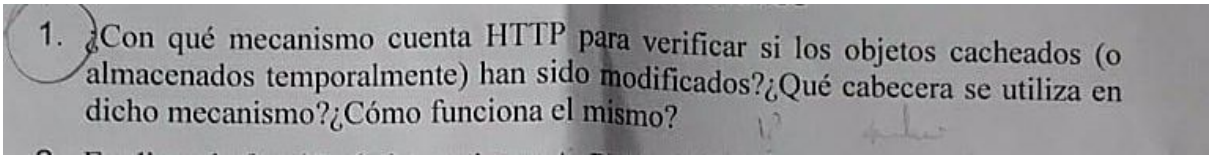
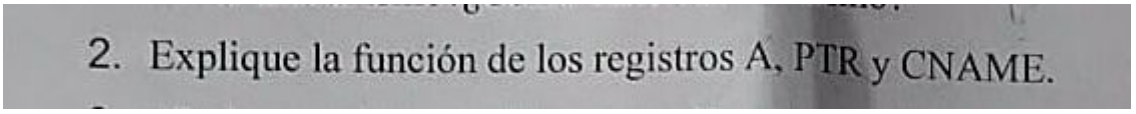
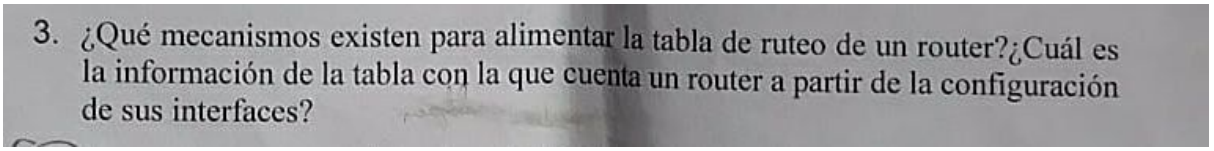
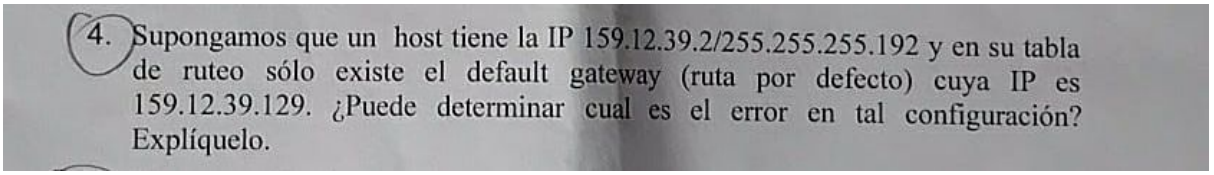
Preguntas de finales

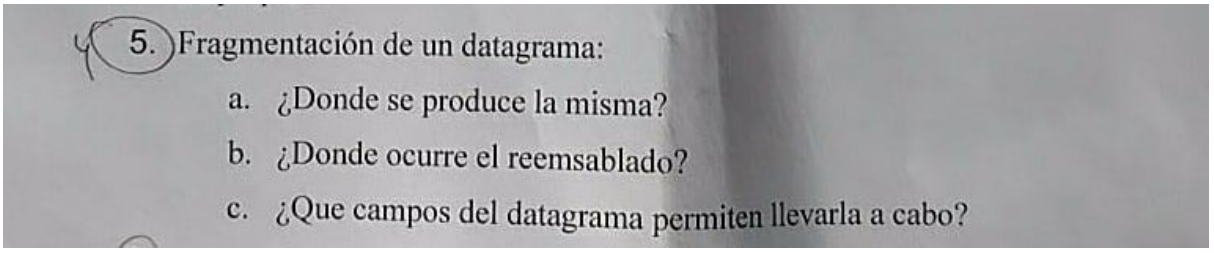
1. 

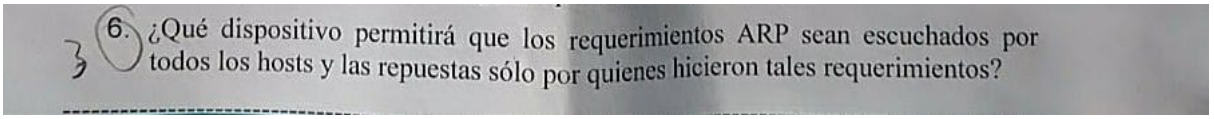
2. 

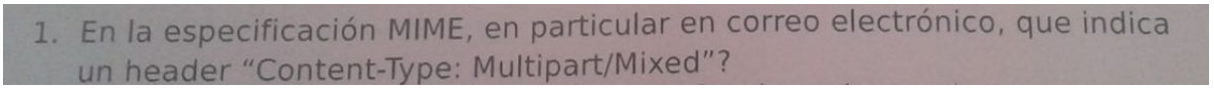
3. 

4. 

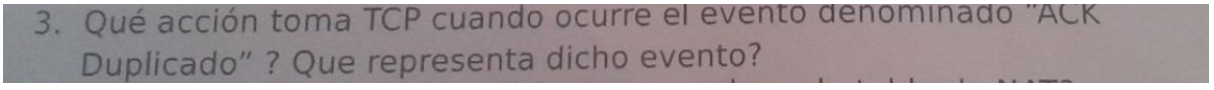
Siendo 192 la máscara es /26 por lo que la cantidad de hosts es 2^6 -2 = 62. El problema de configuración es que el default gateway está en otra red

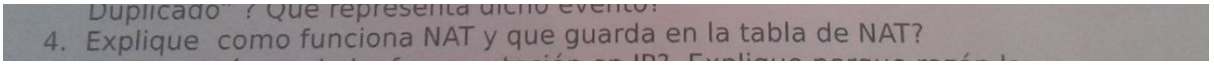
5. 

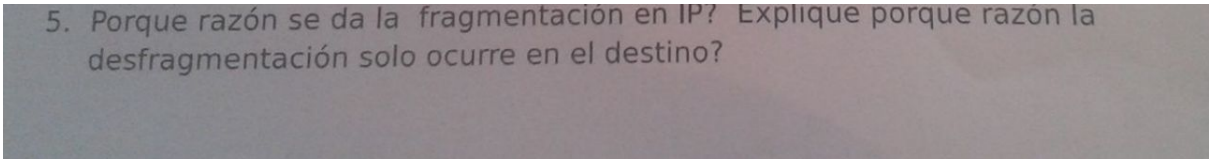
6. 

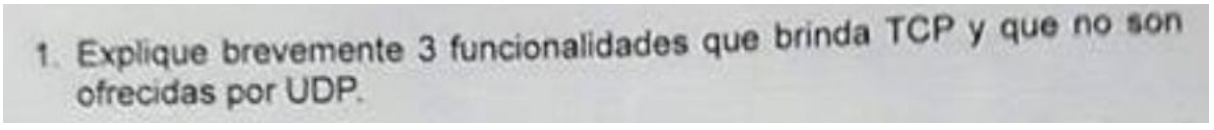
7. 

8. 

9. 

10. 

11. 

12. 

13. 2. Explique cómo se calcula el checksum en UDP y cómo se utiliza el mismo.

14. a) Mencione cuál es la función del protocolo SMTP.

15. b) ¿Qué tipo de conexiones utiliza SMTP? Justifique su respuesta. ¿Qué versión de HTTP trabaja con ese tipo de conexiones?

16. d) ¿Qué otro tipo de protocolo de capa de aplicación se necesita para que el mensaje transmitido sea leído por el usuario "destino". ¿Cuál es su función? Mencione 2 de estos protocolos y explique brevemente las diferencias entre ellos.

17. **Capa de enlace:**

Dada la siguiente afirmación, Indique si la misma es verdadera o falsa. **Justifique** su respuesta.

"En una red local donde todos los dispositivos son Switches y las tablas de switching están vacías, todas las tramas que envíe un host a otro serán vistas por el resto de los hosts de la red".

18. 1. Definir dominios de colisión y dominios de broadcast. Describa que dispositivos tienen la capacidad de dividirlos. Una VLAN (LAN Virtual) que tipo de dominio separa?

19. que tipo de dominio separa?
2. Mencione los 2 eventos que determinan que la ventana de congestión se

20.

21. achique. A que valor la lleva cada uno?
3. Describa el algoritmo que usa Ethernet para volver a intentar transmitir después de una colisión.

22. 3. Describa el algoritmo que usa Ethernet para volver a intentar transmitir después de una colisión.
4. Es posible que un segmento que lleva el bit "S" (Sync) seteado lleve datos válidos? Justifique

23. datos válidos? Justifique
5. Que registros del DNS debe recuperar un servidor de mail que va a conectase a un servidor de mail para el envió de un mensaje? Justifique para que los utiliza.

1 - Fragmentación IP ¿Puede un router detectar algún fragmento faltante? Justifique

2 - Era un ejercicio práctico sobre ARP, pedía la consulta ARP para un host que le quería enviar datos a uno que estaba en

otra red, por ejemplo un Host tiene la IP 192.168.1.5/24 con la MAC AA:BB:AA:BB:AA:BB con puerta de enlace predeterminada

192.168.1.1/24 MAC 1A:22:33:44:55:66 y le quiere enviar un paquete a otro host con IP 192.168.5.2 con MAC 2A:BB:4C:5E:1B:5F.

¿Cómo será la consulta ARP?

3 - ¿Qué significa si un paquete tiene el flag RST activado?

4 - ¿Qué es DHCP? Explique como funciona

5 - ¿Cómo detecta un router un error en un datagrama? ¿Que acciones realiza?

Resumen

Introducción:

Capas:

1. Aplicación: Establece como la aplicación debe estructurarse en los distintos sistemas terminales. PDU: Mensaje.
2. Transporte: Proporciona una *comunicación lógica* entre procesos. PDU: Segmento
3. Red: Proporciona una comunicación lógica entre hosts no fiable (best effort). PDU: datagrama.
4. Enlace: Mover un datagrama desde un nodo a otro adyacente sobre un solo enlace de comunicación. PDU: trama.

Capítulo 2: Capa de aplicación

HTTP
protocolo de capa de aplicación de la web. Se implementa en dos programas (cliente y servidor) que se ejecutan en sistemas terminales diferentes. Se comunican entre sí

intercambiando mensajes HTTP. HTTP define la estructura de estos mensajes y como el cliente y el servidor los intercambian.

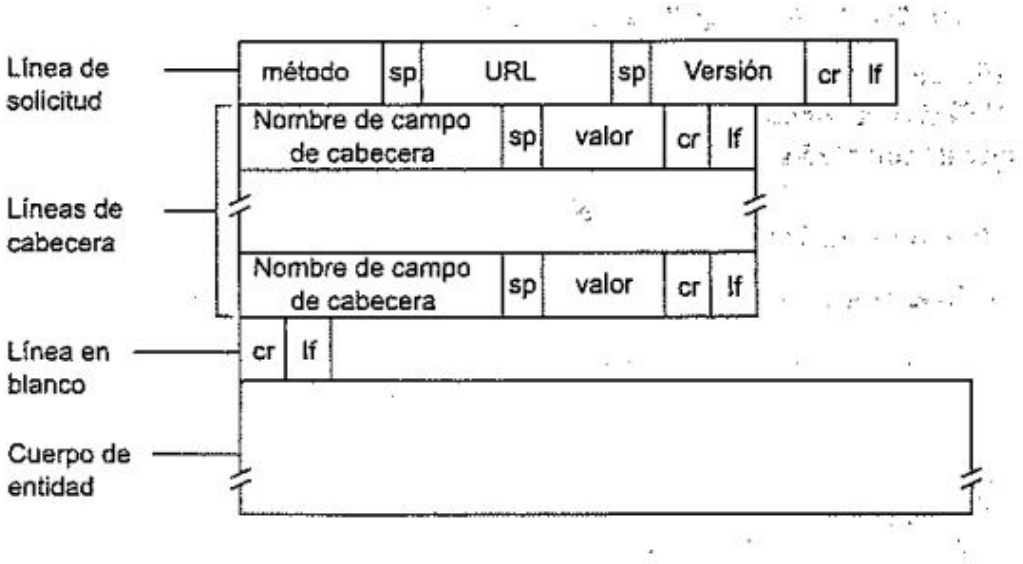
HTTP utiliza TCP como su protocolo de transporte subyacente.

Es un protocolo sin estado, esto quiere decir que por ejemplo si un determinado cliente pide el mismo objeto dos veces en un espacio de tiempo de unos pocos segundos, el servidor no responde diciendo que acaba de servir dicho objeto al cliente; en su lugar, el servidor reenvía el objeto, ya que “ha olvidado por completo” que ya lo había enviado anteriormente.

Conexiones persistentes:

- HTTP 1.0: las conexiones no son persistentes por default, se pueden solicitar de forma explícita *Connection: Keep-Alive*
- HTTP 1.1: conexiones persistentes por default.

Formato de los mensajes Http:



el campo *Cuerpo de entidad* (body) queda vacío cuando se utiliza método GET pero no cuando se utiliza el método POST.

Estados HTTP:

- 1xx: Respuestas informativas
    - 100 Continue: El navegador puede continuar realizando su petición (se utiliza para indicar que la primera parte de la petición del navegador se ha recibido correctamente).
- 2xx: Peticiones correctas:
    - 200 OK: La solicitud se ha ejecutado con éxito y se ha devuelto la información en el mensaje de respuesta.
- 3xx: Redirecciones:
    - 301 Moved Permanently: El objeto solicitado ha sido movido de forma permanente; el nuevo URL se especifica en la línea de cabecera *Location:* del mensaje de respuesta.

    - 304 Not Modified: Indica que la petición a la URL no ha sido modificada desde que fue requerida por última vez. Típicamente, el cliente HTTP provee un encabezado como If-Modified-Since para indicar una fecha y hora contra la cual el servidor pueda comparar. El uso de este encabezado ahorra ancho de banda y reprocesamiento tanto del servidor como del cliente.
- 4xx Errores del cliente:
    - 400 Bad request: código de error genérico que indica que la solicitud no ha sido comprendida por el servidor.
    - 404 Not found: el documento solicitado no existe en el servidor.
- 5xx Errores de servidor
    - 505 HTTP version not supported: la versión de protocolo HTTP solicitada no es soportada por el servidor.

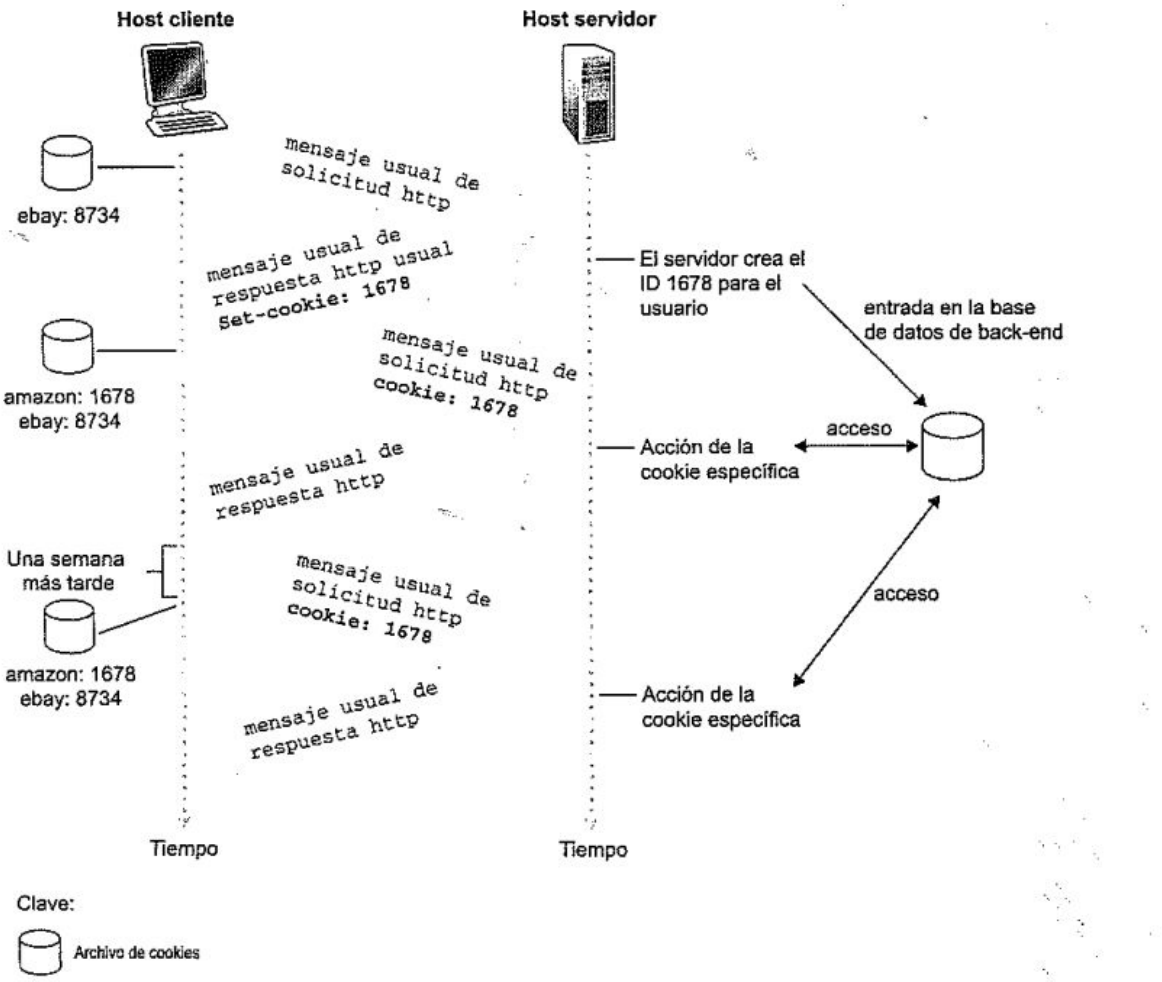
Cookies:
HTTP es un protocolo sin estado, pero en ocasiones es deseable poder identificar a los usuarios, por diferentes motivos:

- para restringir el acceso a usuarios no autorizados.
- para servir contenido en función de la identidad del usuario.

