo de segmentos nuevos, existe aún un inconveniente práctico. Como no recordamos los números de secuencia de una conexión a otra en el destino, no tenemos forma de saber si un segmento CONNECTION REQUEST es un duplicado o una conexión reciente. Para resolverlo se desarrolló el acuerdo de las 3 vías.

Situación	Diagrama
<p><b>Operación normal</b></p> <ul style="list-style-type: none"> <li>• Host 1 escoge un número de secuencia <math>x</math> y envía al host 2 CONNECTION REQ. con dicho número.</li> <li>• Host 2 responde con un SCK para confirmar recepción de <math>x</math> y su número de secuencia propio <math>y</math>.</li> <li>• Host 1 confirma recepción de numero de secuencia de host 2 (<math>y</math>) en el primer segmento de datos que envía.</li> </ul>	
<p><b>Presencia de duplicados con retardo</b></p> <ul style="list-style-type: none"> <li>• El primer segmento es un CONNECTION REQ. duplicado con retardo. Llega a Host 2 sin consentimiento de host 1.</li> <li>• Host 2 reacciona a este y envía a host 1 un segmento ACK para solicitar la comprobación de que haya tratado de establecer conexión.</li> <li>• Host 1 rechaza el establecimiento de conexión y host 2 advierte que ha sido engañado por un duplicado, abandonando la conexión.</li> </ul>	

### Segmentos CONNECTION REQ. y ACK con retardo en la red

- Host 2 recibe un CONNECTION REQ con retardo y lo contesta proponiendo  $y$  como número de secuencia inicial.
- Host 2 luego recibe un paquete de datos con confirmación de número de secuencia  $z$  en lugar de  $y$ , por lo que advierte que este paquete también es un duplicado con retardo.
- Luego llega el rechazo de conexión de host 1 al número de secuencia  $y$ .



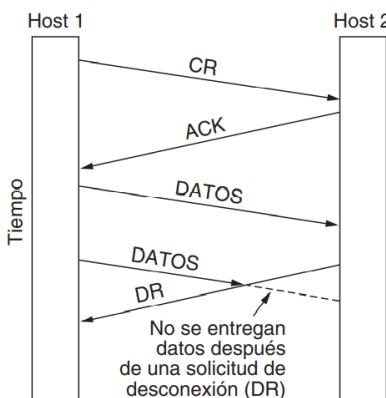
Lo importante es que no existe combinación de segmentos antiguos que pueda provocar una falla en el protocolo. **TCP** lo usa para establecer conexiones, donde los números de secuencia inicial son pseudoaleatorios por seguridad.

## LIBERACIÓN DE UNA CONEXIÓN

Puede realizarse de 2 formas:

- **Liberación asimétrica:** una de las partes interrumpe la conexión.
- **Liberación simétrica:** cada parte se libera por separado.

La liberación asimétrica es abrupta y puede provocar pérdida de datos.



**Figura 6-12.** Desconexión abrupta con pérdida de datos.

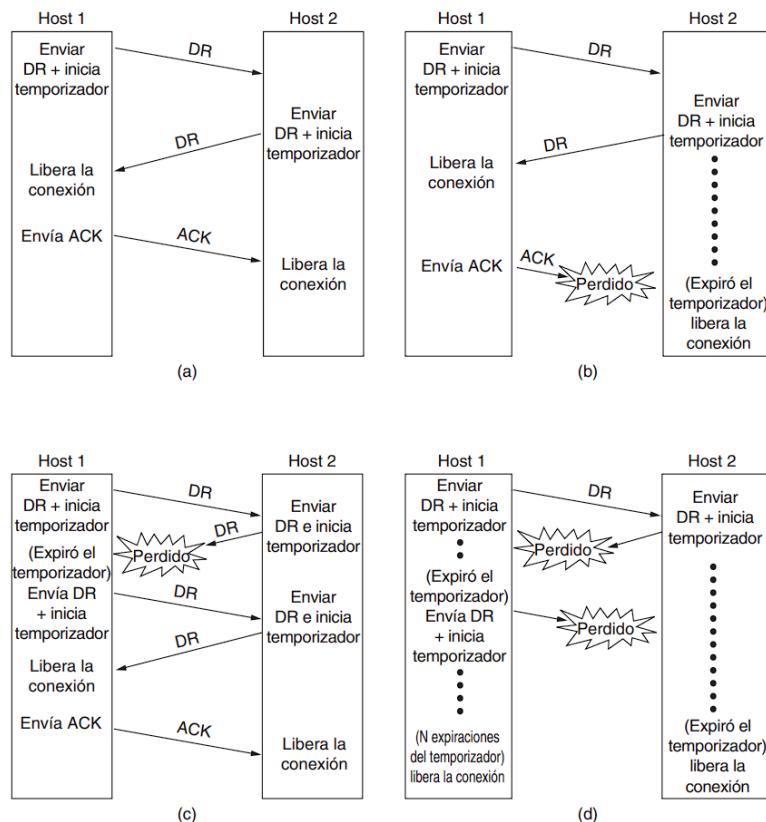
En liberación simétrica un host puede seguir recibiendo datos después de haber enviado DISCONNECT. Tiene el problema de “los dos ejércitos”, en el cual si ninguna de las partes está lista para desconectarse hasta que la otra lo esté, nunca habrá desconexión. Puede eludirse, evitando el acuerdo y pasando el problema al usuario de transporte.

Uno de los usuarios envía un segmento DR para iniciar la liberación de la conexión. El receptor devuelve un DR e inicia un temporizador. El emisor envía un ACK y libera, finalmente el receptor también libera.

Si el segundo DR se pierde, el usuario que inicia la desconexión no recibirá respuesta, expirará su temporizador y todo comenzará de nuevo. Si se pierde el ACK, al expirar el temporizador se libera la conexión.

Si todos los intentos de retransmitir el DR fallan, luego de  $N$  reintentos el emisor libera la conexión, expira el temporizador del receptor y también libera la conexión.

El protocolo falla cuando se pierden el DR inicial y las  $N$  retransmisiones. El emisor se da por vencido liberando la conexión sin que el receptor esté al tanto, por lo que seguirá activo provocando una conexión semiabierta. Una forma de eliminar esta conexión semiabierta es establecer que si no llegan segmentos durante cierto tiempo, se libere la conexión automáticamente. El usuario de transporte debe estar involucrado al decidirse la desconexión.



**Figura 6-14.** Cuatro escenarios de un protocolo para liberar una conexión. (a) Caso normal del acuerdo de tres vías. (b) Pérdida del último ACK. (c) Respuesta perdida. (d) Respuesta perdida y pérdida de los segmentos DR subsecuentes.

## CONTROL DE ERRORES Y ALMACENAMIENTO EN BUFFER

Ahora se detalla la forma en que se manejan las conexiones mientras están en uso. El control de errores asegura que los datos se entreguen con el nivel deseado de confiabilidad. El de flujo evita que un transmisor rápido saturé a un receptor lento. Las soluciones que se usan en capa de transporte utilizan los mismos mecanismos que se usan en capa de enlace:

1. Una trama transporta un CRC o checksum y un número de secuencia para identificarse.
2. El emisor retransmite hasta recibir una confirmación de recepción del receptor (ARQ – solicitud automática de recepción).
3. Existe un número máximo de tramas pendientes que permitirá el emisor en un momento dado y se detiene si el receptor no confirma recepción de las mismas con suficiente rapidez. Si este máximo es de 1 paquete, se denomina **protocolo parada y espera**. Ventanas más grandes permiten canalizaciones y mejoran el desempeño en enlaces largos y rápidos.
4. El protocolo de ventana deslizante combina estas características y se utiliza para transferencia de datos bidireccional.

