

Resumen de:

REDES DE DATOS



Autor: Adrián Botta

Año: 2010

Fuente:
Redes de computadoras - Andrew S. Tanenbaum (3ra y 4ta edición)

UNIDAD 1: Introducción

Redes de Computadoras: Conjunto de computadoras autónomas interconectadas. Se dice que dos computadoras están interconectadas si pueden intercambiar información. Ni Internet ni Web son una red de computadoras:

- Internet no es una red única, sino una red de redes
- Web es un Sistema Distribuido que se ejecuta sobre Internet

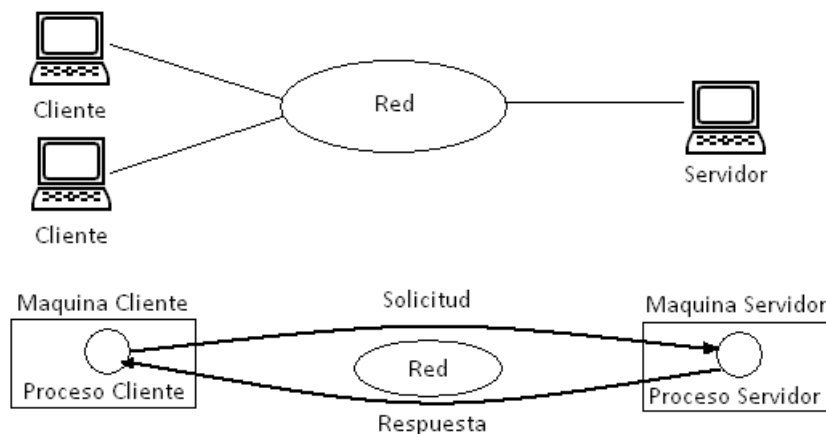
Sistema Distribuido vs Red

En un sistema distribuido, un conjunto de computadoras independientes aparece ante sus usuarios como un sistema consistente y único. Con frecuencia, una capa de software que se ejecuta sobre el sistema operativo, llamada middleware, es la responsable de implementar este modelo. Ej: [WWW \(World Wide Web\)](#).

En una red de computadoras no existe esta consistencia, modelo, ni software. Los usuarios están expuestos a las máquinas reales y el sistema no hace ningún intento porque las máquinas se vean y actúen de manera similar.

De este modo, podemos decir que la diferencia entre ambos sistemas está en el software, y que un sistema distribuido procesa información, mientras que una red intercambia información.

Modelo Cliente Servidor



Usos de Las Redes de Computadoras

1. Objetivos o Fines de las Redes

- a. Compartir Recursos
- b. Compartir Información
- c. Ahorrar tiempo y dinero
- d. Acabar con la tiranía de la geografía

2. Aplicaciones de Negocios

- a. E-Mail (Correo Electrónico)
- b. Trabajo en línea
- c. Video Conferencia
- d. Comercio Electrónico (Negocios por internet)

3. Aplicaciones Domésticas

- a. Acceso a información Remota: Una persona accede a una Base de Datos Remota.
[Ejemplo: Leer un diario, descargar un libro digital, etc.](#)
- b. Comunicación Persona a persona: [Ej: Email, chat, P2P, VoIP, Video-llamada, Teleaprendizaje](#)
- c. Entretenimiento interactivo: [Ej: Juegos, TV Interactiva, Video bajo demanda](#)
- d. Comercio Electrónico: [Ej: Comprar, Subastar, Manejar cuentas bancarias](#)

4. Usuarios Móviles: Notebooks, PDA (Asistentes Personales Digitales)

- a. Oficina portátil
- b. Uso académico
- c. Uso Militar
- d. Comercio móvil

Inalámbrica	Móvil	Aplicaciones
No	No	Computadoras de escritorio en oficinas
No	Si	Una Computadora portátil usada en un cuarto de hotel
Si	No	Redes en construcciones antiguas sin cableado
Si	Si	Oficina Portátil; PDA para inventario de almacén