La tecnología de las cookies utilizan cuatro componentes

1. Una línea de cabecera de la cookie en el mensaje de respuesta HTTP. (línea de cabecera *Set-cookie:* valor
2. Una línea de cabecera de la cookie en el mensaje de solicitud HTTP.
3. El archivo de cookies almacenado en el sistema terminal del usuario y gestionado por el navegador del usuario.
4. una base de datos *back-end* en el sitio web.

Ejemplo:



Supongamos que Susana accede siempre a la web utilizando siempre el mismo navegador en la PC de su casa:

- Navegador Susana entra a amazon.com por primera vez (suponiendo que ya había entrado a ebay.com).
- Servidor Amazon crea un número de identificación único y crea una entrada en su base de datos *back-end* que está indexada por el número de identificación.
- Servidor Amazon responde entonces al navegador de Susana, incluyendo en la cabecera HTTP una línea de cabecera *Set-Cookie:*, que contiene el número de identificación. Por ejemplo: *Set-cookie: 1678*.
- Navegador Susana al ver la cabecera *Set-cookie* añade en su archivo especial de cookies (que ya tenía una entrada para Ebay). Esta línea incluye el nombre de host del servidor y el número de identificación de la cabecera.

A medida que Susana continúe navegando por Amazon cada vez que solicite una página web su navegador consulta el archivo de cookies, extrae su número de identificación para ese sitio y añade una cabecera de cookie que incluye el número de identificación. De esta forma, el servidor Amazon puede seguir la actividad de Susana en su sitio.

Caché web:
Un caché web también denominado servidor proxy, es una entidad de red que satisface solicitudes HTTP en nombre de un servidor web de origen. La caché web dispone de su propio almacenamiento en disco y mantiene en él copias de los objetos solicitados recientemente. El navegador de un usuario se puede configurar de modo que todas sus solicitudes HTTP se dirijan en primer lugar a la caché web.
Ejemplo: un navegador solicita el objeto *http://www.unaescuela.edu.ar/campus.gif*

1. El navegador establece una conexión TCP con la caché web y envía una solicitud HTTP para el objeto a la caché web.
2. La caché web comprueba si tiene una copia del objeto almacenado localmente. Si la tiene la caché devuelve el objeto dentro de un mensaje HTTP al navegador del cliente.
3. Si la caché web no tiene el objeto, abre una conexión TCP con el servidor de origen, es decir con *www.unaescuela.edu.ar*. La caché web envía entonces una solicitud HTTP para obtener el objeto a través de la conexión TCP caché-servidor. Después de recibir esta solicitud, el servidor de origen envía el objeto dentro de un mensaje de respuesta HTTP a la caché web.
4. Cuando la caché web recibe el objeto, almacena una copia en su dispositivo de almacenamiento local y envía una copia, dentro de un mensaje HTTP, al navegador del cliente (a través de la conexión TCP existente entre el navegador del cliente y la caché web).

Una caché es a la vez un servidor y un cliente. Cuando recibe solicitudes de y envía respuestas a un navegador se comporta como un servidor. Cuando envía solicitudes a y recibe respuestas de un servidor de origen, entonces se comporta como un cliente.

Ventajas caché web:

- Reducir tiempo de respuesta a la solicitud de clientes. (si existe una conexión de alta velocidad entre el cliente y la caché)
- Reducir tráfico en el enlace a internet, lo que lleva a no tener que mejorar el ancho de banda tan rápidamente (reducción de costes)

GET condicional:
Mecanismo para verificar que los objetos almacenados en caché están actualizados.

Para verificar si un recurso fue modificado en el servidor, el cliente envía un GET con las cabeceras *If-modified-since* (con el valor de fecha de última actualización de objeto en caché) o *E-Tag* (con el valor del último *E-Tag* recibido del servidor).
Cuando un servidor recibe una solicitud con la cabecera *If-Modified-Since*:

- Chequea si la fecha de modificación del recurso es posterior a la fecha recibida en la cabecera:
    - Si es posterior: devuelve el recurso completo (header y datos)
    - Si no es posterior (el archivo no fue modificado): devuelve solo el header, dado que el cliente ya cuenta con el recurso actualizado.

FTP:

Se utiliza para la transferencia de archivos a y desde un host remoto. Cuenta con autorización mediante usuario y contraseña.

y HTTP tienen muchas características en común, por ejemplo ambos se ejecutan sobre TCP. Sin embargo también cuentan con algunas diferencias importantes, por ejemplo, FTP utiliza dos conexiones TCP paralelas para transferir un archivo:

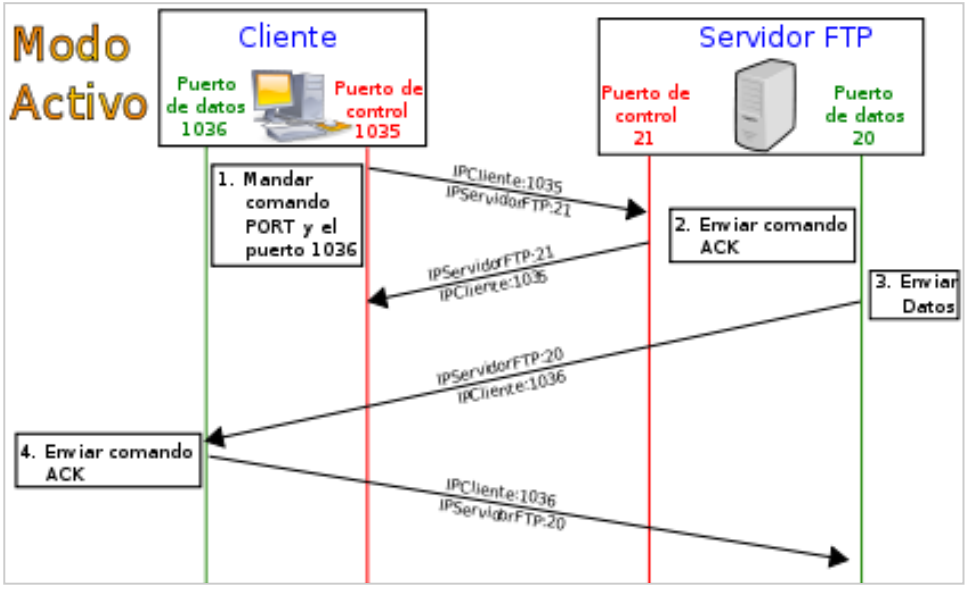
- Conexión de control: se utiliza para enviar información de control entre los dos hosts, como por ejemplo la identidad del usuario, contraseña, comandos para modificar el directorio remoto y comandos para “introducir” (PUT) y “extraer” (GET) archivos. La conexión de datos. Es persistente (se utiliza una para toda la sesión FTP)
- Conexión de datos: se utiliza para enviar un archivo, no son persistentes (se utiliza una por archivo).

Dado que FTP utiliza una conexión de control separada, se dice que FTP envía su información de control “fuera de banda”.

Cuando un usuario FTP inicia una sesión FTP con un host remoto, el lado del cliente de FTP (usuario) establece en primer lugar una conexión de control TCP con el lado del servidor (host remoto) en el puerto del servidor número 21. El lado del cliente de FTP envía la identificación y la contraseña del usuario a través de la conexión de control. El lado del cliente FTP también envía, a través de la conexión de control, comandos para modificar el directorio remoto. Cuando el lado del servidor recibe un comando para realizar una transferencia de archivo a través de la conexión de control (bien a, o desde el host remoto), el lado del servidor inicia una conexión de datos TCP con el lado del cliente.

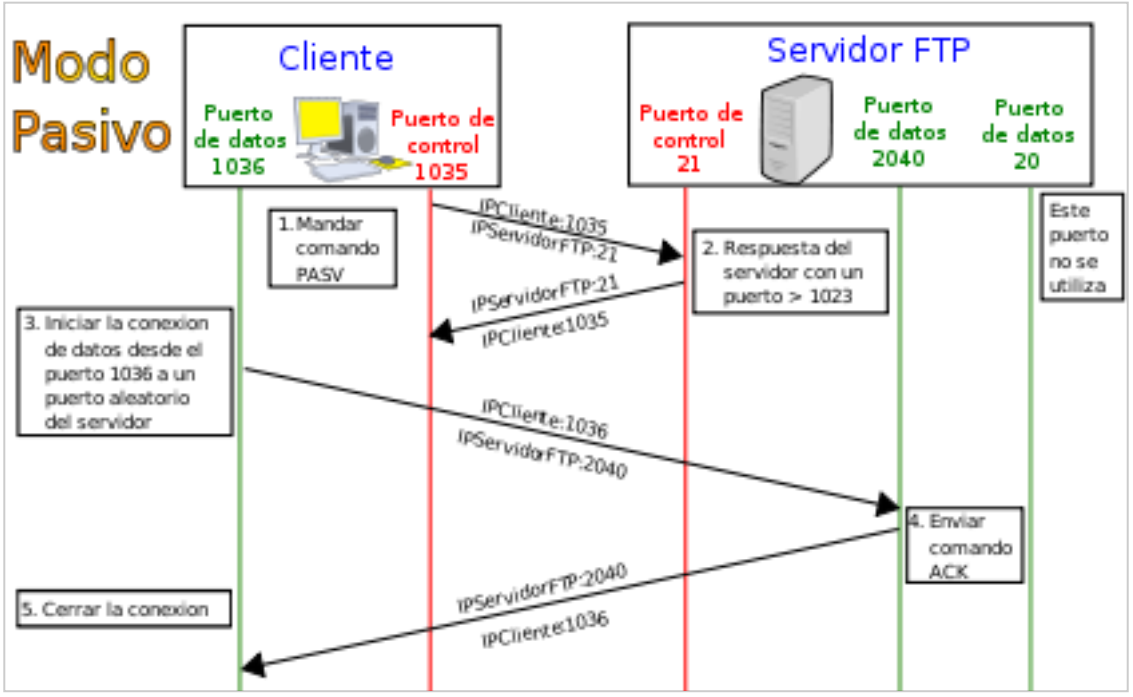
A lo largo de la sesión, el servidor FTP tiene que mantener un **estado** del usuario. En concreto, el servidor debe asociar la conexión de control con una cuenta de usuario específica y debe estar al tanto del directorio actual del usuario cuando éste se mueve por el árbol del directorio remoto. Llevar el control de esta información de estado para cada sesión de usuario activa restringe de forma significativa el número total de sesiones que FTP puede mantener simultáneamente.

Modo activo:



En modo Activo, el servidor siempre crea el canal de datos en su puerto 20, mientras que en el lado del cliente el canal de datos se asocia a un puerto aleatorio mayor que el 1024. Para ello, el cliente manda un comando PORT al servidor por el canal de control indicándole ese número de puerto, de manera que el servidor pueda abrirle una conexión de datos por donde se transferirán los archivos y los listados, en el puerto especificado.
Lo anterior tiene un grave problema de seguridad, y es que la máquina cliente debe estar dispuesta a aceptar cualquier conexión de entrada en un puerto superior al 1024, con los problemas que ello implica si tenemos el equipo conectado a una red insegura como Internet. De hecho, los cortafuegos que se instalen en el equipo para evitar ataques seguramente rechazarán esas conexiones aleatorias. Para solucionar esto se desarrolló el modo pasivo.

Modo pasivo:



Cuando el cliente envía un comando PASV sobre el canal de control, el servidor FTP le indica por el canal de control, el puerto (mayor a 1023 del servidor. Ejemplo:2040) al que debe conectarse el cliente. El cliente inicia una conexión desde el puerto siguiente al puerto de control (Ejemplo: 1036) hacia el puerto del servidor especificado anteriormente (Ejemplo: 2040).
Antes de cada nueva transferencia tanto en el modo Activo como en el Pasivo, el cliente debe enviar otra vez un comando de control (PORT o PASV, según el modo en el que haya conectado), y el servidor recibirá esa conexión de datos en un nuevo puerto aleatorio (si está en modo pasivo) o por el puerto 20 (si está en modo activo). En el protocolo FTP existen 2 tipos de transferencia en ASCII y en binarios.

Correo electrónico:
Tres componentes principales:

1. Agentes de usuario: permiten a los usuarios leer, responder, reenviar y almacenar y componer mensajes
2. Servidores de correo: gestiona y mantiene los mensajes que recibe. Núcleo de la infraestructura de correo electrónico, cada destinatario tiene un buzón de correo ubicado en uno de los servidores de correo.
3. Protocolo simple de transferencia de correo (SMTP): protocolo utilizado para el envío de correo electrónico.

Un mensaje típico inicia su viaje en el agente de usuario del emisor, viaja hacia el servidor de correo del emisor (SMTP) y luego hacia el servidor de correo del destinatario (SMTP), donde es depositado en el buzón del mismo.

SMTP:
Utiliza TCP para transferir el correo en el puerto 25, con conexiones persistentes. Tiene dos lados (cliente y servidor) tanto el lado del cliente SMTP como el lado del servidor se ejecutan en todos los servidores de correo:

- Cuando un servidor de correo envía mensajes de correo a otros servidores de correo, actúa como cliente SMTP.
- Cuando un servidor de correo recibe correos de otros servidores actúa como un servidor SMTP

SMTP es mucho más antiguo que HTTP tiene algunas funcionalidades arcaicas. Por ejemplo restringe el cuerpo (no sólo las cabeceras) de todos los mensajes a formato ASCII de 7 bits. Está restricción tenía sentido a principios de la década de 1980. Pero actualmente causa muchos problemas requiere:

- Los datos binarios multimedia se codifiquen a ASCII antes de ser transmitidos a través de SMTP
- Que los datos sean codificados de ASCII a binarios una vez realizado el transporte SMTP.

SMTP no utiliza servidores de correo intermedios para enviar correo. Si el servidor de destino está fuera de servicio, el servidor de origen conservará el mensaje y lo intentará nuevamente.

¿Por qué los correos no se envían directamente al servidor de correo del destinatario? (Motivo por el cual el correo es enviado desde el agente de usuario al servidor de correo del remitente): Al hacer que se deposite el mensaje primero en servidor de correo del remitente en caso, de que el servidor de destino no esté disponible el servidor del remitente puede realizar reintentos.