A pesar de que estos mecanismos se utilizan en tramas de capa de enlace, existen diferencias en cuanto a función y grado respecto de los usados en capa de transporte:

- **Diferencia en función:** el checksum en capa de enlace protege a una trama en un enlace. En capa de transporte protege a un segmento en toda una trayectoria de red completa. Puede suceder que los paquetes se corrompan dentro de un router, lo cual no es protegido por el checksum de capa de enlace pero si por el de capa de transporte.
- **Diferencia en grado:** la mayoría de los enlaces inalámbricos solo puede tener una trama pendiente del emisor en cualquier momento dado debido al pequeño producto ancho de banda – retardo. Por otro lado, muchas conexiones TCP tienen un producto ancho de banda – retardo mayor a un segmento.

Por lo general, los protocolos de transporte usan ventanas deslizantes más grandes. Debe analizarse la cuestión de colocar los datos en búferes. Un host puede tener muchas conexiones, cada una se trata por separado, esto puede requerir una cantidad considerable de búferes para las ventanas deslizantes. Se necesitan búferes en emisor y receptor.

En emisor se necesitan para contener los segmentos transmitidos cuya recepción no ha sido confirmada. Como el emisor está usando búferes, el receptor tal vez pueda o no dedicar búferes específicos a conexiones específicas. Por ejemplo, un receptor puede tener un solo grupo de búferes para todas las conexiones, al entrar un segmento se intenta adquirir un búfer dinámicamente. Si hay un disponible se acepta el segmento, sino se descarta. Como el emisor está preparado para retransmitir (búferes en emisor) no hay daño permanente aunque se desperdicien algunos recursos.

La mejor solución de compromiso entre usar búferes en origen o destino depende del tipo de tráfico de la conexión:

- Ráfagas bajo BW (como el de terminales interactivas): no dedicar búferes, adquirirlos dinámicamente en cada extremo, confiando en los búferes del emisor si hay que descartar segmentos en receptor.
- Tráfico de alto BW: es mejor dedicar una ventana completa de búferes para permitir que los datos fluyan a la máxima velocidad. Estrategia usada por TCP.

La organización del grupo de búferes depende de los segmentos. Si la mayoría de los segmentos son del mismo tamaño aproximado, los búferes pueden organizarse como un grupo de tamaño idéntico, con un segmento por búfer (a). Esto trae problemas si hay una amplia variación del tamaño de los segmentos. Una forma de enfrentar este problema es usar búferes de tamaño variable (b). La ventaja es un mejor uso de la memoria, al costo de una administración más complicada. Otra posibilidad es dedicar un solo gran búfer circular por conexión. Este sistema es simple, pero solo hace un buen uso de la memoria cuando todas las conexiones están muy cargadas.

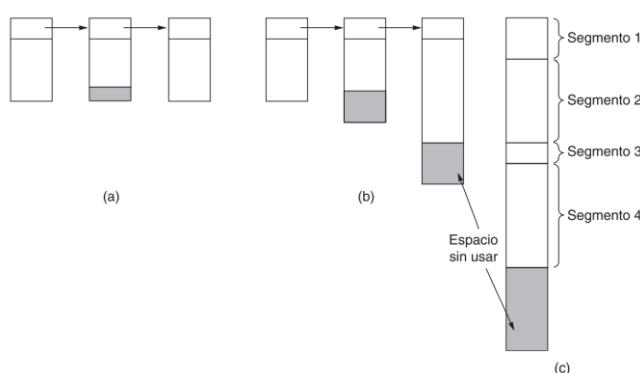


Figura 6-15. (a) Búferes encadenados de tamaño fijo. (b) Búferes encadenados de tamaño variable. (c) Un gran búfer circular por conexión.

Una manera razonable y general de administrar la asignación dinámica de búferes es desacoplar los búferes de las confirmaciones de recepción. Al principio el emisor solicita cierto número de búferes, luego el receptor otorga los que puede. Cada vez que el emisor transmite un segmento debe disminuir su asignación, y cuando ésta llegue a cero debe detenerse por completo. El receptor superpone por separado las confirmaciones de recepción y las asignaciones de búfer.

Si el segmento de asignación se perdió se encuentra en interbloqueo. Para evitarlo cada host debe enviar segmentos de control con confirmación de recepción y el estado de los búferes de cada conexión.

Si el espacio de búfer no limita el flujo máximo aparece otro cuello de botella, la **capacidad de transporte de la red**. Se necesita entonces de un mecanismo que limite las transmisiones del emisor con base en la capacidad de transporte de la red. Se utiliza un esquema de control de ventana deslizante en el que el emisor ajusta en forma dinámica el tamaño de la ventana para igualarla a la capacidad de transporte de la red. Esto significa que la ventana deslizante puede implementar el control de flujo y de congestión.

## MULTIPLEXIÓN

Si solo **hay una dirección de red disponible en un host**, todas las conexiones de transporte de la máquina deberán usarla. Al llegar un segmento, se requiere un mecanismo para saber a cuál proceso asignarlo. Esto se conoce como **multiplexión** (compartición de varias conversaciones a través de conexiones, CVs y enlaces físicos).

Si se tiene un **host con varias trayectorias de red posibles** y un usuario necesita más BW o más confiabilidad, una solución es tener una conexión que distribuya el tráfico entre varias trayectorias de red por round – robin. Esto se denomina **multiplexión inversa**.

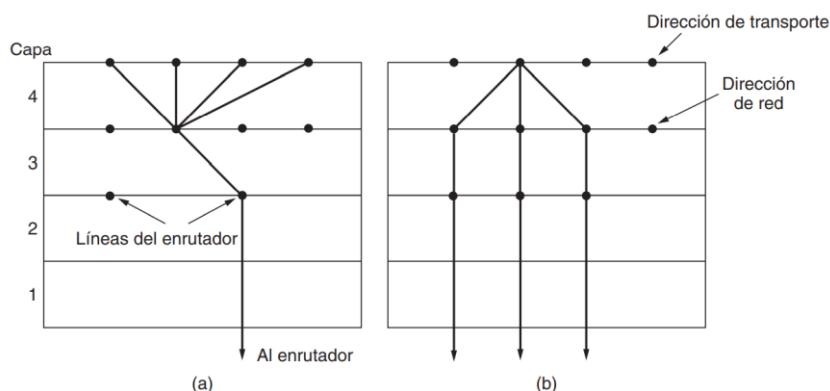


Figura 6-17. (a) Multiplexión. (b) Multiplexión inversa.

## RECUPERACIÓN DE FALLAS

Si un servidor falla a mitad de una transmisión, para recuperar su estado previo puede enviar un segmento de difusión para anunciarlo y solicitar a sus clientes que le informen del estado de las conexiones abiertas. Cada cliente puede estar en dos estados, segmento pendiente (S1) o sin segmentos pendientes (S0). El cliente solo debe retransmitir si está en S1.

Surge un problema si la entidad de transporte del servidor envía primero la confirmación de recepción y se interrumpe la conexión antes de que escriba en el proceso de aplicación. En este caso el cliente erróneamente se encontrará en estado S0. Sucede lo mismo si la entidad primero escribe en la aplicación y luego confirma recepción.

El servidor puede programarse para primero enviar confirmación o escribir en aplicación. El cliente puede programar se de 4 formas: retransmitir siempre último segmento, nunca retransmitir último segmento, retransmitir solo en estado S0 o retransmitir solo en estado S1. Esto da 8 combinaciones servidor – cliente y para cada una existe un conjunto de eventos que hacen fallar al protocolo.