5. Temas Sociales

- a. Grupos de Noticias: Sexo, Política, Religión generan conflictos. No se pueden borrar estos mensajes porque afectarían a la libre expresión de las personas.
- b. Derechos de la información enviada por usuarios. Por ejemplo, cuando los empleadores leen los e-mails de sus subordinados y los usan en su contra.
- c. Derechos del Gobierno y los ciudadanos. Idem anterior, solo que con el gobierno
- d. Mensajes Anónimos
- e. Robo de Identidad
- f. Violación de derechos de autor

HADRWARE DE REDES

1. Clasificación por Tecnología de Transmisión

- a. Enlaces de Difusión (broadcast): Hay un solo canal de comunicación compartido por todas las máquinas. Se permite enviar un mensaje a un destino en particular, a muchos destinos (multidifusión - *multicasting*) y a todos los destinos (*broadcast*). Según como se asigne el canal, se pueden dividir en:
 - i. Estáticas: Desperdicia capacidad de canal cuando una máquina no tienen nada que transmitir al llegar su turno.
 - ii. Dinámicas: Asigna el canal bajo demanda. Los métodos de asignación pueden ser Centralizados (una estación central determina quien transmite), o descentralizados (cada máquina decide cuando transmitir)
- b. Enlaces punto a punto: Son muchas conexiones entre pares individuales entre máquinas. Para ir del origen al destino, un paquete podría tener que visitar primero una o varias máquinas intermedias. La transmisión punto a punto con un emisor y un receptor se conoce como unidifusión (*unicasting*)

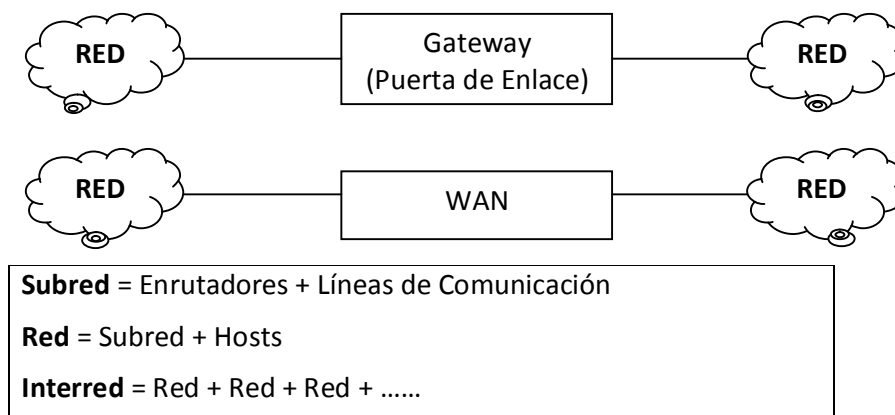
2. Clasificación por Escala

- a. PAN (Personal Area Network): Redes destinadas a una sola persona, en pocos m²
- b. LAN (Local Area Network): Redes de Área local. Ej: [Cuarto](#), [Edificio](#), [Campus](#)
 - i. Tecnología de Transmisión: Mbps, Gbps; bajo retardo; pocos errores
 - ii. Tamaño: El tiempo de transmisión, en el peor de los casos, es limitado y conocido de antemano, lo que permite utilizar ciertos tipos de diseño y simplificar la administración de la red.
 - iii. Topología: Es de difusión. Puede ser Ethernet (Bus con Control descentralizado - 802.3) o Anillo (Token Ring, FDDI)
- c. MAN (Metropolitan Area Network): Abarca una ciudad. Ej: [HFC \(Híbrido Fibra-Coaxil\)](#)
- d. WAN (Wide Area Network): Abarca una gran área geográfica, un país o continente. Está formada por **hosts** (máquinas), conectados mediante una **subred de comunicación**. Los clientes son dueños de los hosts, y generalmente la subred es propiedad de los ISP. La función de la subred es llevar los mensajes de un host a otro. La subred está formada por:
 - i. Líneas de Transmisión: Mueven bits entre máquinas. Ej: [Cobre](#), [Fibra](#), [radio enlaces](#)
 - ii. Elementos de Conmutación: Computadoras especializadas que conectan N líneas de transmisión. Ej: [routers](#), [switchs](#)

Concluyendo, subred es el conjunto de líneas de transmisión y elementos de conmutación que permiten mover información de un host origen a un host destino.