Comparación con HTTP:

HTTP		SMTP
	Se utiliza para transferir	

	archivos	
	Emplea conexiones persistentes	
Principalmente protocolo pull (extracción). El cliente inicia la conexión para obtener un recurso		Fundamentalmente un protocolo push (protocolo de inserción). El servidor inicia la conexión para enviar un recurso
Sin restricciones de formato		Formato ASCII 7 bits
Si un documento consta de texto e imágenes encapsula un objeto por solicitud		Incluye siempre todos los objetos del mensaje en un único mensaje.
	Los mensajes incluyen cabeceras	

Protocolos de acceso:

El usuario ejecuta un agente de usuario en el PC local para acceder a su buzón de correo almacenado en un servidor de correo compartido con otros usuarios. Los protocolos disponibles son:

- POP3: se destaca su simpleza y en contrapartida por su funcionalidad limitada. Utiliza una conexión TCP entre el agente de usuario y el servidor de correo en el puerto 110. Una vez establecida la conexión POP3 pasa a través de tres fases:
    - Autorización: el agente de usuario envía un nombre de usuario y una contraseña (en texto legible) para autenticar al usuario.
    - Transacción: el agente de usuario recupera los mensajes; también durante esta fase el agente de usuario puede marcar mensajes para borrado, eliminar marcas de borrado y obtener estadísticas de correo.
    - Actualización: tiene lugar después de que el cliente haya ejecutado el comando *quit*, terminando la sesión POP3; en este instante el servidor de correo borra los mensajes que han sido marcados para borrado.

  En una transacción POP3, el agente de usuario ejecuta comandos y el servidor devuelve para cada comando una respuesta. Existen dos tipos posibles de respuestas: +OK (seguida en ocasiones por una serie de datos servidor-cliente), utilizada por el servidor para indicar que el comando anterior era correcto; y -ERR, utilizada por el servidor para indicar que hubo algún error en el comando anterior.

  Un Agente de usuario que utilice POP3 puede ser configurado para “descargar y borrar” o para “descargar y guardar”
- IMAP: más complejo que POP3, con más funcionalidades. Un servidor IMAP asociará cada mensaje con una carpeta; cuando el mensaje llega al servidor, se asocia con la carpeta INBOX del destinatario, el cual puede entonces pasar el mensaje a una nueva carpeta creada por el usuario, leer el mensaje, borrarlo, etc. El protocolo IMAP proporciona comandos que permiten a los usuarios crear carpetas y

mover los mensajes de una carpeta a otra. IMAP también proporciona comandos que permiten a los usuarios realizar búsquedas en carpetas remotas para localizar mensajes que cumplan unos determinados criterios. A diferencia de POP3, mantiene información acerca del estado a lo largo de las sesiones IMAP, como por ejemplo los nombres de las carpetas y los mensajes asociados a cada una de ellas. Otra importante característica de iMAP es que dispone de comandos que permiten a un agente de usuario obtener partes componentes de los mensajes, por ejemplo sólo la cabecera.

- HTTP: el agente de usuario es un navegador web corriente y el usuario se comunica con su buzón remoto a través de HTTP. se utiliza para obtener mensajes del servidor de correo local y para enviar mensajes hacia el servidor de correo local (en lugar de SMTP). Sin embargo la comunicación entre servidores de correo se sigue realizando mediante SMTP.

Header Content-Transfer-encoding

Muchos tipos de contenido que podrían transportarse útilmente por correo electrónico están representados, en su formato "natural", como caracteres de 8 bits o datos binarios. Tales datos no pueden transmitirse a través de algunos protocolos de transporte. Por ejemplo, RFC 821 restringe los mensajes de correo electrónico a datos US-ASCII de 7 bits con líneas de 1000 caracteres.
Por lo tanto, es necesario definir un mecanismo estándar para volver a codificar dichos datos en un formato de línea de 7 bits. Tales codificaciones serán indicadas por un campo de encabezado "Content-Transfer-Encoding". El campo Content-Transfer-Encoding se usa para indicar el tipo de transformación que se ha utilizado para representar el cuerpo de forma aceptable para el transporte.
El campo Content-Transfer-Encoding está diseñado para especificar una asignación invertible entre la representación "nativa" de un tipo de datos y una representación que puede intercambiarse fácilmente utilizando protocolos de transporte de correo de 7 bits, como los definidos por RFC 821 (SMTP) . Este campo no ha sido definido por ningún estándar previo. El valor del campo es un token único que especifica el tipo de codificación, los posibles valores son:

Content-Transfer-Encoding : = "BASE64" / "QUOTED-PRINTABLE" /
"8BIT" / "7BIT" /
"BINARY" / x-token

La codificación de transferencia de contenido Base64 está diseñada para representar secuencias arbitrarias de octetos en una forma que no es humanamente legible. Los algoritmos de codificación y descodificación son simples, pero los datos codificados son solamente un 33 por ciento más grandes que los datos no codificados.

DNS:
es:

- Una base de datos distribuida implementada en una jerarquía de servidores DNS
- Protocolo de capa de aplicación que permite a los hosts consultar la base de datos distribuida.

Se ejecuta sobre UDP en el puerto 53. Utilizado por otros protocolos como HTTP, SMTP y FTP para traducir los nombres de hosts suministrados por el usuario en direcciones IP.

Servicios adicionales a la traducción de nombres de hosts en direcciones IP:

- Alias de host: un host con nombre "complicado" puede tener uno o más alias. Por ejemplo, un nombre de host como *relay1.west-coast.enterprise.com* podría tener digamos dos alias como *enterprise.com* y *www.enterprise.com*. En este caso, el nombre de host *relay1.west-coast.enterprise.com* se dice que es el nombre de host canónico. Los alias de nombres de host suelen ser más mnemónicos que los nombres canónicos. Una aplicación puede invocar DNS para obtener el nombre de host canónico de un determinado alias, así como la dirección IP del host.
- Alias de servidor de correo: Es deseable que las direcciones de correo electrónico sean mnemónicas (ej: hotmail.com). Sin embargo el nombre de host es más complicado (por ejemplo el de hotmail.com podría ser relay1.west-coast.hotmail.com). Una aplicación de correo puede invocar al servicio de DNS para obtener el nombre de host canónico para un determinado alias, así como la dirección IP del host. De hecho el registro MX permite al servidor de correo y al servidor web de una empresa tener nombres de host (con alias) iguales; por ejemplo, tanto el servidor web como el servidor de correo de una empresa se pueden llamar *enterprise.com*.
- Distribución de carga: DNS también se utiliza para realizar la distribución de carga entre servidores replicados, como los servidores web replicados. Los sitios con una gran carga de trabajo, están replicados en varios servidores, ejecutándose en cada servidor un sistema terminal distinto, y teniendo cada uno una dirección IP diferente. Cuando los clientes realizan una consulta DNS sobre un nombre asignado a un conjunto de direcciones, el servidor responde con el conjunto completo de direcciones IP, pero **rota el orden de las direcciones de respuesta**. Dado que normalmente un cliente envía un mensaje de solicitud HTTP a la dirección IP que aparece en primer lugar dentro del conjunto, la rotación DNS **distribuye el tráfico entre los servidores replicados**. La rotación DNS también se emplea para el correo electrónico.

Cómo funciona DNS:
Utiliza un diseño distribuido, una base de datos jerárquica y distribuida. Las correspondencias de los hosts están repartidas por los servidores DNS.
Existen principalmente tres clases de servidores DNS:

1. Servidores raíz: en internet existen 13 servidores raíz (etiquetados como A hasta M). Cada uno de los 13 servidores es en realidad una agrupación (*cluster*) de servidores replicados, tanto por propósitos de seguridad como de fiabilidad.
2. Servidores DNS de nivel superior (TLD): responsables de los dominios de nivel superior, como por ejemplo: *.com*, *.org*, *.net*, *edu*, y *gov*, y todos los dominios de

nivel superior correspondientes a los diferentes países, como por ejemplo: *uk*, *fr*, *ck*, y *ar*.

3. Servidores autoritativos: Todas las organizaciones que tienen hosts accesibles públicamente a través de Internet deben proporcionar registros DNS accesibles públicamente que establezca la correspondencia entre los nombres de dichos hosts y sus direcciones IP. Un servidor de registro autoritativo de una organización alberga estos registros DNS. Puede ser propio de la organización o pagar para tener esos registros en un servidor DNS autoritativo de algún proveedor de servicios.

Existe otro tipo de servidores DNS llamado **servidor DNS local**, el cual no pertenece estrictamente a la jerarquía de DNS. Cada ISP tiene un servidor DNS local. Cuando un host se conecta a un ISP, este proporciona al host las direcciones IP de uno o más de sus servidores DNS locales. Generalmente un servidor de DNS local de un host se encuentra relativamente "cerca" de ese host. Cuando un host realiza una consulta DNS, ésta se envía al servidor DNS local, que actúa como proxy, reenviando la consulta a la jerarquía de servidores DNS.Normalmente la consulta procedente del host al servidor DNS local es **recursiva** y las consultas que realiza el servidor DNS local son **iterativas**.

Almacenamiento en caché DNS:
En una cadena de consultas, cuando un servidor DNS recibe una respuesta DNS (que contiene por ejemplo la correspondencia entre un nombre de host y una dirección IP), puede almacenar esta información en su memoria local, y si vuelve a recibir la misma consulta podrá responder sin necesidad de realizar otras consultas. Dado que los hosts y las correspondencias entre nombres de host y direcciones IP no son permanentes, los servidores DNS descartan la información almacenada en caché pasado un cierto periodo de tiempo (normalmente, unos dos días). Un servidor de DNS local también puede almacenar en caché las direcciones IP de los servidores TLD, permitiendo así a los servidores DNS locales saltarse a los servidores DNS raíz en una cadena de consultas.

Registros y mensajes DNS:
Un registro de recurso está formado por los siguientes cuatro campos:

(Nombre, Valor, Tipo, TTL)

TTL es el tiempo de vida del registro de recurso; determina cuándo un recurso debería ser eliminado de una caché.

Los tipos de registros DNS son:

- A: *Nombre* es un nombre de host y *Valor* es la IP correspondiente a dicho nombre. Proporciona la correspondencia estándar nombre de host-dirección IP. Por ejemplo, (*relay1.bar.foo.com*, 145.37.93.126, A) es un registro de tipo A.
- PTR: *Nombre* es una IP y *valor* es un nombre de dominio para esa IP.
- NS: *Nombre* es un dominio (como foo.com) y *Valor* es el nombre de host de un servidor de DNS autoritativo que sabe cómo obtener las direcciones IP de los hosts del dominio. Se utiliza para encaminar las consultas DNS a lo largo de la cadena de consultas. Por ejemplo (*foo.com, dns.foo.com,* NS).
- CNAME: *Valor* es un nombre de host canónico correspondiente al alias especificado por *Nombre*. Proporciona a los hosts que hacen consultas el

nombre canónico correspondiente a un nombre de host. Por ejemplo (*foo.com, relay1.bar.foo.com,* CNAME) es un registro CNAME

- MX: *Valor* es el nombre canónico de un servidor de correo que tiene un alias dado por *Nombre*. Por ejemplo (*foo.com*, *mail.bar.foo.com*, MX) es un registro MX. Los registros MX permiten a los nombres de host de los servidores de correo tener alias simples. Utilizando el registro MX una empresa puede tener el mismo alias para su servidor de correo y para uno de sus otros servidor (ej: el servidor web).

Si un servidor DNS es autoritativo para un determinado nombre de host, entonces el servidor DNS contendrá un registro de tipo A para el nombre de host, (Incluso aunque el servidor DNS no sea autoritativo, puede contener un registro de tipo A en su caché). Si un servidor no es autoritativo para un nombre de host, entonces el servidor contendrá un registro de tipo NS para el dominio que incluye el nombre de host; también contendrá un registro de tipo A que proporcione la dirección IP del servidor DNS en el campo *Valor* del registro NS.

Capítulo 3: Capa de transporte

Función:
Proporciona una *comunicación lógica* entre procesos de aplicación que se ejecutan en hosts diferentes. *Comunicación lógica* quiere decir que desde la perspectiva de la aplicación, es como si los hosts que ejecutan los procesos estuvieran conectados directamente (sin preocuparse por los detalles de la infraestructura utilizada para transportar los mensajes)

Los protocolos de la capa de transporte están implementados en los sistemas terminales, pero no en los routers de la red.

El protocolo de la capa de transporte reside EN EL HOST, y lleva los mensajes desde los procesos de la aplicación hasta la frontera de la red (capa de red), y viceversa.

Para las aplicaciones de red puede haber más de un protocolo de transporte disponible. Internet tiene dos protocolos: TCP y UDP. Cada uno proporciona un conjunto diferente de servicios para la capa de transporte a la aplicación que lo haya invocado.

UDP

Sin conexión no fiable.

Los únicos servicios que tiene son:

- entrega de datos proceso a proceso

- comprobación de errores

Ventajas:

- mejor control en la aplicación sobre qué datos se envían y cuándo
- sin establecimiento de la conexión: no añade ningún retardo a causa de esto.
- sin información del estado de conexión: por ello, un servidor dedicado a una aplicación concreta suele poder soportar más clientes activos cuando la app se ejecuta sobre UDP.
- poca sobrecarga debida a la cabecera de los paquetes: la cabecera de los
- segmentos UDP sólo requiere 8 bytes (contra los 20 de TCP).

TCP

Orientado a la conexión fiable.

Ofrece a las aplicaciones varios servicios adicionales, como:

- entrega de datos fiable (usando técnicas de control de flujo, …)
- garantizar el orden de entrega
- control de congestión: evita que cualquier conexión TCP “inunde” un enlace.

Multiplexación / demultiplexación:

Es un servicio necesario en todas las redes de computadoras.

Demultiplexación: entregar los datos contenidos en un segmento de la capa de transporte al socket correcto.

Multiplexación: reunir los fragmentos de datos en el host de origen desde los diferentes sockets, encapsulando cada fragmento de datos con la información de cabecera.

Require que:

- Los sockets tengan identificadores únicos.
- El segmento tenga campos “especiales” que indiquen el socket al que tiene que entregarse el segmento.

Los campos especiales son

- Número puerto origen (usado como dirección de retorno).
- Número de puerto destino.
-

Puerto: número de 16 bits de 0 a 65535. De 0 a 1023 son reservados (son utilizados por algún protocolo de aplicación conocido uso ej 23 DNS 80 Http).

UDP:

Realiza lo mínimo que un protocolo de transporte debe hacer (multiplexación/demultiplexación) más comprobación de errores. Si se elige UDP para el transporte es prácticamente como si la aplicación se conectará directamente con IP.

UDP toma los mensajes procedentes del proceso de aplicación, asocia los campos correspondientes a los números de puerto origen y destino

(multiplexación/demultiplexación), añade dos campos pequeños más y pasa el segmento resultante a la capa de red.

Es posible que una aplicación disponga de un servicio fiable de transferencia de datos utilizando UDP. Esto puede conseguirse si las características de fiabilidad se incorporan a la propia aplicación

Estructura de los segmentos UDP

El host receptor utiliza la suma de comprobación para determinar si se han introducido errores en el segmento. La suma de comprobación también se calcular para unos pocos campos de la cabecera IP.

Suma de comprobación UDP

UDP en el lado del emisor calcula el complemento a 1 de la suma de todas las palabras de 16 bits del segmento, acarreando cualquier desbordamiento obtenido durante la operación de suma sobre el bit de menor peso. Este resultado se almacena en el campo suma de comprobación del segmento UDP. El complemento a 1 se obtiene convirtiendo todos los 0 en 1 y viceversa.
En el receptor se suman todas las palabras de 16 bits del segmento y luego se le suma la *suma de comprobación* (complemento a 1 de la suma). La suma en el receptor tiene que ser = 1111111111111111. Si uno de los bits es 0 entonces sabremos que el paquete contiene errores.