Como conclusión, la recuperación de una falla en la capa  $N$  solo puede llevarse a cabo en la capa  $N + 1$ , siempre y cuando esta capa retenga suficiente información del estado como para reconstruir la condición en la que se encontraba antes de la falla.

## Los protocolos de transporte de internet: UDP

Protocolo de transporte sin conexión. Solo envía paquetes entre aplicaciones y deja que estas construyan sus propios protocolos.

Transmite segmentos con un encabezado de 8 bytes seguido de la carga útil. Los puertos sirven para identificar los puntos terminales dentro de las máquinas de origen y destino.

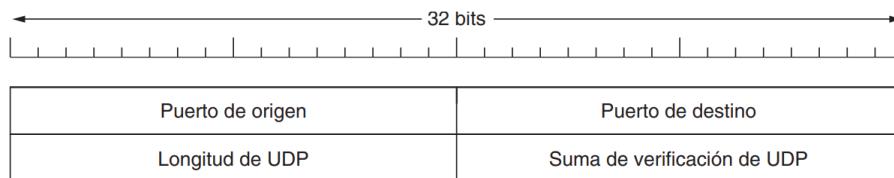


Figura 6-27. El encabezado UDP.

- **Puerto origen:** número de puerto relacionado con la aplicación del remitente del segmento UDP. Representa la dirección de respuesta para el destinatario, por lo que es opcional. Si no se especifica, los 16 bits son ceros y el destinatario no podrá responder.
- **Puerto destino:** puerto de la aplicación en el equipo receptor.
- **Longitud UDP:** incluye encabezado y datos. La mínima es de 8 bytes y la máxima de 65515.
- **Suma de verificación:** controla la integridad del segmento UDP.

El control de flujo, congestión o retransmisión les corresponde a los procesos de usuario.

UDP es ideal para aplicaciones que necesitan tener un control preciso sobre el flujo de paquetes, control de errores o temporización. Es muy útil en situaciones cliente – servidor, ya que el código no solo es simple, sino que se requieren menos mensajes que en TCP. UDP es utilizado por DNS.

## Los protocolos de transporte de internet: TCP

Para la mayoría de las aplicaciones de internet se necesita de una entrega en secuencia confiable, por lo que se utiliza el protocolo orientado a conexión TCP.

TCP proporciona un flujo de bytes confiable de extremo a extremo a través de una interred no confiable. Adapta dinámicamente las propiedades de una interred y se sobrepone a muchas fallas.

Cada máquina que soporta TCP tiene una entidad de transporte TCP que maneja flujos de TCP e interactúa con la capa IP. Acepta flujos de datos de usuario de procesos locales, los divide en fragmentos ( $\leq 64 kB$ ) y envía cada pieza como un datagrama IP independiente. Los datagramas con los datos TCP llegan a la máquina y se pasan a la entidad TCP que reconstruye el flujo de bytes originales. Debe proporcionar un buen desempeño con la confiabilidad que IP no proporciona.

## EL MODELO DEL SERVICIO TCP

El servicio de TCP se obtiene al hacer que servidor y cliente creen puntos terminales llamados sockets.

Cada uno con una dirección que consiste en la IP del host y el puerto.

Puerto	Protocolo	Uso
20, 21	FTP	Transferencia de archivos.
22	SSH	Inicio de sesión remoto, reemplazo de Telnet.
25	SMTP	Correo electrónico.
80	HTTP	World Wide Web.
110	POP-3	Acceso remoto al correo electrónico.
143	IMAP	Acceso remoto al correo electrónico.
443	HTTPS	Acceso seguro a web (HTTP sobre SSL/TLS).
543	RTSP	Control del reproductor de medios.
631	IPP	Compartición de impresoras.

Figura 6-34. Algunos puertos asignados.

Los puertos menores a 1024, denominados **puertos bien conocidos**, están reservados para servicios estándar que pueden iniciar usuarios privilegiados.

Las conexiones TCP son full dúplex y punto a punto. No soporta multidifusión ni difusión y funciona con flujos de bytes. Cuando una aplicación pasa datos a TCP, éste decide si enviarlos o almacenarlos.

## EL PROTOCOLO DE TCP

Cada byte posee su número de secuencia de 32 bits. Estos se transmiten en paquetes para la posición de ventana deslizante en una dirección y para las confirmaciones de recepción en la dirección opuesta.

Un segmento TCP consiste en un encabezado de 20 bytes seguido de cero o más bytes de carga. TCP puede acumular datos para formar un solo segmento o dividirlos en varios segmentos.

El tamaño de segmento es limitado por el tamaño de carga útil de 65515 bytes de IP y por la MTU de cada enlace. Como la MTU suele ser de 1500 bytes, es la que finalmente limita la longitud del segmento. Las implementaciones modernas de TCP utilizan el descubrimiento del MTU de la ruta para evitar fragmentar.

Las entidades TCP utilizan el protocolo de ventana deslizante con tamaño dinámico. Cuando un emisor transmite un segmento inicia un temporizador. Cuando llega el segmento a destino, la entidad TCP receptora devuelve un segmento que contiene un numero de confirmación de recepción igual al siguiente número de secuencia que espera recibir, junto con el tamaño de la ventana remanente. Si el temporizador del emisor expira antes de recibir la confirmación de recepción, el emisor transmite de nuevo el segmento.

## EL ENCABEZADO DE SEGMENTO TCP

Cada segmento comienza con un encabezado de formato fijo de 20 bytes. Este puede ir seguido de opciones, y luego los datos (si los hay). Segmentos sin datos son usados para control y confirmaciones.

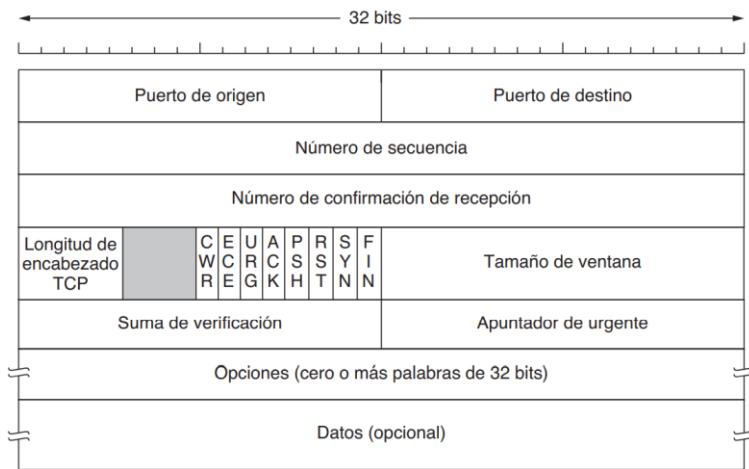


Figura 6-36. El encabezado TCP.