Las WAN pueden ser de Almacenamiento y Reenvío (o Conmutación de paquetes) o Satelitales (difusión)

- e. Interred: Conjunto de redes interconectadas. Tiene alcance mundial. Ej: [Internet](#)



Redes Inalámbricas

1. **Interconexión de Sistemas**: Conexiones de corto alcance, generalmente bajo el paradigma maestro-esclavo. Ej: [Bluetooth](#)
2. **LAN's Inalámbricas (802.11)**: Pueden ser con o sin access point
3. **WAN's Inalámbricas**: Pueden ser de Baja Velocidad (Ej: [Telefonía celular](#)) o de Alta velocidad ([Multipuntos – 802.16](#))

Redes Domésticas

1. Categorías

- a. Computadoras: de escritorio, portátiles, PDA, periféricos compartidos
- b. Entretenimiento: TV, DVD, VCR, MP3, cámara, WebCam
- c. Telecomunicaciones: teléfono, teléfono móvil, intercomunicadores, fax
- d. Aparatos Electrodomésticos: microondas, heladera, reloj, horno, AA, luces
- e. Telemetría: metro utilitario, alarmas, termostatos, cámaras inalámbricas

2. Requisitos

- a. Los productos deben trabajar al 100% desde su compra
- b. La red y los dispositivos deben estar probados en operación
- c. Precio Bajo
- d. Mayor ancho de banda para soportar multimedia
- e. Mantener un formato de conexión para permitir agregar nuevos dispositivos
- f. Seguridad y Confianza

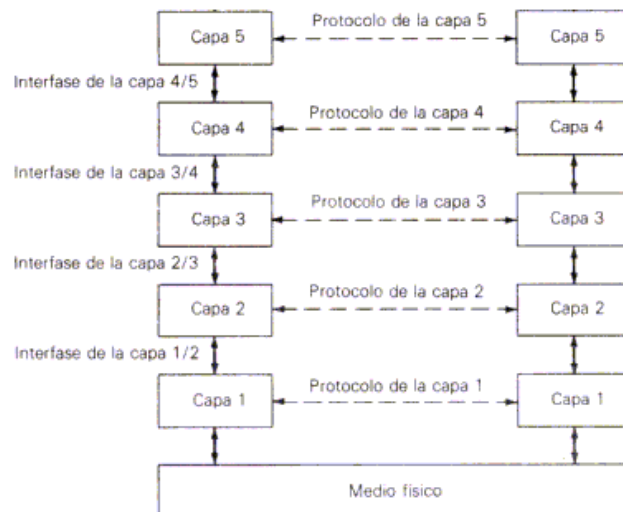
SOFTWARE DE REDES

Jerarquías de Protocolos

Protocolo: Acuerdo entre las partes en comunicación sobre cómo se debe llevar a cabo la comunicación

Arquitectura de Red: Conjunto de capas y protocolos. No incluye detalles ni especificaciones de la implementación

Pila de Protocolos: Todos los protocolos usados por un sistema



La mayoría de las redes está organizada como una pila de **capas** o **niveles**, cada una constituida a partir de la que está debajo de ella. El nº de capas, su nombre, contenido y función, difieren de red a red. El propósito de cada capa es ofrecer varios servicios a las capas superiores, a las cuales no se les muestran los detalles reales de implementación de los servicios ofrecidos.

La capa n de una máquina mantiene una conversación con la capa n de otra máquina. Las reglas y convenciones utilizadas en esta conversación, se conocen de manera colectiva como protocolo de capa n . Aquí juegan un papel importante los **encabezados** en los mensajes.

En realidad, los datos no se transfieren de manera directa desde la capa n de una máquina a la capa n de la otra, sino que cada capa pasa los datos y la información de control a la capa inmediatamente inferior, hasta que se alcanza la capa más baja.

Entre cada par de capas adyacentes, está una **interfaz**. Ésta define qué operaciones y servicios primitivos pone la capa más baja a disposición de la capa superior inmediata. Las interfaces deben estar bien definidas para que su desempeño sea óptimo y permitir independizar las capas.