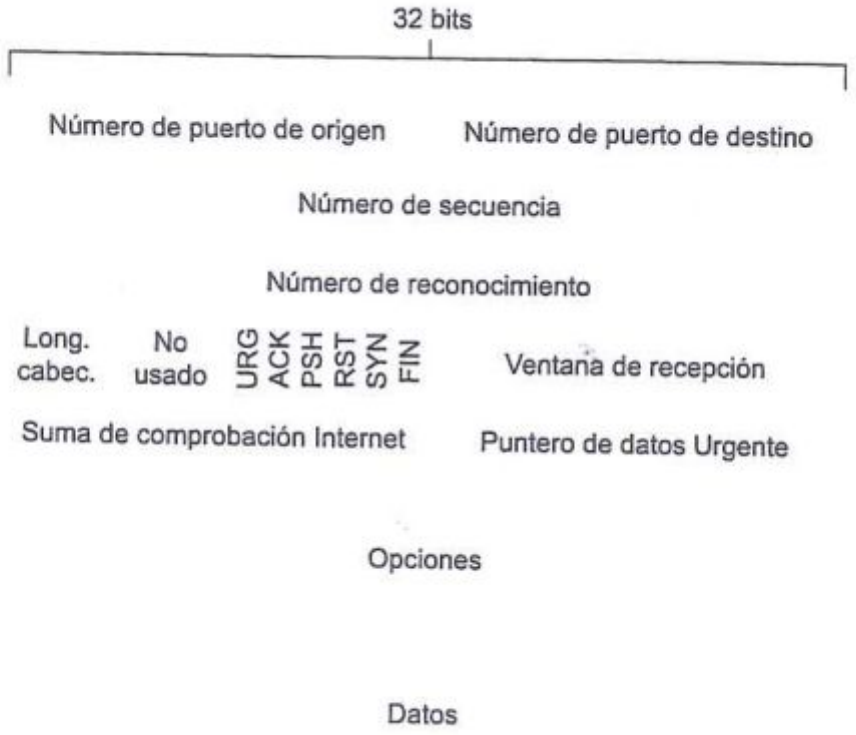
Motivación para comprobación de errores si otras capas también lo implementan: no existe ninguna garantía de que todos los enlaces existentes entre el origen y el destino proporcionen un mecanismo de comprobación de errores. Además incluso si los segmentos se transfieren correctamente a través del enlace, es posible que se introduzcan errores de bit cuando un segmento se almacena en la memoria de un router.

UDP no hace nada para recuperarse el error. Algunas implementaciones descartan el segmento dañado y otras lo pasan a la aplicación con una advertencia.

TCP

Se dice que está orientado a la conexión porque antes de que un proceso de la capa de aplicación pueda comenzar a enviar datos a otro, los dos procesos deben primero "establecer una comunicación" entre ellos. Una conexión TCP proporciona un servicio **full-duplex**.
Una conexión TCP casi siempre es una conexión punto a punto (un emisor y un receptor). El broadcast no es posible en TCP en una sola operación.

Estructura del segmento TCP:



- Número de secuencia y número de reconocimiento: campos de 32 bits. Son utilizados para implementar el servicio de transferencia de datos fiable
- Venta de recepción: 16 bits. Se utiliza para el control de flujo. Se emplea para indicar el número de bytes que un receptor está dispuesto a aceptar
- Longitud de cabecera: 4 bits. Especifica la longitud de cabecera TCP en palabras de 32 bits. La cabecera TCP puede tener una longitud variable a causa del campo opciones (generalmente está vacío, por lo que la longitud de la cabecera generalmente es de 20 bytes)
- Opciones: opcional y de longitud variable. Se utiliza cuando el emisor y receptor negocian el tamaño máximo de segmento (MSS) o como un factor de escala de la ventana en las redes de alta velocidad.
- Indicador: 6 bits.
    - Bit **ACK**: se utiliza para indicar que el valor transportado en el campo de reconocimiento es válido (el segmento contiene un reconocimiento para un segmento que ha sido recibido correctamente).

- **RST**, **SYN** y **FIN**: se utilizan para el establecimiento y cierre de las conexiones.
- **PSH**: indica que el receptor deberá pasar los datos a la capa superior de forma inmediata.
- **URG**: indica que hay datos en ese segmento que la capa de aplicación del lado del emisor ha marcado como urgentes.

Números de secuencia y de reconocimiento:

Los números de secuencia hacen referencia al flujo de bytes transmitidos y no a la serie de segmentos transmitidos. El número de secuencia de un segmento es el número del primer byte del segmento dentro del flujo de bytes.

Números de reconocimiento: es el número del siguiente byte que un host está esperando del otro en una conexión (un cliente de un servidor o un servidor de un cliente).

Establecimiento y cierre de la conexión:

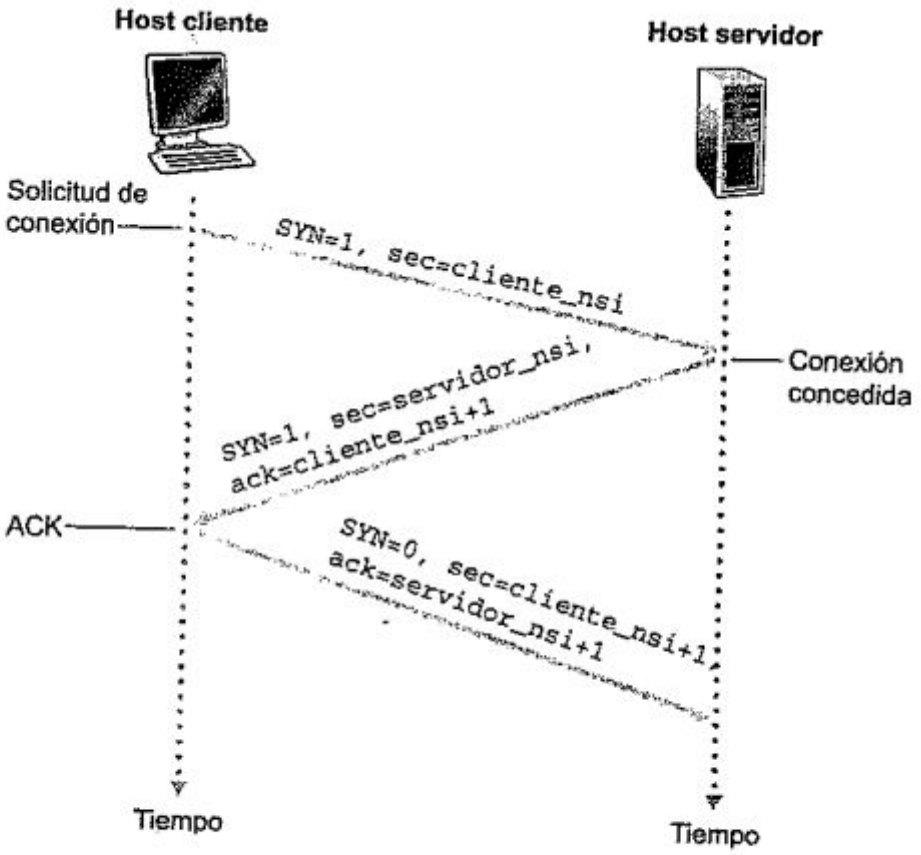


Figura 3.39 • El proceso de acuerdo en tres fases de TCP: intercambio de segmentos.

1. TCP del lado del cliente envía un segmento TCP especial (SYN). Este segmento TCP especial no contiene datos de la capa de aplicación. Pero uno de los bits indicadores de la cabecera del segmento, el bit SYN se pone a 1. Además el cliente selecciona un número de secuencia inicial aleatorio (cliente_nsi) y se lo coloca en el campo número de secuencia del segmento. Este segmento se encapsula dentro de un datagrama IP y se envía al servidor.
2. Una vez que el datagrama IP que contiene el segmento SYN llega al host servidor, el servidor extrae dicho segmento SYN del datagrama, asigna los buffers y variables TCP a la conexión y envía un segmento de conexión concedida al cliente TCP (SYNACK). Este segmento de conexión concedida tampoco contiene datos de la capa de aplicación. Sin embargo, contiene tres fragmentos de información importantes de la cabecera del segmento. El primero, el bit SYN se pone a 1. El segundo, el campo reconocimiento de la cabecera del segmento TCP se hace igual al número de secuencia del cliente + 1 (cliente_nsi + 1). Por último el servidor elige su propio número de secuencia inicial (servidor_nsi) y almacena este valor en el campo número de secuencia de la cabecera del segmento TCP.
3. Al recibir el segmento SYNACK, el cliente también asigna buffers y variables a la conexión. El host cliente envía entonces al servidor otro segmento; este último segmento de conexión concedida del servidor. El bit SYN se pone en 0

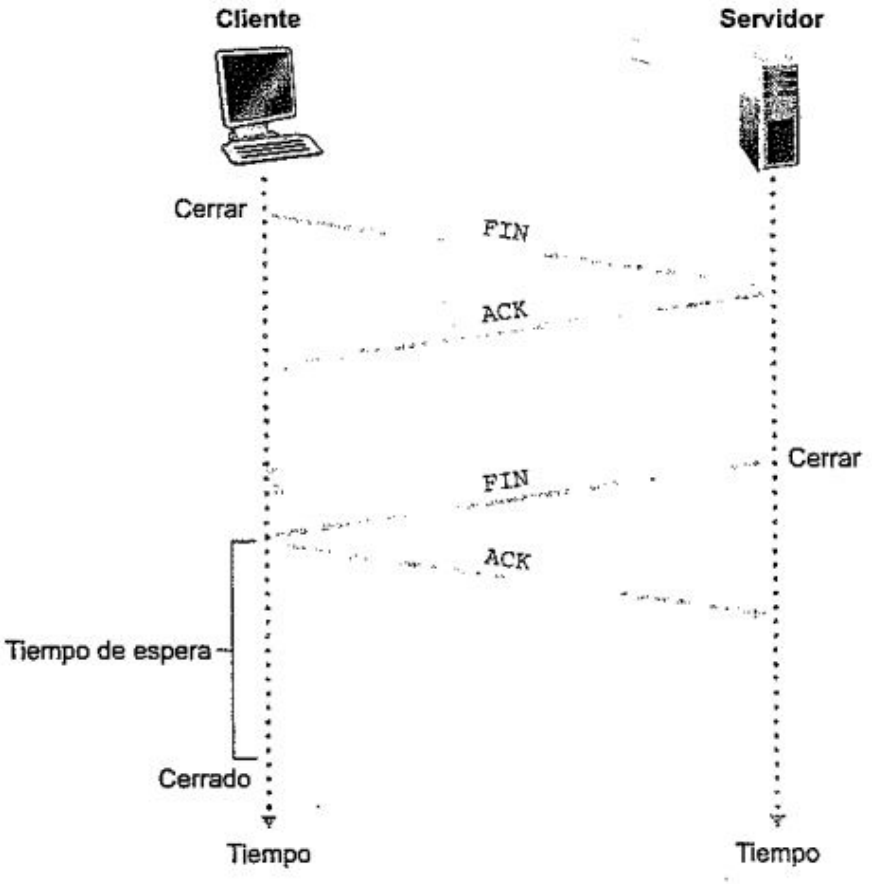


Figura 3.40 • Cierre de una conexión TCP.

Suponiendo que el cliente cierra la conexión los pasos son:

1. El cliente TCP envía un segmento especial TCP al proceso servidor. Contiene un bit indicador en la cabecera del segmento, el bit FIN puesto en 1. Entra en estado FIN_WAIT_1
2. El servidor recibe el segmento y envía al cliente un segmento de reconocimiento. El cliente cuando lo recibe pasa al estado FIN_WAIT_2
3. El servidor envía un segmento con el bit FIN puesto en 1. El cliente cuando lo recibe pasa al estado TIME_WAIT
4. El cliente envía un segmento de reconocimiento.

Control de flujo:
TCP proporciona un **servicio de control de flujo** a sus aplicaciones para eliminar la posibilidad de que el emisor desborde el buffer del receptor. El control de flujo es por tanto un servicio de adaptación de velocidades (adapta la velocidad en la que el emisor está transmitiendo frente a la velocidad en la que la aplicación receptora está leyendo).

El emisor debe mantener una variable conocida como **ventana de recepción**. Informalmente la ventana de recepción, llamada VentanaRecepcion, se emplea para proporcionar al emisor una idea de cuánto espacio libre hay disponible en el buffer del receptor. Puesto que TCP es una conexión full-duplex, el emisor de cada lado de la conexión mantiene una ventana de recepción diferente. La VentanaRecepcion es dinámica

Ej: Host A envía un archivo “grande” a host B a través de una conexión TCP. El host B asigna un buffer de recepción a esta conexión (Buffer-recepcion). De vez en cuando el proceso de aplicación del host B lee el contenido del buffer. El búfer de recepción del host B se llena, de forma que VentanaRecepcion = 0. Después de indicar al host A que VentanaRecepcion = 0, suponga también que B no tiene nada que enviar a A. Considere qué sucede ahora. Cuando el proceso de aplicación en B vacíe el buffer, TCP no enviará segmentos nuevos con el nuevo valor de VentanaRecepcion a A: ya que TCP envía un segmento al host A sólo si tiene datos o si tiene un reconocimiento que enviar. Por lo tanto, el host A no es informado de que se ha abierto un espacio en el almacén de recepción de B (¡el host A está bloqueado y no puede transmitir más datos!) Para resolver este problema, la **especificación de TCP exige que el host A continúe enviando segmentos con un byte de datos aun cuando la ventana de recepción sea cero**. Estos segmentos serán reconocidos por el receptor. Eventualmente, el búfer se irá vaciando, y los reconocimientos contendrán un valor de VentanaRecepcion distinto de cero.

Control de congestión:
TCP para el control de congestión debe utilizar un mecanismo de control de la congestión de extremo a extremo en lugar de uno asistido por la red, ya que la capa IP no proporciona realimentación explícita alguna a los sistemas finales sobre la congestión de la red.

El método empleado por TCP consiste en que cada emisor limite la velocidad a la que transmite el tráfico a través de su conexión en función de la congestión de red percibida. Si un emisor de la red percibe que en la ruta entre él mismo y el destino apenas existe congestión, entonces incrementará su velocidad de transmisión; por el contrario si, el emisor

percibe que existe congestión a lo largo de la ruta, entonces reducirá su velocidad de transmisión. Pero este enfoque deja pendientes tres cuestiones:

1. ¿Cómo hace el emisor TCP para limitar la tasa de emisión a la que envía tráfico por su conexión?
2. ¿Cómo hace un emisor TCP para percibir que existe congestión en el camino entre él y el destino?
3. ¿Qué algoritmo debería utilizar el emisor para cambiar su tasa de emisión en función de la congestión entre extremos percibida?