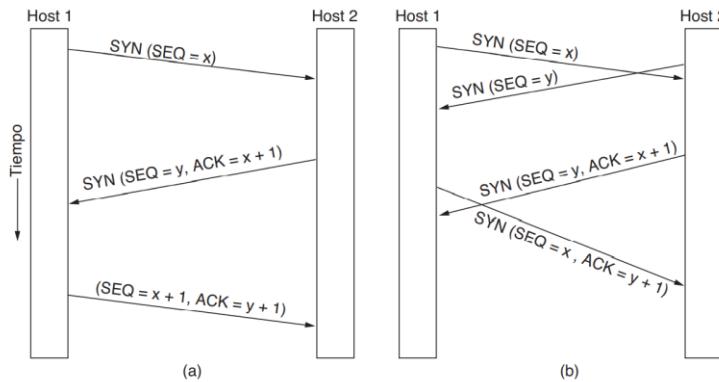
- **Puerto origen y destino:** puntos terminales locales de la conexión.
- **Numero de secuencia y de confirmación de recepción:** el segundo especifica el siguiente byte en el orden esperado, no el último recibido de manera correcta.
- **Longitud del encabezado TCP:** indica la cantidad de palabras de 32 bits contenidas en el encabezado TCP, dando a conocer el comienzo de los datos en el segmento.
- **Bits sin uso:** reservados para corregir errores en el diseño del protocolo.
- **Bits CWR y ECE:** manejan la congestión. CWR indica una ventana de congestión reducida del emisor al receptor.
- **Bit URG:** se establece en 1 si está en uso el apuntador urgente.
- **Apuntador urgente:** indica el desplazamiento en bytes a partir del numero de secuencia actual en el que se encuentran los datos urgentes.
- **Bit ACK:** en 1 indica que el número de confirmación de recepción es válido.
- **Bit PSH:** indica datos que se deben transmitir de inmediato.
- **Bit RST:** reestablece repentinamente una conexión.
- **Bit SYN:** establecimiento de conexiones. Diferencia segmentos CONNECTION REQ. de ACCEPTED mediante el valor del bit ACK.
  - Solicitud de conexión:  $SYN = 1, ACK = 0$ .
  - Respuesta de conexión:  $SYN = 1, ACK = 1$ .
- **Bit FIN:** libera una conexión y avisa que el emisor que no tiene más datos.
- **Tamaño de ventana:** cantidad de bytes que se pueden enviar, comenzando por el byte cuya recepción se ha confirmado.
- **Suma de verificación:** realizada con el encabezado, datos y pseudoencabezado conceptual. No es obligatoria.
- **Opciones:** posibilidad de agregar características adicionales.

## ESTABLECIMIENTO DE UNA CONEXIÓN TCP

Utiliza el acuerdo de las 3 vías:

1. Uno de los lados espera en forma pasiva una conexión mediante las primitivas LISTEN y ACCEPT.
2. El otro lado ejecuta CONNECT especificando dirección y puerto, tamaño máximo de segmento TCP y opcionalmente algunos datos de usuario. CONNECT envía un segmento TCP con el bit SYN encendido y el ACK apagado y espera respuesta.

3. Cuando un segmento llega a destino, la entidad TCP de ahí revisa si hay un proceso que haya ejecutado LISTEN en el puerto indicado. Si no lo hay, responde con el bit RST encendido para rechazar la conexión.
4. Si algún proceso está escuchando en el puerto, recibe el segmento TCP entrante y puede o no aceptar la conexión.
5. Si acepta la conexión, se devuelve un segmento de confirmación de recepción.



**Figura 6-37.** (a) Establecimiento de una conexión TCP en el caso normal. (b) Establecimiento de una conexión simultánea en ambos lados.

En caso de que dos hosts intenten establecer conexión al mismo tiempo entre los mismos sockets, solo se establece una conexión.

## LIBERACIÓN DE UNA CONEXIÓN TCP

Se visualiza a la conexión dúplex como un par de conexiones simplex y cada una se libera de manera independiente. Cualquiera de las partes puede enviar un segmento con el bit FIN establecido. Al confirmarse la recepción de FIN, se apaga.

Los datos pueden seguir fluyendo por el otro sentido. Cuando se apagan ambos sentidos se libera la conexión. Generalmente se requieren 4 segmentos TCP: un FIN y un ACK para cada sentido. Es posible que el primer ACK y el segundo FIN estén en el mismo segmento, con lo cual serían 3.

Ambos extremos de la conexión TCP pueden enviar segmentos FIN al mismo tiempo. Ambos se confirman de manera usual y se termina la conexión.

Para evitar el problema de los 2 ejércitos se usan temporizadores. Si no hay respuesta a un FIN, el emisor libera la conexión. El receptor notará que nadie está del otro lado de la conexión, expirará su temporizador y terminará la conexión.

## CAPA DE APLICACIÓN

Capa donde se encuentran las aplicaciones. Las capas inferiores proporcionan servicios de transporte, pero no hacen ningún trabajo verdadero para los usuarios.

Se necesitan protocolos de apoyo que permitan el funcionamiento de las aplicaciones. Uno de estos es el DNS, que maneja la asociación de nombres dentro de internet.

## DNS: sistema de nombres de dominio

Consiste en un esquema jerárquico de nombres basado en dominios y un sistema de base de datos distribuido. Asocia nombres de hosts con direcciones IP.

El funcionamiento es el siguiente:

- Para asociar un nombre con una IP, un programa de aplicación llama al **resolvedor** y le pasa el nombre.
- El resolvedor envía una consulta con el nombre a un servidor DNS local que busca y devuelve la dirección IP.
- El resolvedor le devuelve esta IP al programa de aplicación que lo llamó.
- El programa ahora puede establecer conexión TCP o enviar paquetes UDP a la IP recibida.

Todas estas consultas se realizan mediante paquetes UDP.

## EL ESPACIO DE NOMBRES DE DNS

Internet se divide en dominios de nivel superior que abarcan muchos hosts. Cada dominio se divide en subdominios que pueden también dividirse en más subdominios. Pueden representarse mediante un árbol donde las hojas son los subdominios finales. Los dominios de nivel superior se dividen en genéricos y países y son operados por registradores nombrados por la ICANN.

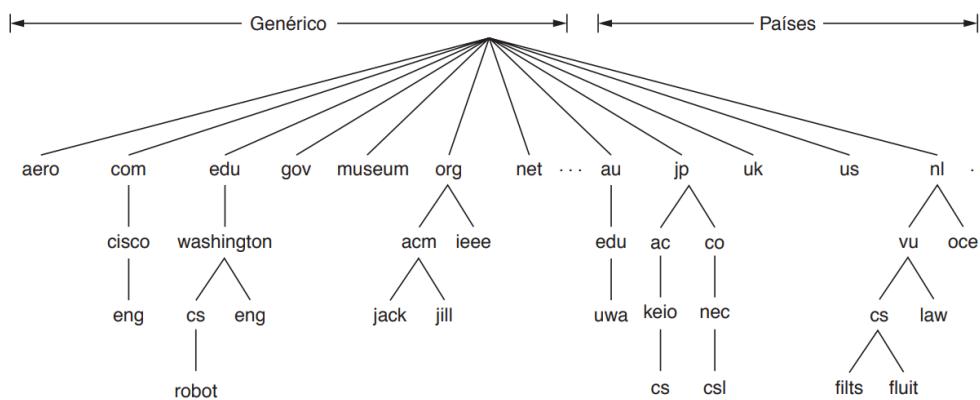


Figura 7-1. Una porción del espacio de nombres de dominio de Internet.

Cada dominio se nombra por la ruta hacia arriba, desde él hacia la raíz. Los componentes se separan por puntos. Los nombres de dominio pueden ser absolutos (terminan en punto) o relativos. Reflejan límites organizacionales, no redes físicas.

## REGISTROS DE RECURSOS DE DOMINIO

Cada dominio puede tener un grupo de registros de recursos asociados (base de datos del DNS). En un host individual, el registro más común es su IP. Cuando un resolvedor asigna un nombre de dominio al DNS, recibe registros de recursos asociados a dicho nombre. La principal función de DNS es relacionar los nombres de dominio con los registros de recursos.

Un registro de recursos es una 5 – tupla con el formato:

(Nombre_de_dominio, Tiempo_de_vida, Clase, Tipo, Valor)
---

1. **Nombre\_de\_dominio:** primary key de la búsqueda para atender consultas.
2. **Tiempo\_de\_vida:** indica la estabilidad del registro (cuanto mayor, más estable).
3. **Clase:** para información de internet es siempre IN.
4. **Tipo:** tipo de registro
5. **Valor:** número, nombre de dominio o cadena ASCII.