Aspectos de diseño de las capas

- ✓ Cada capa necesita un mecanismo para identificar a los emisores y a los receptores
- ✓ Se necesita una forma de **direccionamiento** para precisar un destino específico entre varios
- ✓ Se necesitan reglas de la transferencia de datos: sentidos de comunicación, canales lógicos
- ✓ Control de Errores: códigos de detección/corrección de errores. El receptor debe tener algún medio para indicarle al emisor si recibió o no correctamente los mensajes
- ✓ Secuencia: El receptor debe poder unir las partes de un mensaje en el orden correcto
- ✓ Control de Flujo: que un receptor rápido no sature con datos a uno lento
- ✓ Mecanismos para desensamblar, transmitir y reensamblar mensajes
- ✓ Multiplexión y desmultiplexión de mensajes
- ✓ Enrutamiento

Servicios Orientados y No Orientados a la Conexión

Las capas pueden ofrecer 2 tipos de servicios a las capas que están por sobre ellas:

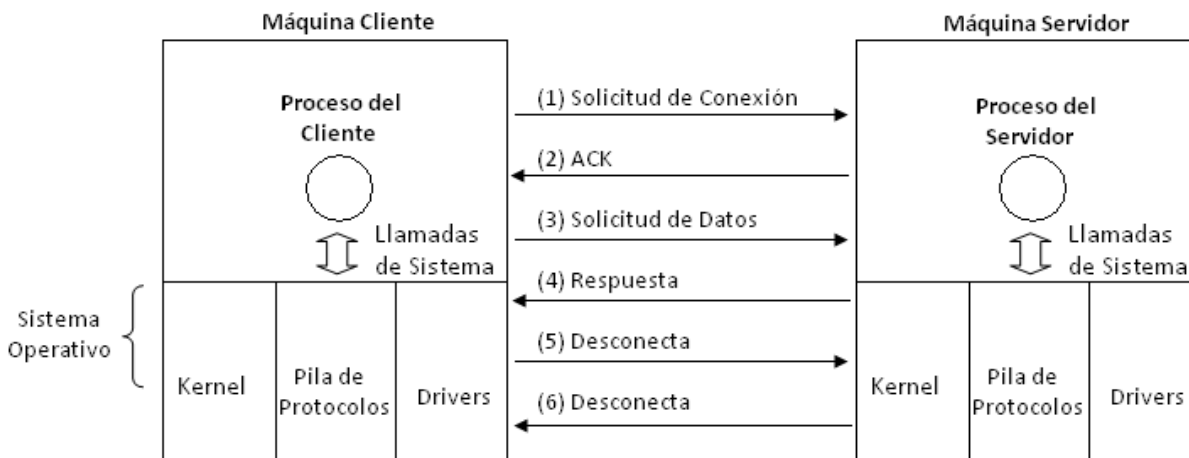
1. **Servicio Orientado a la conexión:** El usuario del servicio primero establece una conexión (negociación), la utiliza, y luego la libera. La conexión funciona como un tubo: el emisor empuja bits en un extremo, y el receptor los toma en el otro extremo. En la mayoría de los casos se conserva el orden de secuencia. Ej: [Sistema Telefónico](#). Según la calidad del servicio, se clasifican en:
 - a. Orientado a la conexión Confiable:
 - i. Secuencia de mensajes: Conserva los límites de mensajes
 - ii. Flujo de bytes: no conserva los límites de los mensajes. Ej: [Login Remoto](#)
 - b. Orientado a la conexión No confiable: No confirma la recepción. Ej: [VoIP](#)
2. **Servicio No orientado a la conexión:** Cada mensaje lleva completa la dirección del destino, y cada uno se enruta a través del sistema, independientemente del resto. Se utiliza principalmente en aplicaciones de tiempo real.
 - a. No Orientado a la conexión Confiable: Servicio de Datagramas Confirmados. Ej: [Correo Certificado](#)
 - b. No Orientado a la conexión No Confiable: Servicio de Datagramas. Ej: [Spam](#)
 - c. Solicitud-Respuesta: El emisor transmite un solo datagrama que contiene una solicitud, a continuación el servidor envía la respuesta. Ej: [Consulta de BD](#)

Primitivas de Servicio

Un servicio se especifica formalmente como un conjunto de **primitivas** disponibles a un proceso de usuario para que acceda al servicio. Estas primitivas le indican al servicio que desempeñe alguna acción o reporte sobre una acción que ha tomado una entidad igual. Si la pila de protocolos se ubica en el Sistema Operativo, por lo general las primitivas son llamadas al sistema.