Cada lado de una conexión TCP está compuesto por un búfer de recepción, un búfer de envío, y varias variables. El mecanismo TCP de control de la congestión hace que cada lado de la conexión mantenga una nueva variable: la ventana de congestión. La ventana de congestión, llamada VentanaCongestion, impone una restricción sobre la tasa a la que el emisor TCP puede enviar tráfico en la red. Concretamente, la cantidad de datos pendientes de reconocimiento en el emisor no puede exceder del mínimo entre VentanaCongestion y VentanaRecepcion, esto es:

UltimoByteLeido - UltimoByteReconocido <= min{VentanaCongestion,VentanaRecepcion}

Para poder centramos en el control de la congestión (como opuesto al control de flujo), vamos, por lo tanto, a asumir que el búfer de recepción es tan grande que la restricción asociada a la ventana de recepción puede ser ignorada; por ello, la cantidad de datos no reconocidos en el emisor está limitada únicamente por VentanaCongestion.
La restricción anterior restricción limita la cantidad de datos no reconocidos en el emisor, y por lo tanto limita la tasa de envío del emisor. Para ver esto, considere una conexión en la cual las pérdidas de los paquetes y los retardos de transmisión son insignificantes. Entonces, aproximadamente al comienzo de cada tiempo de ida y vuelta (RTT), la anterior restricción permite al emisor enviar VentanaCongestion bytes por la conexión, y al final del RTT el emisor recibe los reconocimientos de los datos. *Por ello, la tasa de emisión del emisor es más o menos de VentanaCongestion/RTT bytes/segundo. Ajustando el valor de VentanaCongestion, el emisor puede, por lo tanto, ajustar la tasa a la que envía datos por su conexión*.
Veamos cómo percibe un emisor TCP que existe congestión en el camino entre sí y el destino. Definamos un “suceso de pérdida” en un emisor TCP como el hecho que se produzca el fin de la temporización, o bien la recepción de tres ACK repetidos. Cuando existe una congestión excesiva, entonces se desbordarán uno (o más) búferes en los routers a lo largo del camino, produciendo datagramas desechados. El datagrama desechado, a su debido tiempo, producirá un evento de pérdida en el emisor (ya sea el fin del tiempo de espera o la recepción de tres ACK repetidos) que será tomado por el emisor como una indicación de la congestión en el camino entre el emisor y el receptor.
Ahora vamos a considerar el mejor de los casos, cuando no existe congestión en la red, es decir, cuando no se producen pérdidas de paquetes. En este caso, el emisor TCP recibirá los paquetes de reconocimiento ACK correspondiente a los segmentos anteriormente no reconocidos. TCP interpretará la llegada de estos paquetes ACK como una indicación de que todo está bien (es decir que los segmentos están siendo entregados correctamente) y empleará esos paquetes de reconocimiento para incrementar el tamaño de la ventana de

congestión. Si la velocidad de llegada de los paquetes ACK es lenta, entonces el tamaño de la ventana de congestión se incrementará a una velocidad relativamente lenta. Por el contrario, si la velocidad de llegada de los paquetes ACK es rápida, entonces la ventana de congestión se incrementará más rápidamente. Dado que TCP utiliza los paquetes de reconocimiento para provocar (o temporizar) sus incrementos del tamaño de la ventana de congestión, se dice que TCP es **auto-temporizado**.
En conclusión TCP determina la velocidad de transmisión utilizando los siguientes principios:

- Un segmento perdido implica congestión y, por tanto, la velocidad del emisor TCP debe reducirse cuando se pierde un segmento.
- Un segmento que ha sido reconocido indica que la red está entregando los segmentos del emisor al receptor y, por tanto, la velocidad de transmisión del emisor puede incrementarse cuando llega un paquete ACK correspondiente a un segmento que todavía no había sido reconocido.
- Tanteo del ancho de banda

Finalmente, estamos en posición de considerar el algoritmo que utiliza el emisor TCP para regular su tasa de envío como una función de la congestión percibida. Este algoritmo es el celebrado algoritmo de control de la congestión TCP. El algoritmo tiene tres componentes principales:

1. **Arranque lento (*slow start*)**: durante esta fase la velocidad de transmisión crece exponencialmente hasta el primer suceso de pérdida de paquete, luego comienza un nuevo proceso de arranque lento pero define una variable *Umbral* = *VentanaCongestion/2* (mitad del tamaño de la ventana de congestión cuando se detectó la pérdida). Está fase puede terminar cuando:
    a. Primer suceso de pérdida (comienza nuevamente).
    b. La ventana de congestión llega al valor *Umbral* (pasa a modo evitación de la congestión).
    c. Si se detectan 3 paquetes ACK duplicados, (retransmisión rápida pasa al estado de recuperación rápida)
2. **Evitación de la congestión (*congestion avoidance*)**: Se aumenta linealmente la venta de congestión
3. **Recuperación rápida (*fast recovery*)**: El valor de ventana de congestión se incrementa en uno por cada ACK duplicado recibido correspondientemente al segmento que falta y que ha causado que TCP entre en estado de recuperación rápida

Capítulo 4: Capa de red

El servicios que presta la capa de red es mover paquetes de un host emisor a un host receptor. En la realización de esta tarea podemos encontrar 3 importantes funciones (2 para redes de datagramas y 3 para redes de circuitos virtuales):

- Reenvío (forwarding): cuando un paquete llega al enlace de entrada de un router, éste tiene que pasar el paquete al enlace de salida apropiado (tabla de ruteo).

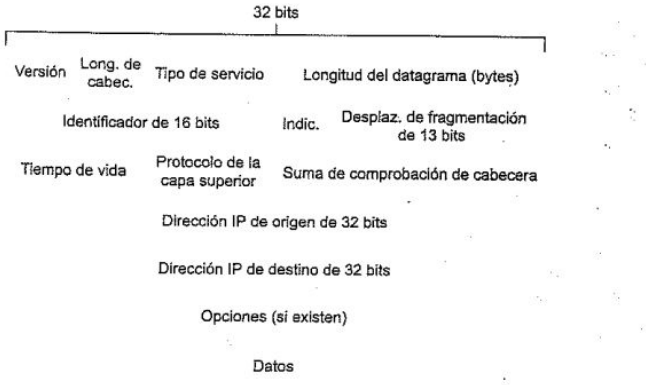
- Enrutamiento (routing): la capa de red tiene que determinar la ruta o camino que deben seguir los paquetes a medida que fluyen de un emisor a un receptor. Los algoritmos que calculan estas rutas se los conoce como **algoritmos de enrutamiento**.
- Configuración de la conexión: En el caso de redes de circuitos virtuales (No internet), la arquitectura de la capa de red requiere que los routers a lo largo del camino desde el emisor al destino, acuerden unos con otros para establecer el estado necesario antes de que los paquetes de datos de la capa de red comiencen a fluir.

La capa de Red permite la conexión desde un host origen a un host destino, enrutando los datagramas IP entre máquinas. En TCP/IP esta capa está implementada en el protocolo IP (cada componente de Internet que tengan una capa de red deben ejecutarlo), e interviene en cada host y encaminador intermedio (router), ya que define los campos del datagrama IP y cómo actúan los sistemas terminales y routers sobre los mismos. El PDU es el datagrama, que encapsula los segmentos de transporte agregándole las direcciones IP origen y destino entre otras cosas.

La capa de red provee un único servicio conocido como **servicio de mejor esfuerzo (best-effort service)**.

Las redes de computadoras que sólo proporcionan un servicio de conexión en la capa de red se conocen como **Redes de circuitos virtuales (VC)** (similar a TCP en capa de enlace); las redes que sólo proporcionan un servicio sin conexión en la capa de red se conocen como **Redes de datagramas** (similar a UDP en la capa de enlace).

Formato de los datagramas:
Datagrama IP versión 4:



Los campos claves son los siguientes:

- Número de versión: 4 bits, especifican la versión del protocolo IP del datagrama. Le sirve al router para determinar cómo interpretar el resto del datagrama IP
- Longitud de cabecera: 4 bits, son necesarios para determinar donde comienzan realmente los datos del datagrama IP. La mayoría de de los datagramas IP no contienen opciones, por lo que el datagrama típico tiene una cabecera de 20 bytes.
- Tipo de servicio: Sirve para poder diferenciar entre los distintos tipos de datagramas IP. Por ejemplo: puede resultar útil diferenciar datagramas en tiempo real (ej: telefonía IP) del tráfico que no es en tiempo real (ej: sesión FTP).

- Longitud del datagrama: es la longitud total del datagrama IP (cabecera más los datos) en bytes. Puesto que este campo tiene una longitud de 16 bytes el tamaño máximo teórico del datagrama IP es de 65535 bytes. Sin embargo rara vez los datagramas tienen una longitud mayor de 1500 bytes.
- Identificador, indicadores, desplazamiento de desfragmentación: estos 3 campos tienen que ver con lo que se denomina fragmentación IP. (IPv6 no permite la fragmentación en los routers)
- Tiempo de vida (TTL): utilizado para que los datagramas no estén eternamente en circulación a través de la red. Este campo se decrementa en una unidad cada vez que un router procesa un datagrama. Si el valor alcanza a 0, el datagrama debe ser descartado
- Protocolo: este campo sólo se utiliza cuando un datagrama IP alcanza su destino final. El valor de este campo indica el protocolo específico de la capa de transporte a la que se pasarán los datos contenidos en ese datagrama IP.
- Suma de comprobación de cabecera: ayuda a los routers a detectar un error de bits en un datagrama IP recibido. Se calcula tratando cada pareja de 2 bytes de la cabecera como un número y sumando y sumando dichos números utilizando complemento a 1 (similar a UDP)
- Direcciones IP origen y destino: cuando el datagrama es creado (en el host origen) el host inserta su dirección IP en el campo dirección origen y la del destino final en el campo destino.
- Opciones: permite ampliar una cabecera IP. La idea original era que rara vez se emplearan. Sin embargo, la mera existencia de este campo complica las cosas, ya que las cabeceras de los datagramas tienen una longitud variable y no puede determinarse a priori donde comienza el campo datos, (IPv6 eliminar este campo) debe calcularse
- Datos: razón de ser del datagrama, contiene el segmento de la capa de transporte (TCP o UDP) que va a entregarse al destino. Sin embargo puede transportar otro tipo de datos, como mensajes ICMP.

Fragmentación del datagrama IP:

No todos los protocolos de la capa de enlace pueden transportar paquetes de la capa de red del mismo tamaño, por ejemplo, las tramas Ethernet pueden transportar hasta 1500 bytes de datos, mientras que las tramas para otros enlaces de área más extensas no pueden transportar más de 576 bytes. La cantidad máxima de datos que una trama de la capa de enlace puede transportar se conoce como unidad máxima de transmisión (MTU, Maximum Transmission Unit).

Puesto que cada datagrama IP se encapsula dentro de una trama de la capa de enlace para ir de un router al siguiente, la MTU del protocolo de la capa de enlace impone un límite estricto a la longitud de un datagrama IP. La solución a este problema consiste en fragmentar los datos del datagrama IP en dos o más datagramas IP más pequeños, encapsular cada uno de los datagramas IP más pequeños en una trama de la capa de enlace distinta y enviar dichas tramas a través del enlace de salida. Cada uno de los datagramas más pequeños se los conoce como **fragmentos**.

Los fragmentos tienen que ser reensamblados antes de llegar a la capa de transporte del destino. Dado que reensamblar los datagramas en los routers añadiría una complejidad

significativa al protocolo y reduciría su rendimiento. Por esto se decidió dar el trabajo a los sistemas terminales.
Para que el host destino puede llevar a cabo la tarea de reensamblado, IPv4 tiene los campos *identificación*, *indicador* y *desplazamiento de fragmentación* en la cabecera del datagrama IP.

- Identificación:
    - Cuando se crea un datagrama el host emisor marca un datagrama con un número de identificación (generalmente incremental).
    - Cuando un router necesita fragmentar marca cada fragmento con la dirección de origen, la dirección de destino y el número de identificación del datagrama original
    - Cuando el host destino recibe una serie de datagramas correspondientes del mismo host emisor, puede examinar los números de identificación de los datagramas para determinar cuáles de ellos son fragmentos de un mismo datagrama más largo.
- Indicador: Dado que IP es un servicio no fiable, es posible que uno o más de los fragmentos nunca lleguen a su destino. Por esta razón el último fragmento tiene el bit indicador puesto a 0 mientras que los demás fragmentos tienen el bit indicador puesto a 1.
- Desplazamiento: dado que IP no garantiza el orden ni que lleguen todos los fragmentos, se utiliza el campo desplazamiento para especificar en qué posición dentro del datagrama IP original encaja el fragmento

En el destino, la carga útil del datagrama (datos) se pasa a la capa de transporte sólo después de la que la capa IP haya reconstruido completamente el datagrama original. Si uno o más de los fragmentos no llegan a destino, el datagrama completo se descarta y no se pasa a la capa de transporte.

Pros: Desempeña un papel importante en el ensamblado de las muchas y dispares tecnologías de la capa de enlace.
Contras:

- Añade complejidad a los routers y sistemas terminales
- Se puede utilizar para crear ataques.

Router y Funciones:
El router es considerado como el dispositivo de la capa de red. El mismo se encarga de interconectar redes, es decir, es necesario un router para que un host que se encuentran en diferentes redes puedan comunicarse entre sí.
Sus principales funciones son:

- Determinar el camino de un paquete entre un host y otro.
- Dado un conjunto de routers con enlaces entre ellos, debe encontrar un buen camino (costo mínimo)

Ruteo estático vs dinámico:

En los algoritmos de ruteo estáticos, las rutas cambian muy lentamente a lo largo del tiempo, a menudo como resultado de intervención humana (por ejemplo cuando alguien modifica las tablas de ruteo de manera manual). Los algoritmos de ruteo dinámicos cambian el camino según los cambios en el tráfico de la red o de la topología. Un algoritmo dinámico puede ser ejecutado ya sea de manera periódica o de manera directa con respuesta a los cambios de topología o cambio en los costos de enlace.

Mientras que los algoritmos de ruteo dinámicos responden más con los cambios en la red, también son más susceptibles a los problemas tales como oscilaciones en las rutas y lazos en las rutas.
El ruteo dinámico se divide en estado de enlace y vector de distancia.

- Estado de enlace (Dijkstra): la topología de la red y todos los costes de enlaces son conocidos. Esto se puede obtener haciendo que cada nodo realice una transmisión de paquetes estado de enlace hacia todos los demás nodos en la red. Cada paquete contiene las identificaciones y los costes en los enlaces que se encuentran directamente enlazados a ese nodo.
- Vector de distancia: es iterativo, asincrónico y distribuido. Se dice que es distribuido porque cada nodo recibe información de uno o más de sus nodos vecinos directamente enlazados a él, realiza el cálculo y luego distribuye de regreso el resultado de los cálculos a sus vecinos. Se dice que es iterativo en el sentido de que este proceso continúa hasta que no se intercambia información entre los vecinos. Se dice que es asincrónico, pues no requiere que los nodos operen de manera conjunta entre ellos.

Direccionamiento IPv4:
Un host normalmente posee un único enlace a la red; cuando IP en el host desea enviar un datagrama lo hace a través de este enlace. El límite entre el host y el enlace físico se denomina **interfaz**.
Los routers tienen varios enlaces, y por ende, varias interfaces, donde cada interfaz tiene su propia dirección IP. Cada una de las interfaces de un host o router de Internet tienen que tener asociada una dirección IP que es globalmente única (excepto en el caso de las interfaces utilizadas para NAT) y estará determinada por la subred a la que está conectada.

La estrategia de asignación de direcciones en Internet se conoce como **Enrutamiento entre dominios sin clase (CIDR, *Classless Interdomain Routing*)**. CIDR generaliza la noción de direccionamiento de subred. Al igual que sucede con las subredes IP, la dirección IP de 32 bits se divide en dos partes (red y host), y de nuevo se expresa en notación decimal con punto como *a.b.c.d/x*, donde *x* indica el número de bits de la primera parte de la dirección.
Por ejemplo, si los 21 primeros bits de la dirección CIDR a.b.c.d/21 especifican el prefijo de la red de una organización y son comunes a las direcciones IP de todos los dispositivos de dicha organización. Los restantes 11 bits identifican entonces a los hosts específicos de la organización. La estructura interna de la organización puede ser tal que esos 11 bits de más a la derecha se empleen para dividir en subredes a la organización, por ejemplo, a.b.c.d/24 podría hacer referencia a una subred específica de la organización.

Antes de que se adoptara el enrutamiento CIDR, la parte de red de una dirección IP estaba restringida a 8, 16 o 24 bits, un esquema de direccionamiento conocido como **direccionamiento con clases**, ya que las subredes de direcciones de 8, 16 y 24 bits se conocían respectivamente como redes de clase A, B y C.

Existe otro tipo de dirección IP, la dirección de difusión 255.255.255.255, la cual se utiliza para broadcast.

Cómo obtener bloques de direcciones:
ISP o ICANN

Cómo obtener direcciones de host.
Se pueden configurar manualmente, o utilizando el **Protocolo de configuración dinámica de host (DHCP, *Dynamic host configuration protocol*)**. DHCP permite a un host obtener (permite que se le asigne) automáticamente una dirección IP. Un administrador de red puede configurar DHCP de modo un host dado reciba la misma dirección IP cada vez que se conecte a la red, o un host puede ser asignado a una **dirección IP temporal** que será diferente cada vez que el host se conecte a la red. DHCP también permite que un host obtenga información adicional, como por ejemplo una máscara de subred, la dirección del router del primer salto, y la dirección de su servidor DNS local.
Gracias a la capacidad de DHCP de automatizar el proceso de conexión de un host a la red, a menudo se dice que es un **protocolo Plug-and-play**. Usos: redes residenciales, y LAN inalámbricas, en las que los hosts se unen a la red y salen de ellas frecuentemente. ISP que trabajan en el mercado residencial, por ejemplo, si un ISP residencial tiene 2.000 clientes, pero no más de 400 clientes están en línea al mismo tiempo. En este caso en lugar de necesitar un bloque de 2048 direcciones, un servidor DHCP que asigne direcciones IP de forma automática sólo necesitará un bloque de 512 direcciones.
DHCP es un protocolo cliente-servidor. Normalmente un cliente es un host recién llegado que desea obtener información de configuración de la red, incluyendo una dirección IP para sí mismo. En el caso más simple cada subred tendrá un servidor DHCP. Si en la subred no hay ningún servidor es necesario un agente de retransmisión DHCP (normalmente un router) que conozca la dirección de un servidor DHCP para dicha red.