TIPO	Significado	Valor
A	dirección IPv4 de host.	entero de 32 bits.
AAAA	dirección IPv6 de host.	entero de 128 bits.
NS	servidor de nombres	nombre de un servidor para este dominio
CNAME	nombre canónico	nombre de dominio
PTR	apuntador	alias de una dirección IP

## SERVICIOS DE NOMBRES

En teoría un solo servidor de nombres podría contener toda la base de datos DNS y responder todas las consultas, pero en la práctica estaría sobrecargado. Para evitarlo, el espacio de nombres DNS se divide en zonas. Para evitarlo, el espacio de nombres DNS se divide en zonas que no se sobreponen, cada cual con una parte del árbol.

Cada zona se asocia a uno o más servidores de nombres, los cuales son hosts con la base de datos de la zona. Una zona debe tener un servidor de nombres primario (obtiene información de un archivo en su disco) y uno o más servidores secundarios (obtienen información del primario).

Al proceso de buscar nombre y encontrar dirección se lo denomina **resolución de nombres**. Si un resovedor tiene una consulta sobre un nombre de dominio, la pasa a uno de los servidores de nombre locales, si este dominio está bajo jurisdicción del servidor de nombres, devuelve los **registros de recursos asociados** (provienen de la autoridad administradora y siempre es correcto).

En caso de consulta remota, como *flits.cs.vu.nl* a *cs.washington.edu*:

- *flits.cs.vu.nl* envía su consulta al servidor de nombres local, este maneja la resolución de *flits* hasta que pueda devolver la respuesta deseada (consulta recursiva).
- El servidor raíz de nombres devuelve respuesta parcial y avanza a la siguiente consulta. El servidor de nombres local continua con la resolución, para lo cual emite otras consultas adicionales (consulta iterativa).
- La resolución de un nombre puede implicar ambos mecanismos de consulta.
- Las respuestas se colocan en caché para reducir los pasos de consulta. Estas son no autorizadas ya que los cambios en el host destino puede que no se propaguen a todas las cachés.

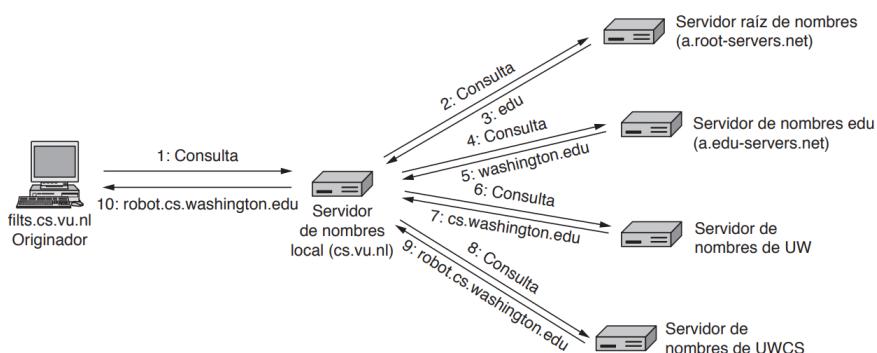


Figura 7-6. Ejemplo de un resovedor que busca un nombre remoto en 10 pasos.

Los mensajes DNS se envían en paquetes UDP con formato simple para consultas. Si no llega respuesta dentro de un intervalo de tiempo, el cliente DNS repite la consulta e intenta con otro servidor para ese dominio después de unos cuantos reintentos.

## WWW: World Wide Web

Marco arquitectónico para acceder a cierto contenido distribuido en millones de máquinas en internet.

## PANORAMA DE LA ARQUITECTURA

Desde el punto de vista usuario, la web es una colección de contenido en forma de páginas web que pueden contener vínculos a otras páginas (hipertexto). Cualquier elemento (texto, imagen, etc.) asociado a otra página se denomina hipervínculo.

Para obtener una página se envía una solicitud a uno o más servidores que responden con el contenido de la misma. El protocolo solicitud – respuesta utilizado es HTTP, basado en texto y opera sobre TCP.

Una página es estática si siempre es el mismo documento y dinámica si el contenido se generó bajo demanda mediante un programa o si contiene uno.

### El lado del cliente

Un navegador puede mostrar una página y capturar clics. Al seleccionar un elemento sigue al hipervínculo y obtiene la página seleccionada.

A cada página se le asigna una **URL** que es el nombre mundial de la página. El URL contiene 3 partes:

- Protocolo o esquema.
- Nombre DNS de la máquina.
- Ruta a la página específica.



Cuando un usuario hace click en un hipervínculo, el navegador realiza los siguientes pasos:

1. Determina URL seleccionada.
2. Pide al DNS al IP del servidor.
3. DNS responde con la IP.
4. Navegador realiza conexión TCP a la dirección IP recibida en el puerto 80 (puerto HTTP).
5. Navegador envía una solicitud HTTP para pedir la página (la ruta).
6. Servidor envía la página como respuesta HTTP.
7. Si la página incluye los localizadores URL necesarios para desplegar en pantalla, el navegador obtiene los otros URL mediante el mismo proceso (imágenes, videos, etc.).
8. El navegador despliega la página.
9. Se liberan las conexiones TCP si no hay más solicitudes.

Los esquemas más comunes de las URL son http, https, ftp, mailto, rtsp, etc.

### El lado del servidor

Los pasos que realiza el servidor al enviar la página solicitada por el navegador son:

1. Aceptar una conexión TCP de un cliente (navegador).
2. Obtener la ruta a la página.
3. Obtener el archivo de su disco.
4. Enviar el archivo al cliente.
5. Liberar la conexión TCP.

El acceso a los archivos suele ser el cuello de botella, las lecturas de disco son muy lentas comparadas con la ejecución de un programa. Una mejora es la permanencia en caché de los  $n$  archivos leído más recientemente en el servidor.

Otro problema es que solo procesa una solicitud a la vez. Para lidiar con ello puede hacerse al servidor multihilos. En un diseño, el servidor consiste en un módulo front – end que acepta todas las solicitudes entrantes y  $k$  módulos de procesamiento (hilos). Los  $k + 1$  hilos pertenecen al mismo proceso, tienen acceso a caché.

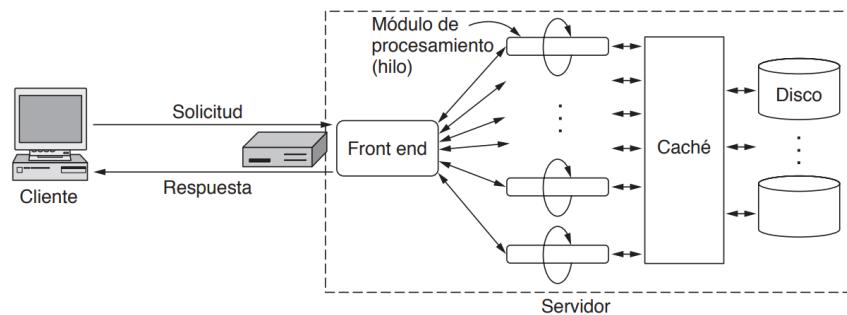


Figura 7-21. Un servidor web multihilos con un *front-end* y módulos de procesamiento.

Mientras uno o más hilos están bloqueados otros pueden estar trabajando en otras solicitudes.

El módulo front – end pasa cada solicitud al primero módulo disponible, que los atiende mediante algún subconjunto de los siguientes pasos:

1. Resuelve el nombre de la página web solicitada.
2. Realiza control de acceso en la página web.
3. Verifica caché.
4. Obtiene la página del disco o ejecuta un programa para construirla.
5. Determina el resto de la respuesta.
6. Regresa la respuesta al cliente.
7. Realiza una entrada en el registro.

## Cookies

Cadena con nombres que el servidor asocia a un navegador. Son almacenadas por los navegadores, así persisten entre una invocación y la otra del mismo. Permiten orientar a sesión un protocolo no orientado a sesión como HTTP.

## PÁGINA WEB ESTÁTICAS

Páginas web en su forma más simple. Son archivos guardados en servidor que se presentan de la misma forma cada vez. Los elementos básicos de estas son:

- **HTML – lenguaje de marcado de hipertexto:** permite producir páginas con texto, gráficos, videos, hipervínculos, etc.
- **Entradas y formularios:** cuadros o botones para que usuarios proporcionen información o tomen decisiones y envíen dicha información al dueño de la página.
- **CSS – hojas de estilo en cascada:** permite a los autores asociar texto con un estilo lógico.

## PÁGINAS WEB DINÁMICAS Y APLICACIONES WEB

Las aplicaciones web se ejecutan en el navegador y los datos de usuario se almacenan en servidores. Utilizan protocolos web para acceder a la información a través de internet y el navegador para mostrar una interfaz de usuario.

Las páginas web dinámicas generan contenido mediante programas que se ejecutan en servidor o navegador. El funcionamiento es el siguiente:

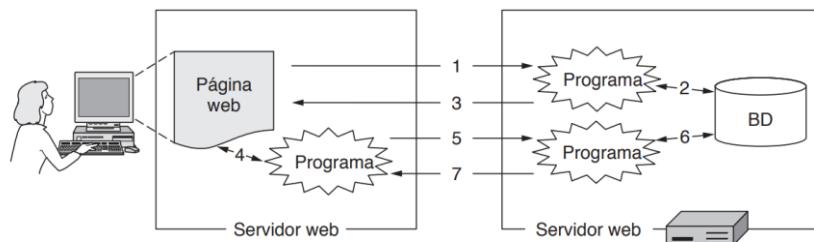


Figura 7-29. Páginas dinámicas.

1. La solicitud provoca la ejecución de un programa en servidor.
2. El programa consulta una base de datos para generar la página apropiada.
3. El servidor devuelve la página generada al navegador.
4. Esta puede contener programas que se ejecuten en navegador.
5. Para manejar interacciones, puede que el programa necesite más datos del servidor.
6. Obtiene más datos de la base de datos.
7. Devuelve una respuesta al programa.

### Generación de páginas web dinámicas del lado del servidor

La forma en que un servidor ejecuta un programa depende de su diseño. Se han desarrollado varias API estándar para que los servidores web invoquen programas:

1. **Primera API (CGI):** provee una interfaz para permitir que servidores web se comuniquen con programas de soporte y secuencias de comando que acepten entradas y generen páginas HTML como respuesta. Los programas viven en el directorio *cgi-bin*, visible en la URL.
2. **Segunda API:** incrusta pequeñas secuencias de comando dentro de las páginas HTML y hace que las ejecute el mismo servidor para generar la página. Un lenguaje popular es PHP.

### Generación de páginas web dinámicas del lado del cliente

Para responder a movimientos del mouse o interactuar directo con el usuario se necesitan secuencias de comando incrustadas en páginas HTML que se ejecuten en la máquina del cliente. Estas secuencias son posibles gracias a la etiqueta *<script>*. El lenguaje más popular es JavaScript.

Aunque PHP y JavaScript se ven similares, se procesan de manera muy diferente.

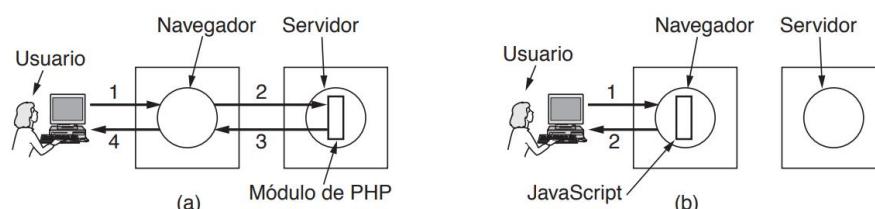


Figura 7-32. (a) Secuencias de comandos del lado servidor con PHP. (b) Secuencias de comandos del lado cliente con JavaScript.

# HTTP: protocolo de transferencia de hipertexto

Protocolo simple de solicitud – respuesta para transportar información entre servidores web y clientes que opera sobre TCP. Especifica qué mensajes pueden enviar los clientes a los servidores y qué respuestas reciben estos mensajes.

## CONEXIONES

La forma más común es una conexión TCP por el puerto 80 del servidor. Como utiliza TCP, ni navegador ni servidor deben manejar congestión, confiabilidad y sobrecarga.

- HTTP1.0 establece conexión envía una solicitud y envía una respuesta. Luego libera.
- HTTP1.1 implementa conexión persistente. Luego de establecer la conexión puede enviarse una solicitud, recibir la respuesta y enviar solicitudes adicionales para obtener respuestas adicionales. Reutiliza la conexión, pero sigue sin ser orientado a sesión, solo utiliza una conexión por más tiempo para mejorar la eficiencia de otros elementos. Permite canalizar solicitudes.
- HTTP2.0 si es orientado a sesión.

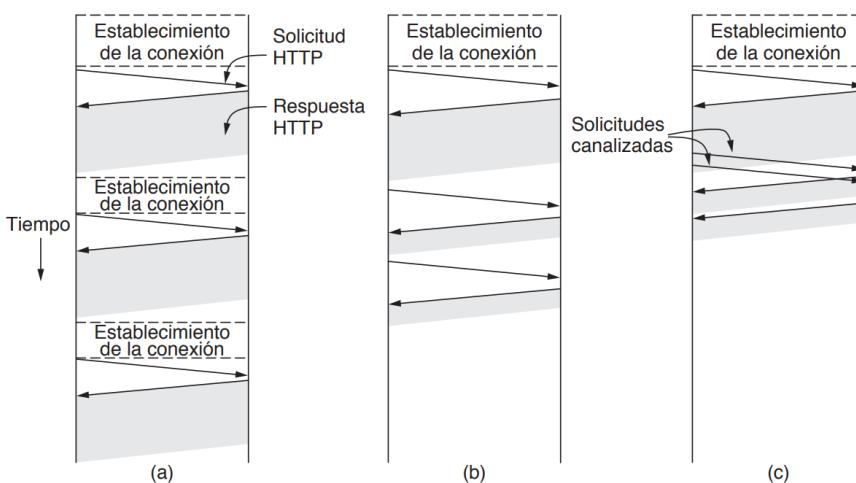


Figura 7-36. HTTP con (a) varias conexiones y solicitudes secuenciales. (b) Una conexión persistente y solicitudes secuenciales. (c) Una conexión persistente y solicitudes canalizadas.

## MÉTODOS

HTTP se diseñó para web pero se ha vuelto más general, por lo que soporta otras operaciones llamadas métodos. Una solicitud consiste en una o más líneas de texto ASCII. La primer palabra de la primera línea es el método solicitado.

Método	Descripción
GET	Ler página web
HEAD	Ler encabezado de página web
POST	Adjuntar página web
PUT	Almacenar página web
DELETE	Eliminar página web
TRACE	Repetir solicitud entrante
CONNECT	Conectarse a través de un proxy
OPTIONS	Consultar las opciones para una página

Cada solicitud obtiene una respuesta con una línea de estado y posiblemente información adicional.