Suponiendo que hay un servidor DHCP en una red para un host recién llegado a la red, el protocolo DHCP es un proceso de cuatro pasos:

- Descubrimiento del servidor DHCP: se hace mediante un **mensaje de descubrimiento DHCP**, que envía un cliente dentro de un paquete UDP al puerto 67. El paquete UDP se encapsula en un datagrama IP con ip destino = 255.255.255.255 y una dirección de origen = a 0.0.0.0
- Oferta(s) del servidor DHCP: el servidor DHCP responde al cliente con un **mensaje de oferta DHCP** con IP destino = 255.255.255.255. Cada mensaje de oferta de servidor contiene el ID de transacción del mensaje de descubrimiento recibido, la dirección IP propuesta para el cliente, la máscara de red, y el **tiempo de arrendamiento de la dirección** IP (el tiempo durante el que la dirección IP será válida)

- Solicitud DHCP: el cliente recién llegado seleccionará entre las ofertas del servidor y responderá la oferta seleccionada con un **mensaje de solicitud DHCP**, devolviendo los parámetros de configuración
- ACK DHCP: el servidor contesta al mensaje de solicitud DHCP con un **mensaje ACK DHCP**.

Dado que un cliente puede desear utilizar su dirección por más tiempo del arrendado, DHCP también proporciona un mecanismo que permite a un cliente renovar su tiempo de arrendamiento de una dirección IP.

Traducción de direcciones de red (NAT)

Direcciones privadas: un ámbito con direcciones privadas hace referencia a una red cuyas direcciones sólo tienen significado para los dispositivos internos de dicha red. Los paquetes enviados hacia afuera de una red que contiene direcciones privadas no pueden utilizar estas direcciones (ni como origen ni como destino), porque existen cientos de miles de redes que emplean el mismo bloque de direcciones.

El router NAT *no parece* un router a ojos del mundo exterior. En su lugar el router NAT, se comporta, de cara al exterior, como un único dispositivo con una dirección IP única. El router NAT oculta los detalles de la red doméstica al mundo exterior. La ip exterior generalmente el router la obtiene mediante DHCP de su ISP y el router ejecuta un servidor DHCP para proporcionar direcciones a las computadoras dentro del espacio de direcciones de la red doméstica controlada por el router NAT-DHCP.

Si todos los datagramas que llegan al router NAT procedente de la WAN tienen la misma dirección IP de destino ¿Cómo sabe el router a qué host interno debería reenviar un datagrama dado?. Para esto se utiliza una **tabla de direcciones NAT** almacenada en el router NAT, e incluir los números de puerto, así como las direcciones IP en las entradas de la tabla.

Ejemplo:
Suponga que un usuario de una red doméstica que utiliza el host con la dirección IP 10.0.0.1 solitica una página web almacenada en un servidor web (puerto 80) con la dirección IP 128.119.60.186. El host 10.0.0.1 asigna el número de puerto origen (arbitrario) 3345 y envía el datagrama a LAN. El router NAT recibe el datagrama, genera un nuevo número de puerto origen 5001, para el datagrama, sustituye la dirección IP de origen con su dirección IP de la red WAN 138.76.29.7, y sustituye el número de puerto origen original 3345 por el nuevo número de puerto origen 5001. Al generar un nuevo número de puerto origen, el router NAT puede seleccionar cualquier número de puerto origen que actualmente no se encuentre en la tabla de traducciones NAT. En el router, NAT también añade una entrada a su tabla de traducciones. El servidor web, responde con un datagrama cuya dirección de destino es la dirección IP del router NAT y cuyo puerto de destino es 5001. Cuando este datagrama llega al router NAT, éste indexa la tabla de traducciones NAT utilizando la dirección IP de destino y el número de puerto de destino (3345) apropiados para el navegador de la red doméstica. A continuación, el router reescribe la dirección de destino y el número de puerto destino del datagrama y lo reenvía a la red doméstica.

ICMP

Los hosts y los router utilizan ICMP para intercambiarse información acerca de la capa de red. El uso más típico de ICMP es la generación de informes de error.
ICMP a menudo se considera parte de IP, pero en sentido arquitectónico, se encuentra justo por encima de IP, ya que los mensajes ICMP son transportados dentro de los datagramas IP, al igual que los segmentos TCP o UDP. Cuando un host recibe un datagrama IP con ICMP especificado como el protocolo de la capa superior, demultiplexa el contenido del datagrama para ICMP, al igual que lo haría para TCP o UDP.
Los mensajes ICMP tienen un campo de tipo y un campo de código, y contienen la cabecera y los 8 primeros bytes del datagrama IP que ha dado lugar a generación del mensaje ICMP en primer lugar (de modo que el emisor puede determinar qué datagrama ha producido el error)

Tipo ICMP	Código	Descripción
0	0	respuesta de eco (para ping)
3	0	red de destino inalcanzable
3	1	host de destino inalcanzable
3	2	protocolo de destino inalcanzable
3	3	puerto de destino inalcanzable
3	6	red de destino desconocida
3	7	host de destino desconocido
4	0	regulación del origen (control de congestión)
8	0	solicitud de eco
9	0	anuncio de router
10	0	descubrimiento de router
11	0	TTL caducado
12	0	Cabecera IP errónea

El programa *ping* envía un mensaje ICMP de tipo 8 y código 0 al host especificado. El host de destino, al ver la solicitud de eco, devuelve una respuesta de eco ICMP de tipo 0 y código 0.
Otro interesante mensaje de ICMP es el de regulación de origen. Este mensaje rara vez se emplea en la práctica. Su propósito original es llevar a cabo el control de congestión (permitir a un router congestionado enviar un mensaje ICMP de este tipo a un host para forzarle a reducir su velocidad de transmisión).

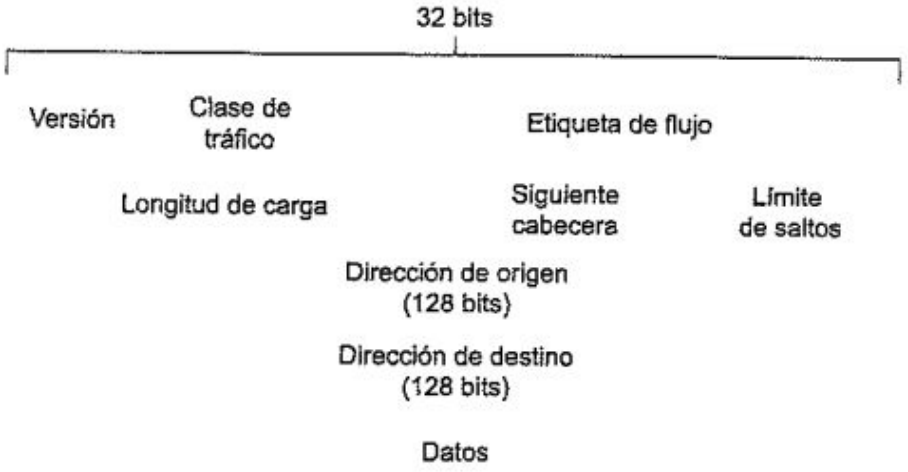
IPv6
Cambios principales:

- Capacidades ampliadas de direccionamiento: IPv6 aumenta el tamaño de la dirección IP de 32 a 128 bits. De esta manera se asegura que el mundo no se

quedará sin direcciones IP. Además de las direcciones de unidifusión y de multidifusión, IPv6 ha introducido un nuevo tipo de dirección denominado **dirección anycast**, que permite entregar un datagrama a uno cualquiera de un grupo de hosts.

- Cabecera de 40 bytes simplificada: la cabecera de longitud fija permite un procesamiento más rápido del datagrama IP. Una nueva codificación de las opciones permite un procesamiento más flexible de las mismas.
- Prioridad y etiquetado de flujo: IPv6 utiliza una definición bastante amplia de **flujo**, esto permite, según RFC: "etiquetar los paquetes que pertenecen a determinados flujos para los que el emisor solicita un tratamiento especial, como un servicio en tiempo real o una calidad de servicio no predeterminados". Por ejemplo, la transmisión de audio y video podría tratarse como un flujo, pero otros elementos más tradicionales, como la transferencia de archivos y el correo electrónico, no se pueden tratar como flujos. Es posible que el tráfico transportado por un usuario de alta prioridad (por ejemplo, alguien que paga para obtener un mejor servicio para su tráfico) tenga que ser tratado como un flujo. La cabecera IPv6 también tiene un campo de 8 bits para definir la clase de tráfico. Este campo al igual que el campo TOS de IPv4, se puede utilizar para dar prioridad a determinados datagramas de un flujo, o ciertas aplicaciones (ejemplo: ICMP) con respecto a datagramas de otras aplicaciones (ejemplo: noticias en red

Formato del datagrama:



- Versión: 4 bits, identifica el número de versión IP. IPv6 transporta un valor de 6 en este campo. Incluir en este campo el valor 4 no implica que se cree un datagrama IPv4 válido.
- Clase de tráfico: 8 bits, similar al TOS de IPv4.
- Etiqueta de flujo: 20 bits, se utiliza para identificar un flujo de datagramas.
- Longitud de la carga útil: 16 bits, se trata como un entero sin signo que proporciona el número de bytes del datagrama IPv6 incluidos a continuación de la cabecera del datagrama, que es de 40 bytes y tiene longitud fija

- Siguiente cabecera: identifica el protocolo al que se entregará el contenido (campo datos) de este datagrama (ej: TCP o UDP). Utiliza mismos valores que el campo de protocolo de cabecera de IPv4.
- Límite de saltos: similar a TTL de IPv4
- Direcciones de origen y de destino: direcciones de 128 bits.
- Datos: igual que IPv4.

Campos que no aparecen en IPv4:

- Fragmentación / Reensamblado: IPv6 no permite ni la fragmentació ni el reensamblado en routers intermedios; estas operaciones sólo pueden ser realizadas por el origen y el destino. Si un router recibe un datagrama IPv6 y es demasiado largo para ser reenviado por el enlace de salida, el router simplemente lo descarta y envía directamente al emisor un mensaje de error ICMP “Paquete demasiado grande”. El emisor puede entonces reenviar los datos utilizando entonces un tamaño de datagrama IP más pequeño. La fragmentación y el reensamblado son operaciones que consumen mucho tiempo, por lo que eliminando esta funcionalidad de los routers se acelera considerablemente el reenvío IP dentro de la red.
- Suma de comprobación de cabecera: puesto que los protocolos de la capa de transporte (Ej: TCP y UDP) y de la capa de enlace a datos (Ethernet) en las capas de internet realizan sumas de comprobación, probablemente se pensó que era demasiado redundante en la capa de red y podría eliminarse, con el fin de que el procesamiento sea más rápido
- Opciones: La cabecera ya no incluye un campo opciones; sin embargo las opciones no han desaparecido. En su lugar, el campo opciones es una de las posibles siguientes cabeceras apuntadas dentro de la cabecera IPv6. Es decir, al igual que las cabeceras de los protocolos TCP o UDP pueden ser la siguiente cabecera dentro de un paquete IP, también puede serlo un campo de opciones.

Capítulo 5: Capa de enlace

Servicios proporcionados por la capa de enlace:
Para transmitir un datagrama a través de un enlace individual se utiliza un protocolo de la capa de enlace. El **protocolo de la capa de enlace** define el formato de de los paquetes intercambiados por los nodos situados en los extremos del enlace, así como las acciones que estos nodos llevan a cabo cuando se envían y reciben los paquetes. Las unidades de

datos intercambiadas por la capa de enlace se denominan **tramas (*frames*)**, y cada trama de la capa de enlace suele encapsular un datagrama de la capa de red.
Acciones del protocolo de capa de enlace:

- Detección de errores:
- Retransmisión:
- Control de flujo:
- Acceso aleatorio:

Tarea nodo a nodo de mover los datagramas de la capa de red a través de un *único enlace* dentro de la ruta. Un mismo datagrama puede ser transportado por diferentes protocolos de la capa de enlace en los distintos enlaces que forman la ruta. Por ejemplo, un datagrama podría ser transportado mediante Ethernet en el primer enlace, PPP en el último enlace y mediante un protocolo WAN de la capa de enlace en los enlaces intermedios. Los servicios proporcionados por los distintos protocolos en una ruta terminal a terminal de la capa de enlace pueden ser distintos. Por ejemplo, algunos protocolos de la capa de enlace proporcionan una entrega fiable, mientras que otros no lo hacen. Por lo tanto la capa de red debe realizar su trabajo terminal a terminal en un conjunto heterogéneo de servicios individuales de la capa de enlace.

Posibles servicios que un protocolo de la capa de enlace puede ofrecer:

- Entramado: casi todos los protocolos de la capa de enlace encapsulan cada datagrama de la capa de red dentro de una trama de la capa de enlace antes de transmitirla a través del enlace. Una trama consta de un campo de datos, en el que se inserta el datagrama de la capa de red, y de una serie de campos de cabecera. La estructura de la trama está especificada por el protocolo de la capa de enlace.
- Acceso al enlace: Un protocolo de control de acceso al medio (MAC, *Medium Access Control*) especifica las reglas que se utilizan para transmitir una trama a través del enlace. Para los enlaces punto a punto que tengan un único emisor en un extremo del enlace y un único receptor en el otro extremo, el protocolo MAC es muy simple (o no existe): el emisor puede enviar una trama siempre que el enlace esté inactivo. El caso más interesante es cuando hay varios nodos compartiendo un mismo enlace de difusión, en cuyo caso se presenta el denominado problema del acceso múltiple. En ese caso, el protocolo MAC sirve para coordinar la transmisión de las tramas de los múltiples nodos
- Entrega fiable: Cuando un protocolo de la capa de enlace proporciona un servicio de entrega fiable, garantiza que va a transportar cada datagrama de la capa de red a través del enlace sin que se produzcan errores. Suele implementarse de forma similar a los servicios de entrega fiable de la capa de transporte mediante reconocimientos y retransmisiones. A menudo se utiliza este servicio en los enlaces que suelen presentar altas tasas de errores, como por ejemplo en los enlaces inalámbricos.
- Control de flujo: Los nodos situados en cada extremo de un enlace tienen una capacidad limitada de almacenamiento en buffer de las tramas. Esto puede ser un problema cuando el nodo receptor puede recibir las tramas a más velocidad de la que puede procesarlas.

- Detección de errores: El hardware de la capa de enlace en un nodo receptor pudiera llegar a decidir, incorrectamente, que un bit contenido en una trama es cero cuando fue transmitido con un uno, y viceversa. Dichos errores de bit se introducen debido a la atenuación de las señales y al ruido electromagnético. Normalmente es más sofisticada que la detección de errores de capa 3 y 4 y se implementa en el Hardware.
- Corrección de errores: Similar a la detección de errores salvo que no sólo se detecta si hay bits erróneos en la trama, sino que también determina en qué puntos de la trama se han producido estos errores (y luego corrige esos errores)
- Semiduplex y full duplex: con la transmisión full-duplex los nodos de ambos extremos de un enlace pueden transmitir paquetes al mismo tiempo. Sin embargo con la transmisión semiduplex un mismo nodo no puede transmitir y recibir al mismo tiempo.

¿Donde se implementa la capa de enlace?

En su mayor parte la capa de enlace se implementa en un **adaptador de red**, también denominado a veces **Tarjeta de interfaz de red (NIC, *Network Interface Card)***. El corazón de la tarjeta adaptador de red es el controlador de la capa de enlace, que normalmente es un único chip de propósito especial que implementa muchos de los servicios de la capa de enlace. Buena parte de la funcionalidad del controlador de la capa de enlace se implementa por hardware. Ejemplo: controlador Ethernet, protocolo WiFi 802.11.