## ENCABEZADOS DE MENSAJE

A la línea de solicitud le pueden seguir líneas con más información denominadas encabezados de solicitud. El encabezado **User – Agent** permite que el cliente informe al servidor la implementación de

su navegador así el servidor puede ajustar sus respuestas. Los 4 encabezados Accept indican al servidor lo que el cliente está dispuesto a aceptar. **Authorization** es necesario para páginas protegidas.

## ALMACENAMIENTO EN CACHÉ

Da la ventaja de no tener que repetir la transferencia de una página si está en caché. La desventaja es que el navegador debe guardar las páginas, por lo que debe determinarse si una copia de una página en caché es la misma que se obtendrá del servidor. Para solucionarlo HTTP utilizó 2 esquemas:

- Validación de páginas:** la determinación de la validez de la página se logra mediante el encabezado *Expires*, obtenido cuando se obtuvo la página por primera vez junto con la fecha y hora.
- Preguntar al servidor si la copia en caché sigue siendo válida:** mediante un GET condicional se realiza una solicitud al servidor. Si este sabe que la copia es válida envía una respuesta corta para decirlo. En caso contrario, envía la respuesta completa.

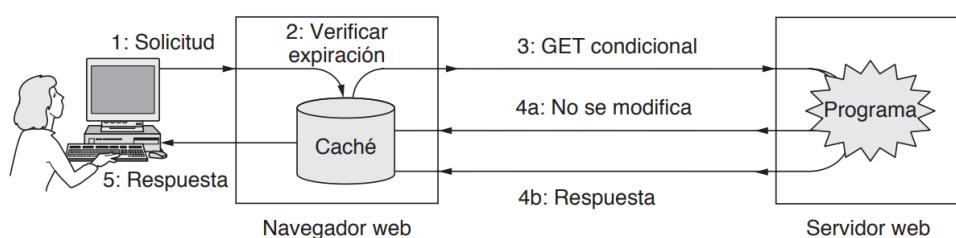
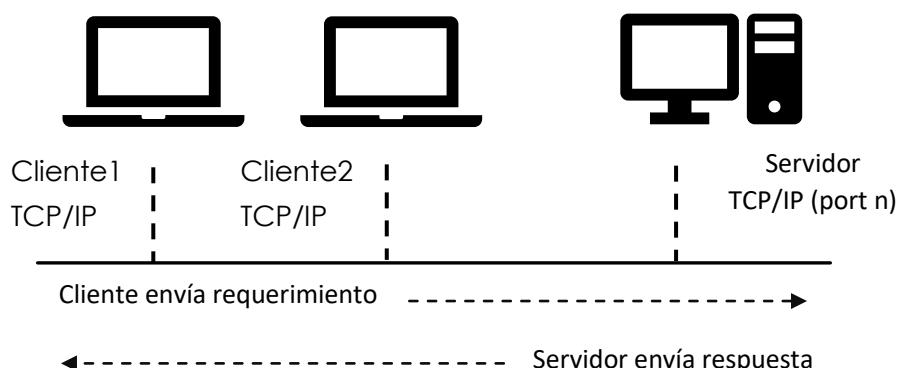


Figura 7-40. Almacenamiento en caché en HTTP.

## MODELO CLIENTE/SERVIDOR

Se divide el trabajo entre máquinas donde unas toman el rol de servidores y otras el de clientes. Cliente y servidor son aplicaciones, las podemos definir como:

- **Servidor:** aplicación accesible a través de la red que cuenta con información a la espera de requerimientos de los clientes. Espera requerimientos, procesa y responde. Siempre está en ejecución esperando solicitudes. Como existen  $2^{16}$  puertos (65536) debe optimizarse su asignación. La mejor solución es asignar puertos según el servicio prestado (puertos bien conocidos). El servidor es una aplicación compleja, ya que debe evaluar autorización de clientes, acceder a bases de datos, atender a más de un cliente a la vez, etc.
- **Cliente:** aplicación que solicita información a un servidor y espera su respuesta. Se ejecuta solo para hacer requerimientos. Es una aplicación menos compleja que el servidor. Utiliza un puerto aleatorio en cada conexión. Como es quién inicia la conexión, debe conocer IP y puerto del servidor (nombre y servicio).



El cliente hace una llamada a sistema e inicia un requerimiento hacia el servidor. Una aplicación en servidor recibe los datos del cliente. Los datos de cliente se comunican con los del proceso servidor. El cliente inicia la comunicación y el servidor responde. Para la comunicación utilizan un protocolo, el cual es un acuerdo entre ambas aplicaciones sobre cómo realizar la comunicación.

## Socket

Los programas para utilizar la red utilizan la interfaz de socket. Como los protocolos de red incluidos en el SO y las aplicaciones por fuera del mismo (parte de usuario), se necesitó de una API para acceder a los protocolos de red desde capa de usuario. El **Socket** es un IPC para procesos no relacionados, sin estándares definidos por las RFCs. El estándar “de facto” se denomina Socket de Berkeley y es implementado en UNIX BSD, luego adaptado a todos los SO.

En la capa superior se encuentra la aplicación que corresponde al modo usuario. Las capas inferiores corresponden al modo SO.

La aplicación puede decidir usar TCP o UDP en la capa de transporte. Luego en la de red utiliza el protocolo IP.

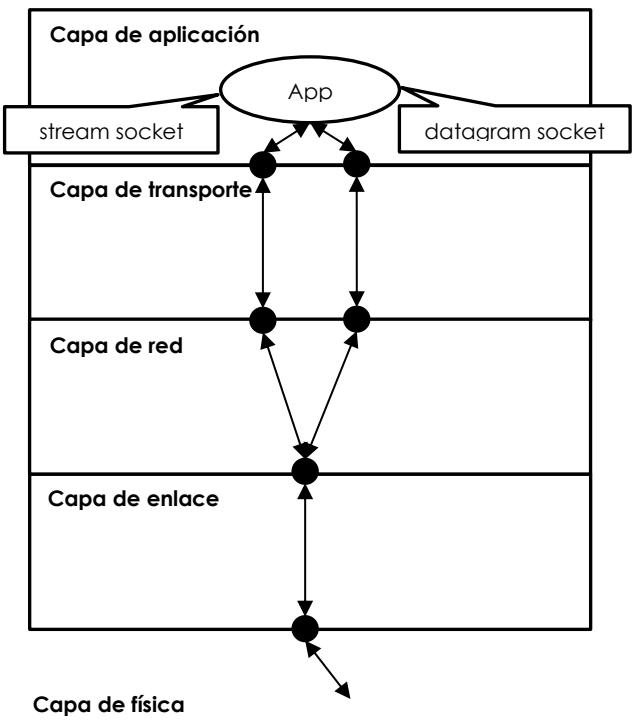
Si la aplicación utiliza TCP se trata de un socket stream, similar a una tubería. En caso contrario se utiliza un socket datagram.

Se creó en UNIX, por lo que sigue este paradigma:

- Abre un descriptor de archivo, lee o escribe y lo cierra.
- Usa abstracción de descriptor de archivo.

En la interfaz de socket aparecen además los términos:

- **Endpoint:** se refiere al par dirección y puerto al que va a conectarse el socket.
- **Socket pasivo:** socket asociado a un solo endpoint (local). Un ejemplo es un servidor en modo listen.
- **Socket activo:** asociado a dos endpoint (local y remoto). Usado por clientes, ya que de principio conocen su endpoint local y el remoto al que quieren conectarse (servidor).