En el lado emisor, el controlador toma un datagrama que haya sido creado y almacenado en la memoria del host por las capas superiores de la pila de protocolos, encapsula el datagrama en una trama de la capa de enlace y luego transmite la trama al enlace de comunicaciones, de acuerdo con el protocolo de acceso al enlace. En el lado receptor, un controlador recibe la trama completa y extrae el datagrama de la capa de red. Si la capa de enlace realiza detección de errores, entonces será el controlador del emisor quien se encargue de configurar los bits de detección de errores en la cabecera de la trama, mientras que el controlador del receptor llevará a cabo la detección de errores. Si la capa de enlace realiza control de flujo, entonces los controladores del emisor y del receptor intercambian información de control de flujo de modo que el emisor envíe las tramas a una velocidad que el receptor sea capaz de aceptar.

Algunos protocolos de la capa de enlace se implementan en software que se ejecuta en la CPU del host. Los componentes de software de la capa de enlace, normalmente implementan la función de más alto nivel de esa capa, como por ejemplo la recepción del datagrama desde la capa de red, el ensamblado de la información de direccionamiento de la capa de enlace y la activación del hardware del controlador. En el lado receptor, el software de la capa de enlace responde a las interrupciones procedentes del controlador (por ejemplo, debidas a la recepción de una o más tramas), se encarga de gestionar las condiciones de error y pasa el datagrama hacia la capa de red.

Técnica de detección y corrección de errores:

Normalmente los datos que hay que proteger incluyen no sólo el datagrama recibido de la capa de red para su transmisión a través del enlace, sino también la información de direccionamiento de la capa de enlace, los números de secuencia y otros campos de la cabecera de la trama de enlace.

Las técnicas de detección y corrección de errores permiten al receptor detectar en ocasiones, *pero no siempre,* que se han producido errores en los bits. Incluso utilizando bits de detección de errores pueden seguir existiendo **errores de bit no detectados**; es decir, el receptor podría perfectamente ser inconsciente de que la información recibida contiene errores en los bits. Lo que intentaremos será elegir un esquema de detección de errores que haga que la probabilidad de que se produzcan estos casos sea pequeña. Por regla general, cuanto más sofisticadas son las técnicas de detección y corrección de errores, mayores son los recursos adicionales necesarios.
Algunas técnicas son:

- Comprobaciones de paridad: Es el esquema más sencillo para detectar errores y consiste en añadir un bit de paridad al final del bloque de datos. El valor de este bit se determina de tal forma que el carácter resultante tenga un número impar de unos (paridad impar) o un número par (paridad par). Si se utiliza paridad impar y se invierte uno o un número impar de bits erróneamente durante la transmisión, el receptor detectará un error. Sin embargo si se invierten dos o un número par de bits aparecerá un error no detectado. Algo similar ocurre si se usa paridad par. Típicamente la paridad par se usa para la transmisión sincrónica y la impar para la asincrónica.
- Métodos basados en suma de comprobación: los *d* bits de datos se tratan como una secuencia de enteros de *k* bits. Un método simple de suma de comprobación consiste en sumar estos enteros de *k* bits y utilizar la suma restante como bits de detección de errores. La **suma de comprobación de Internet** está basada en este enfoque: los bytes de datos se tratan como enteros de 16 bits y se suman. Entonces, se utiliza el complemento a 1 de la suma de los datos recibidos (incluyendo la suma de comprobación) y comprobando si el resultado tiene todos los bits a 1. Proporciona una protección relativamente débil frente a otros métodos como CRC. Dado que la implementación de errores en la capa de enlace se implementa por hardware se pueden utilizar métodos más costosos costoso en cuanto a recursos como CRC.
- Comprobación de redundancia cíclica (CRC): Es uno de los mecanismos más comunes y potente. Dado un bloque o mensaje de k bits el transmisor genera una secuencia de n bits denominada secuencia de comprobación de la trama (FCS) de tal manera que la trama resultante con n+k bits sea divisible por algún número determinado. El receptor dividirá la trama recibida por ese número y si no hay resto en la división se supone que no ha habido errores. Se puede resolver el procedimiento de tres maneras: usando aritmética de módulo 2, usando polinomios o usando lógica digital. Si usamos polinomios podemos decir que FCS es el resto de la división de: (Mensaje * Xr) / Polinomio Generador, donde r es el grado del polinomio generador.

Protocolos de acceso múltiple:
Existen dos tipos de enlace de red:

- Enlaces punto a punto: único emisor y único receptor
    - Protocolo punto a punto (PPP, *Point-to-Point Protocol*)
    - Protocolo de control del enlace de datos de alto nivel (HDLC, *High-level Data Link Control*)

- Enlaces de difusión (*broadcast*): puede tener múltiples emisores y receptores, todos conectados al mismo y único canal de difusión compartido
    - Ethernet
    - Redes LAN inalámbricas

Cómo coordinar el acceso de múltiples nodos emisores y receptores a un canal de difusión compartido, lo cual se conoce con el nombre de **problema del acceso múltiple**.

Protocolos de acceso múltiple, protocolos mediante los cuales los nodos se encargan de regular sus transmisiones al canal de difusión compartido. Son necesarios en una amplia variedad de escenarios de red, incluyendo las redes de área local tanto cableadas como inalámbricas y las redes satélite.
Dado que todos los nodos son capaces de transmitir tramas, podría darse el caso de que más de dos nodos transmitieran tramas al mismo tiempo. Cuando esto sucede, todos los nodos reciben varias tramas simultáneamente; es decir, las tramas transmitidas **colisionan** en todos los receptores. Normalmente, cuando se produce una colisión ninguno de los nodos receptores puede interpretar ninguna de las tramas transmitidas; en cierto sentido, las señales de las tramas que han colisionado se entremezclan y no pueden separarse. Por lo tanto todas las tramas implicadas en la colisión se pierden y el canal de difusión está desaprovechado durante el intervalo de colisión.
Para poder garantizar que el canal de difusión realice un trabajo útil aun cuando haya múltiples nodos activos, es necesario coordinar de alguna manera las transmisiones de esos nodos activos. Casi todos los protocolos de acceso múltiple se pueden clasificar en tres categorías:

1. Protocolos de particionamiento de canal
2. Protocolos de acceso aleatorio
3. Protocolos de toma de turnos.

Idealmente, un protocolo de acceso múltiple para un canal de difusión con una velocidad de *R* bits por segundo debería tener las siguientes características deseables:

1. Cuando solo haya un nodo que tenga datos para enviar, a dicho nodo se le asignará una tasa de transferencia de *R* bps.
2. Cuando haya *M* nodos con datos para enviar, cada uno de esos nodos tendrá una tasa de transferencia de *R/M* bps. Esto no implica que cada uno de los nodos tenga una tasa de transferencia constante de *R/M*, sino más bien que cada nodo tendrá una tasa media de transferencia de *R/M* a lo largo de un intervalo de tiempo definido adecuadamente.
3. El protocolo será descentralizado; es decir no habrá ningún nodo maestro que pueda actuar como punto único de fallo para la red.

Direcciones MAC:
No son los nodos (es decir, los hosts o routers) los que tienen asignadas direcciones de la capa de enlace, sino que las direcciones de la capa de enlace se asignan a los adaptadores instalados en cada nodo. A las direcciones de la capa de enlace se las denomina de diversas formas como **dirección LAN, dirección física** o **dirección MAC**. La dirección

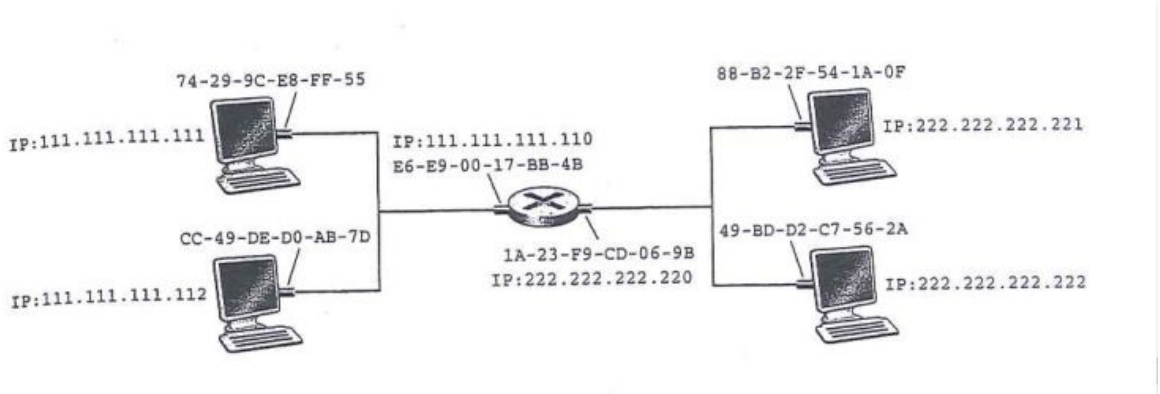
MAC tiene 6 bytes de longitud y suelen expresarse en notación hexadecimal, indicándose cada byte de la dirección mediante una pareja de números hexadecimales.
Una propiedad interesante de las direcciones MAC es que nunca puede haber dos adaptadores con la misma dirección.
La dirección MAC de un adaptador tiene una estructura plana y nunca varía independientemente de donde se lleve el adaptador.
Cuando un adaptador de un emisor quiere enviar una trama a otro adaptador de destino, inserta la dirección MAC del de destino en la trama y luego la envía a través de la red LAN. Si la red LAN es una LAN de difusión, la trama será recibida y procesada por todos los demás adaptadores de la LAN. En particular, cada adaptador que reciba la trama comprobará si la dirección MAC de destino contenida en la trama se corresponde con su propia dirección MAC. Si no hay una correspondencia entre ambas direcciones, el adaptador descarta la trama, sin pasar el datagrama de la capa de red hacia arriba por la pila de protocolos. De este modo, sólo el nodo de destino será interrumpido cuando se reciba la trama.
Sin embargo, *si se quiere* que todos los demás adaptadores de la LAN reciban y *procesen* la trama que va a enviar. En este caso, el adaptador emisor inserta una **dirección de difusión** MAC especial en el campo de dirección de destino de la trama. Para las redes LAN que utilizan direcciones de 6 bytes, la dirección de difusión es una cadena compuesta por 40 unos (1) consecutivos (es decir, FF-FF-FF-FF-FF-FF en notación hexadecimal).

Protocolo de resolución de direcciones (ARP)
Encargado de traducción entre IP y MAC. Un módulo ARP en el nodo emisor toma como entrada cualquier dirección IP de la misma LAN y devuelve la dirección MAC correspondiente.
ARP resuelve una dirección IP en una dirección MAC. En muchos sentidos, esto es análogo a DNS. SIn embargo, una diferencia importante entre los dos resolvedores es que DNS resuelve nombres de host para hosts ubicados en cualquier lugar de Internet, mientras que ARP resuelve direcciones IP sólo para los nodos de una misma subred.

¿Cómo funciona ARP?
Cada nodo (host o router) tiene en su memoria una **tabla ARP**, que contiene las correspondencias entre las direcciones IP y las direcciones MAC. La tabla ARP también contiene un valor de tiempo de vida (TTL), que indica cuándo se eliminará cada correspondencia de la tabla. Algunos nodos pueden no estar en la tabla.
Si un nodo quiere enviar un datagrama con direccionamiento IP a otro nodo de dicha subred, que no está en su tabla ARP, el nodo emisor construye un paquete especial denominado **paquete ARP.** Un paquete ARP contiene varios campos, incluyendo las direcciones MAC e IP del emisor y el receptor. Los paquetes de consulta y de respuesta ARP tienen el mismo formato. El propósito del paquete de consulta ARP es consultar a todos los demás nodos de la subred con el fin de determinar la dirección MAC correspondiente a la dirección IP que está resolviendo. Para esto el adaptador encapsula el paquete ARP en una trama de la capa de enlace, utilizando la dirección de difusión (FF:FF:FF:FF:FF:FF) como dirección de destino de la trama y la transmite a la subred. La trama que contiene la consulta ARP es recibida por todos los demás adaptadores existentes en la subred (a causa de la dirección de difusión) y cada adaptador pasa la consulta ARP

contenida en la trama al módulo ARP de dicho nodo. Cada nodo realiza una comprobación para ver si su dirección IP se corresponde con la dirección IP de destino del paquete ARP. El único nodo que que se produzca la coincidencia devolverá al nodo que ha realizado la consulta una respuesta ARP con la correspondencia deseada. El nodo que ha realizado la consulta podrá entonces actualizar su tabla ARP y enviar su datagrama IP, encapsulado dentro de una trama de la capa de enlace cuya dirección de destino MAC es la del nodo que ha contestado la anterior consulta ARP

Envío de un datagrama a un nodo fuera de la subred



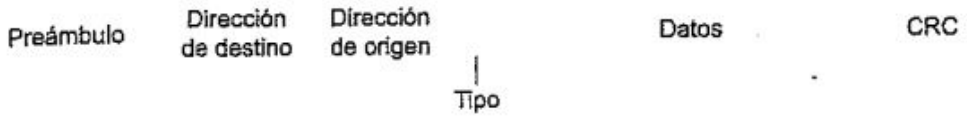
Ejemplo:
Existen dos tipos de nodos: hosts y routers. Cada host tiene exactamente una dirección IP y un adaptador. Pero, un router tiene una dirección IP para *cada una* de sus interfaces. Para cada interfaz de router existe también un módulo ARP (en el router) y un adaptador. Dado que el router del ejemplo tiene dos interfaces, tendrá dos direcciones IP, dos módulos ARP y dos adaptadores. Por su puesto cada adaptador tendrá su propia dirección MAC.
La subred 1 tiene la dirección de red 111.111.111/24 y la Subred 2 tiene la dirección de red 222.222.222/24.
¿Cómo un host de la Subred 1 enviaría un datagrama a un host de la Subred 2?
Supongamos que el host 111.111.111.111 desea enviar un datagrama IP a un host 222.222.222.222. El host emisor pasa el datagrama a su adaptador. Pero el host emisor también tiene que indicar una dirección MAC de destino apropiada. Una posibilidad sería usar la dirección MAC del host 222.222.222.222, sin embargo eso no funcionaría dado que si el adaptador del emisor utilizaría dicha dirección MAC, entonces ninguno de los adaptadores de la Subred 1 se molestaría en pasar el datagrama IP a su capa de red, ya que la dirección de destino de la trama no coincidiría con la dirección MAC de ningún adaptador de la Subred 1.
Para que un datagrama vaya desde 111.111.111.111 a un nodo de la Subred 2, el datagrama tiene en primer lugar que ser enviado a la interfaz del router 111.111.111.110, que es la dirección IP del router del primer salto en el camino hacia su destino final. Por tanto, la dirección MAC apropiada para la trama es la dirección del adaptador de la interfaz de router 111.111.111.110 (E6-E9-00-17-BB-4B).
Pasos que realiza el emisor para obtener la dirección MAC:

- Obtiene la dirección MAC del router (111.111.111.110) utilizando ARP
- Crea una trama (que contiene el datagrama direccionado a 222.222.222.222) y envía la trama hacia la Subred 1

- El adaptador del router ve que la trama de enlace se dirige hacia él y, por tanto, pasa la trama a la capa de red del router.
- El router determina la interfaz a la cual debe enviar el paquete utilizando su tabla de reenvío (tabla de ruteo).
- Obtiene la dirección MAC del host destino (222.222.222.222) utilizando ARP
- Encapsula el datagrama en una nueva trama y la envía a la subred 2.

Ethernet:
Razones del éxito de Ethernet:
- Fue la primera LAN de alta velocidad ampliamente implantada
- Token Ring, FDDI y ATM eran más complejas y caras que Ethernet
- La razón más determinante para cambiar a otra tecnología LAN era la más alta velocidad de datos, sin embargo Ethernet siempre se defendió produciendo versiones que operaban a velocidades iguales o incluso mayores

Estructura de la trama Ethernet:



Ejemplo:
Consideremos que el adaptador de envío, el adaptador A, tiene la dirección física AA-AA-AA-AA-AA-AA, y el adaptador receptor, el adaptador B, tiene la dirección física BB-BB-BB-BB-BB-BB. El adaptador de envío encapsula el datagrama IP en el marco Ethernet y pasa el marco a la capa física. El adaptador receptor recibe el marco desde la capa física, extrae el datagrama IP, y pasa dicho datagrama a la capa de red.