## Llamadas a sistemas de socket

### SOCKET STREAM

#### Creación del socket

La llamada a sistema `socket()` crea un socket devolviendo su descriptor de archivo. Ambas aplicaciones deben crearlo para comunicarse.

```
#include <sys/socket.h>
fd = socket(Domain, Type, protocol);
```

- **Domain:** dominio de la comunicación.
  - AF\_UNIX: aplicaciones locales. Estructura de dirección sockaddr\_un.
  - AF\_INET: aplicaciones conectadas por IPv4. Estructura de dirección sockaddr\_in.
  - AF\_INET6: aplicaciones conectadas por IPv6. Estructura de dirección sockaddr\_in6.
- **Type:** tipo de socket.
  - SOCK\_STREAM
  - SOCK\_DGRAM
- **Protocol:** siempre 0 (protocolo por defecto para la familia elegida).

### Vinculación del socket a un endpoint

La llamada `bind()` asocia una dirección a un socket existente. Es llamada por el servidor, el cual debe asociarla a una dirección conocida para que los clientes puedan conectarse.

```
#include <sys/socket.h>
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

Devuelve 0 si tuvo éxito y -1 en caso de error. Los argumentos son:

- `sockfd`: descriptor del socket previamente creado.
- `sockaddr`: puntero a estructura que da la dirección a la que el socket se va a enlazar.
- `addrlen`: tamaño de la estructura.

La familia de protocolos TCP/IP utiliza la siguiente estructura de direcciones:

```
struct_sockaddr_in{
    char sin_len          // longitud total de la dirección
    char sin_family       // familia de la dirección (AF_INET)
    short sin_port        // número de puerto de protocolo
    struct_in_addr sin_addr // dirección IP del host
}
```

### Escuchar conexiones entrantes

La llamada `listen()` marca al socket stream `sockfd` como pasivo. Este posteriormente se usará para aceptar conexiones entrantes, por lo que también es una llamada realizada por el servidor.

```
#include <sys/socket.h>
int listen(int sockfd, int backlog);
```

Devuelve 0 si tuvo éxito y -1 en caso de error. El argumento `backlog` limita el número de conexiones pendientes del servidor.

### Aceptar conexión entrante

La llamada `accept()` acepta conexiones entrantes al socket `sockfd`. La llamada bloquea al proceso hasta que se recibe una solicitud de conexión, por lo que se ejecuta en servidor.

```
#include <sys/socket.h>
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

Devuelve un descriptor de socket activo en caso de éxito y -1 en caso de error. Los argumentos son:

- sockfd: descriptor del socket creado y en modo pasivo.
- \*addr: puntero a estructura dirección especificada en bind().
- \*addrlen: longitud del argumento anterior.

### Conección cliente a servidor

La llamada connect() conecta un socket activo al socket de escucha pasivo del servidor.

```
#include <sys/socket.h>
int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);
```

Devuelve 0 si tuvo éxito y -1 en caso de error. Sus argumentos son:

- sockfd: descriptor del socket cliente que va a conectarse al socket del servidor.
- \*serv\_addr: puntero a endpoint del socket de servidor.
- addrlen: longitud del argumento anterior.

Si el intento de conexión falla, para volver a intentar la conexión debe cerrarse el socket, crear uno nuevo y volver a intentar.

### Cierre de conexión

La manera usual es mediante la llamada close().

```
#include <sys/socket.h>
int close(int sockfd);
```

Devuelve -1 en caso de error.

## SOCKETS DATAGRAM

### Intercambio de datagramas

La llamada recvfrom() recibe datagramas de un socket datagram.

```
#include <sys/socket.h>
ssize_t recvfrom(int sockfd, void *buffer,
                 size_t length, int flags,
                 struct sockaddr *src_addr, socklen_t *addrlen);
```

Devuelve el número de bytes leídos y -1 en caso de error. Sus argumentos son:

- sockfd: descriptor del socket local.
- buffer: buffer donde se almacenarán los bytes leídos.
- length: longitud del buffer.
- flags: mascara de bits de control.
- src\_addr y addrlen: dirección y longitud de dirección del endpoint remoto del que se recibe.

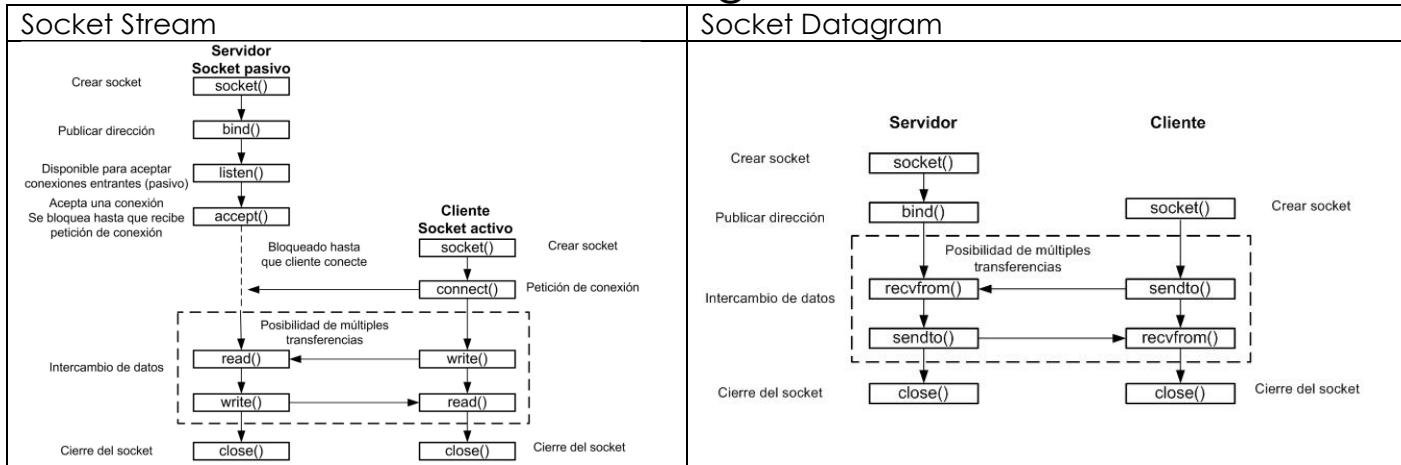
La llamada sendto() envía datagramas a un socket datagram remoto.

```
#include <sys/socket.h>
ssize_t sendto(int sockfd, const void *buffer, size_t length,
               int flags, const struct sockaddr *dest_addr, socklen_t addrlen);
```

Devuelve el número de bytes enviados y -1 en caso de error. Sus argumentos son:

- **sockfd**: descriptor del socket local.
- **buffer**: buffer donde se almacenan los bytes a enviar (mensaje a enviar).
- **length**: longitud del mensaje a enviar.
- **flags**: mascara de bits de control.
- **dest\_addr** y **addrlen**: dirección y longitud de dirección del endpoint remoto al que se envía.

## Socket Stream vs. Socket Datagram



**localhost**: nombre reservado en los host para referenciarse a sí mismos. Es habitualmente traducido como la dirección IP de loopback 127.0.0.1 en IPv4.

## Apéndice: estructuras de direcciones en sockets

```
/*Estructura usada por el kernel para almacenar la mayoría de las direcciones*/
struct sockaddr{
    u_char sa_len;                                // longitud total
    u_char sa_family;                             // familia de direcciones (dominio)
    char sa_data[14];                            // valor de dirección
};


```

```
/*Dirección de socket, estilo internet*/
struct sockaddr_in {
    u_char sin_len;
    u_char sin_family;
    u_short sin_port;
    struct in_addr sin_addr;
    char sin_zero[8];
};


```

```
/*Direcciones de internet, estructura histórica*/
struct in_addr{
    u_int32_t s_addr;
};


```