Campos:
- Campo de datos (46 a 1.500 bytes), lleva el datagrama IP. La unidad de transferencia máxima (MTU) de Ethernet es de 1.500 bytes. Esto significa que si el datagrama IP supera los 1.500 bytes, entonces el host debe fragmentar el datagrama. El tamaño mínimo del campo de datos es de 46 bytes. Esto significa que si el datagrama es menor de 46 bytes, el campo de datos tiene que ser rellenado hasta los 46 bytes. Cuando se utiliza el rellenado, los datos pasados a la capa de red contienen el relleno, así como un datagrama IP. La capa de red utiliza el campo longitud de la cabecera del datagrama IP para eliminar el relleno.
-
- Dirección de destino (6 bytes): contiene la dirección LAN del adaptador del destino, BB-BB-BB-BB-BB-BB. Cuando el adaptador B recibe un marco Ethernet con una dirección de destino distinta de su propia dirección física, BB-BB-BB-BB-BB-BB, o de la dirección de difusión de la LAN, lo desecha. Si no, pasa el contenido del campo de datos a la capa de red.
- Dirección de origen (6 bytes): contiene las direcciones LAN del adaptador que transmite el marco a la red, AA-AA-AA-AA-AA-AA.

- Campo de tipo (2 bytes): permite multiplexar los protocolos de la capa de red. Para comprender esta idea, necesitamos tener en cuenta que los host pueden utilizar otros protocolos distintos de la capa de red además de IP. De hecho, un host dado puede soportar múltiples protocolos de la capa de red que utiliza protocolos diferentes para aplicaciones diferentes. Por esta razón, cuando el marco Ethernet llega al adaptador B, éste necesita saber qué protocolo de la capa de red debe pasar (es decir, demultiplexar) el contenido del campo de datos. IP y otros protocolos tienen todos su propio número de tipo estandarizado. Además, el protocolo ARP tiene su propio número de tipo. El campo de tipo es análogo al campo de protocolo del datagrama de la capa de red y a los campos de número de puerto del segmento de la capa de transporte; todos estos campos sirven para enlazar un protocolo de una capa con un protocolo de la capa superior.
- Comprobación de redundancia cíclico (CRC) (4 bytes), el propósito del campo CRC es permitir al adaptador de recepción, el adaptador B, detectar los errores de bit de la trama.
- Preámbulo (8 bytes), cada uno de los primeros 7 bytes del preámbulo tiene un valor de 10101010; el último byte es 10101011. Los primeros 7 bytes del preámbulo sirven para “despertar” a los adaptadores receptores y para sincronizar sus relojes con el reloj del emisor. ¿Por qué deben realizar la sincronización los relojes? El adaptador A intenta transmitir el marco a 10 Mbps, 100 Mbps, o 1 Gbps, dependiendo del tipo de la LAN Ethernet. Sin embargo, como no hay nada absolutamente perfecto, el adaptador A no transmitirá el marco exactamente a la tasa destino; habrá siempre algún desplazamiento de la tasa objetivo, un desplazamiento que no es conocido a priori por los otros adaptadores de la LAN. Un adaptador receptor puede bloquear un reloj del adaptador A simplemente bloqueando los bits en los primeros 7 bytes del preámbulo. Los últimos dos bits del octavo byte del preámbulo (los primeros dos unos consecutivos) alertan al adaptador B de que el material importante está a punto de llegar. Cuando el host B ve los dos unos consecutivos, sabe que los 6 siguientes bytes son la dirección de destino. Un adaptador puede decir cuándo un marco finaliza simplemente detectando la ausencia de corriente.

Servicio sin conexión no fiable:
Todas las tecnologías Ethernet proporcionan un servicio sin conexión a la capa de red; es decir cuando el adaptador A desea enviar un datagrama al adaptador B, el adaptador A encapsula el datagrama en una trama Ethernet y la envía a la red LAN, sin establecer previamente un acuerdo con el adaptador B. Es análogo al servicio de datagramas de la capa 3 de IP y al servicio sin conexión de la capa 4 UDP.
Las tecnologías Ethernet proporcionan un servicio no fiable a la capa de red. Específicamente cuando el adaptador B recibe una trama procedente del adaptador A ejecuta una comprobación CRC, pero ni envía el mensaje de reconocimiento cuando la trama pasa la comprobación CRC, ni envía un mensaje de reconocimiento negativo cuando la comprobación CRC falla, en el último caso simplemente lo descarta.

Algoritmo de control de colisiones:
- Después de la primer colisión se espera 0 o 1 intervalo de tiempo.
- Después de la segunda colisión se espera 0, 1, 2, o 3 intervalos M aleatoriamente.

- Después de la colisión i se elige un número entre o y 2i -1
- El número máximo de intentos es 16. Después de esto se reporta al host.

Conmutadores de la capa de enlace (Switchs):

La función de un conmutador es recibir las tramas de la capa de enlace entrantes y reenviarlas a los enlaces de salida. El propio conmutador es **transparente** para los nodos.

Reenvío y filtrado:

El **filtrado** es la función del conmutador que determina si una trama debe ser reenviada a alguna interfaz o debe ser descartada. El **reenvío** es la función del conmutador que determina las interfaces a las que una trama debe dirigirse y luego envía la trama a esas interfaces.

Las funciones de filtrado y reenvío del conmutador se realizan utilizando la **tabla del conmutador**. Esta tabla contiene entradas para algunos nodos, no necesariamente todos, de una red LAN. Una entrada de la tabla del conmutador contiene

- La dirección MAC de un nodo
- La interfaz del conmutador que lleva hacia el nodo
- El instante en el que la entrada para el nodo fue incluida en la tabla.

Ejemplo:

Una trama con la dirección destino DD-DD-DD-DD-DD-DD llega a la interfaz *x* del conmutador. Éste buscará en su tabla la dirección MAC DD-DD-DD-DD-DD-DD. Hay 3 posibilidades:

- No hay ninguna entrada en la tabla para DD-DD-DD-DD-DD-DD. En este caso, el conmutador reenvía copias de la trama a los buffers de salida que preceden a *todas* las interfaces salvo a la interfaz *x*. En otras palabras difunde la trama.
- Existe una entrada en la tabla que asocia DD-DD-DD-DD-DD-DD con la interfaz *x*. En este caso, la trama procede de un segmento de la LAN que contiene al adaptador DD-DD-DD-DD-DD-DD. No existe la necesidad de reenviar la trama a ninguna de las restantes interfaces, el conmutador lleva a cabo la función de filtrado descartando la trama.
- Existe una entrada en la tabla que asocia DD-DD-DD-DD-DD-DD con la interfaz *y != x*. En este caso, la trama tiene que ser reenviada al segmento de la LAN conectado al a interfaz *y*. El conmutador lleva a cabo su función de reenvío colocando la trama en un buffer de salida que precede a la interfaz *y*.

Auto-aprendizaje: los conmutadores tienen la propiedad de que su tabla se construye de forma automática, dinámica y autónoma, sin intervención de un administrador de redes ni de ningún protocolo de configuración. Esta capacidad se lleva a cabo de la siguiente forma:

1. Inicialmente, la tabla del conmutador está vacía.
2. Para cada trama entrante recibida en una interfaz, el conmutador almacena en su tabla:
    a. La dirección MAC especificada en el *campo dirección de origen* de la trama
    b. La interfaz de la que procede la trama

c. La hora actual

3. El conmutador borra una dirección de la tabla si no se recibe ninguna trama con esa dirección como dirección de origen transcurrido un cierto periodo de tiempo (**tiempo de envejecimiento**)

Ventajas de conmutadores (switches) en lugar de enlaces de difusión (Hubs)

- Eliminación de las colisiones: En una red LAN construida con conmutadores (y sin concentradores) no se desperdicia ancho de banda a causa de las colisiones. Los conmutadores almacenan las tramas en un buffer y nunca transmiten más de una trama a un segmento simultáneamente. Al igual que con los routers, la tasa máxima de transferencia agregada de un conmutador es la suma de todas las tasas de las interfaces del conmutador. Por tanto, los conmutadores proporcionan una mejora significativa en cuanto al rendimiento respecto al de las redes LAN con enlaces de difusión
- Enlaces heterogéneos: Dado que un conmutador aísla un enlace de otro, los distintos enlaces de una LAN pueden operar a velocidades diferentes y pueden utilizar diferentes medios físicos.
- Administración Además de proporcionar una seguridad mejorada, un conmutador también facilita las tareas de gestión de la red. Por ejemplo, si un adaptador de red funciona mal y envía continuamente tramas Ethernet, un conmutador puede detectar el problema y desconectar internamente el adaptador que está funcionando incorrectamente.

Redes de área local virtuales (VLAN):

Un conmutador compatible con **redes de área local virtuales (VLAN, *Virtual Local Area Network*)** permite definir múltiples redes de área local *virtuales* sobre una única infraestructura de red de área local *física*. Los hosts de una VLAN se comunican entre sí como si sólo ellos (y ningún otro host) estuvieran conectados al conmutador. En una VLAN basada en puertos, el administrador de la red divide los puertos (interfaces) del conmutador en grupos. Cada grupo constituye una VLAN, con los puertos de cada VLAN formando un dominio de difusión (el tráfico de difusión de un puerto sólo puede llegar a los demás puertos del grupo). Es fácil imaginar cómo se configura y funciona el conmutador para redes VLAN: el administrador de la red declara que un puerto pertenece a una determinada VLAN (los puertos no declarados pertenecen a una VLAN predeterminada) utilizando un software de gestión de conmutadores; en el conmutador se mantiene una tabla de correspondencias entre puertos y redes VLAN.

¿Cómo puede enviarse tráfico entre 2 VLAN?

Una forma de resolver esto sería conectando un puerto del conmutador VLAN a un router externo y configurando dicho puerto para que pertenezca a las 2 VLAN. Aunque ambas VLAN compartan el mismo conmutador físico la configuración lógica sería como si tuvieran conmutadores separados conectados a través de un router. Los fabricantes de conmutadores hacen que dicha tarea de configuración resulte sencilla para los administradores de red, incorporando en un único dispositivo in conmutador VLAN y un router (con lo que no es necesario utilizar un router externo separado).

¿mismas VLAN con más de un conmutador?
La solución consiste en interconectar los conmutadores VLAN utilizando la técnica conocida como **troncalización VLAN (VLAN *Trunking)***. Con esta técnica, un puerto especial de cada conmutador se configura como un puerto troncal para intercontectar los dos conmutadores VLAN. El puerto troncal pertenece a todas las VLAN y las tramas enviadas a cualquier CLAN son reenviadas a través del enlace troncal hacia el otro conmutador. Pero ¿Cómo sabe un conmutador que una trama que ha llegado a un puerto troncal pertenece a una VLAN concreta? el IEEE ha definido un formato de trama Ethernet ampliado, 802.1.q, para las tramas que atraviesan un enlace troncal VLAN. La trama 802.1Q está formada por la trama Ethernet estándar más una **etiqueta VLAN** de cuatro bytes añadida a la cabecera que transporta la identidad de la VLAN a la que pertenece la trama.

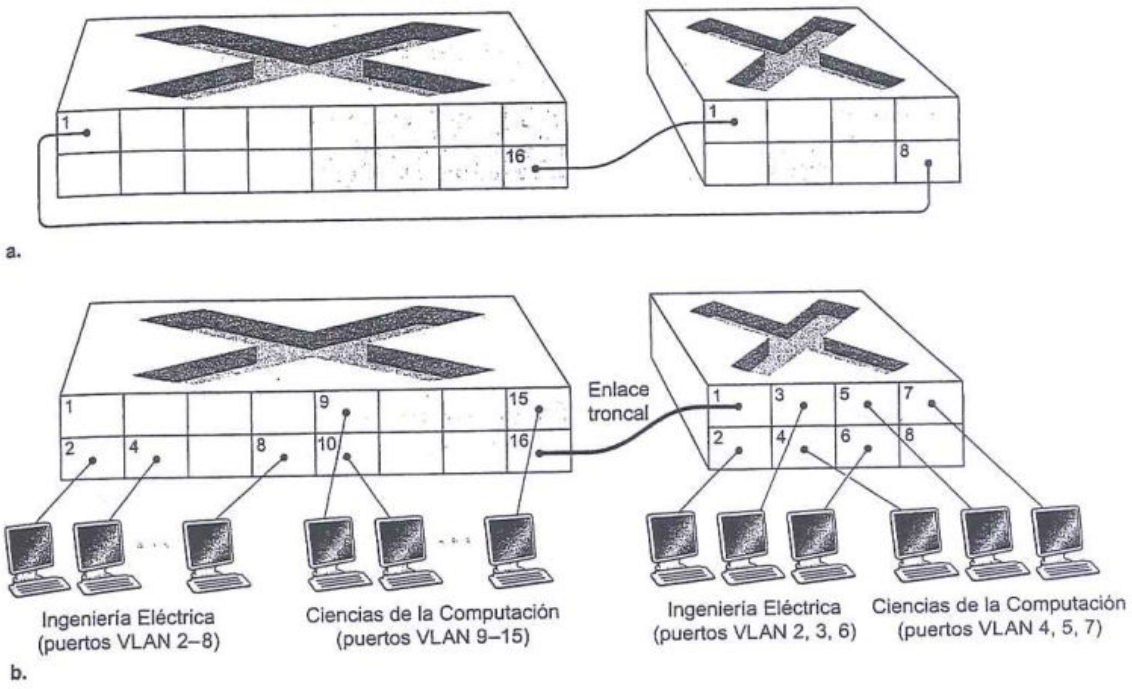


Figura 5.31 • Conexión de dos conmutadores VLAN con dos redes VLAN: (a) dos cables (b) enlace troncal.

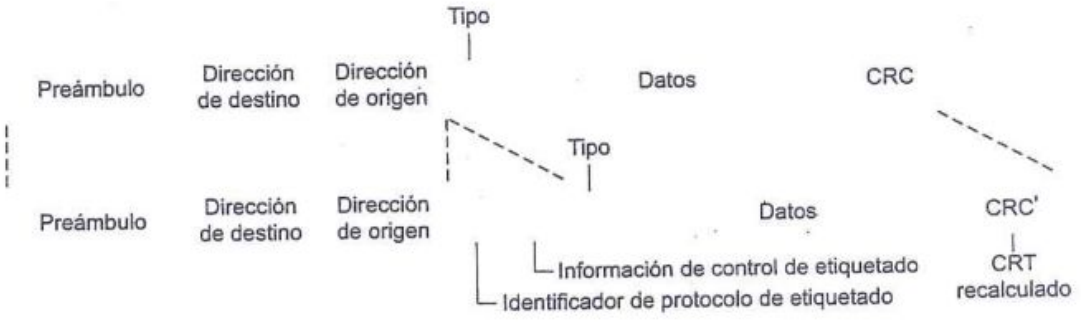


Figura 5.32 • Trama Ethernet original (arriba), trama Ethernet con etiquetado VLAN 802.1Q (abajo).

Dispositivos dividen dominios de broadcast? ¿Y dominios de colisión?

Un dominio de difusión (Broadcast) es un área lógica en una red en la que cualquier ordenador conectado a la red puede transmitir directamente a cualquier otro en el dominio sin precisar ningún dispositivo de encaminamiento, dado que comparten la misma subred, dirección de puerta de enlace y están en la misma VLAN (VLAN por defecto o instalada).

Un dominio de colisión es un segmento físico de una red de computadores donde es posible que las tramas puedan "colisionar" (interferir) con otros. Estas colisiones se dan particularmente en el protocolo de red Ethernet.

A medida que aumenta el número de nodos que pueden transmitir en un segmento de red, aumentan las posibilidades de que dos de ellos transmitan a la vez. Esta transmisión simultánea ocasiona una interferencia entre las señales de ambos nodos, que se conoce como colisión. Conforme aumenta el número de colisiones disminuye el rendimiento de la red.

Un dominio de colisión puede estar constituido por un solo segmento de cable Ethernet en una Ethernet de medio compartido, o todos los nodos que afluyen a un concentrador Ethernet en una Ethernet de par trenzado, o incluso todos los nodos que afluyen a una red de concentradores y repetidores.

Para segmentar dominios de colisión, se utilizan Routers, y Switch/Bridge.

El Hub no genera ningún tipo de división (ni de colisión, ni de broadcast), ya que es una especie de “extensor” de la señal que pertenece a la capa física.