Las primitivas de servicio orientado a la conexión difieren a las de un no orientado a la conexión.

Ejemplos de Primitivas para un servicio O.C.: [LISTEN](#), [CONNECT](#), [RECIEVE](#), [SEND](#), [DISCONNECT](#)



Servicio vs Protocolo

Servicio: Es un conjunto de primitivas (operaciones) que una capa proporciona a la capa que está sobre ella. No dice nada de cómo se implementan tales operaciones. Un servicio está relacionado con la interfaz entre 2 capas: la capa inferior provee un servicio, y la superior lo recibe.

Protocolo: Conjunto de reglas que rigen el formato y significado de los paquetes o mensajes, que se intercambiaron las entidades iguales en una capa. Las entidades utilizan protocolos para implementar sus definiciones del servicio. Son libres de cambiar sus protocolos cuando lo deseen, siempre y cuando no cambie el servicio visible a sus usuarios. De esta manera, el servicio y el protocolo son independientes entre sí.

MODELOS DE REFERENCIA

Modelo OSI (Interconexión de Sistemas Abiertos)

Es un modelo de 7 capas. Los **principios** usados para definir dichas capas fueron:

1. Una capa se debe crear donde se necesite una abstracción diferente
2. Cada capa debe realizar una función bien definida
3. La función de cada capa se debe elegir con la intención de definir protocolos estandarizados internacionalmente
4. Los límites de las capas se deben elegir a fin de minimizar el flujo de información a través de las interfaces
5. La cantidad de capas debe ser lo suficientemente grande para no tener que agrupar funcionalidades distintas en la misma capa, y lo bastante pequeña para que la arquitectura no se vuelva inmanejable

Esto dio por resultado las siguientes 7 capas:

- 1- **Capa Física:** Se lleva a cabo la transmisión de bits puros a través de un canal de comunicación. Respecto al diseño, se manejan interfaces eléctricas, temporización y el medio físico.
- 2- **Capa de Enlace de Datos:** Se encarga de transformar un medio de transmisión puro en una línea de comunicación que, al llegar a la capa de red, aparezca libre de errores de transmisión. Logra esto dividiendo la información del emisor en fragmentos llamados **tramas**, y transmitiéndolas secuencialmente. Si el servicio es confiable, se utilizan tramas

de confirmación de recepción. También define mecanismos de control de tráfico, y acceso al canal compartido (subcapa MAC).

- 3- **Capa de Red:** Controla las operaciones de la subred. Determina cómo se enrutan los paquetes del origen al destino y la QoS. Se deben resolver todos estos problemas para permitir que redes heterogéneas se conecten.

IMPORTANTE: Las capas anteriores operan encadenadas, y por lo general del lado del ISP. Las capas siguientes, operan de extremo a extremo y en la PC del cliente.

- 4- **Capa de Transporte:** Acepta los datos provenientes de capas superiores, los divide en unidades más pequeñas si es necesario, pasa éstos a la capa de red, y se asegura que todas las piezas lleguen correctamente al otro extremo. Todo debe hacerse con eficiencia, y de modo que cambios en el hardware no repercutan en capas superiores. La capa de transporte es una verdadera conexión de extremo a extremo, en toda la ruta del origen al destino.
- 5- **Capa de Sesión:** Permite que los usuarios de máquinas diferentes establezcan sesiones entre ellos. Las sesiones ofrecen varios servicios, como el control de diálogo, administración de token y sincronización.
- 6- **Capa de Presentación:** Le corresponde la sintaxis y la semántica de la información transmitida. A fin de que las computadoras con diferentes representaciones de datos se puedan comunicar.
- 7- **Capa de Aplicación:** Contiene varios protocolos que los usuarios requieren con frecuencia, como http, que es la base de WWW.

Modelo TCP/IP

Ante el temor del Departamento de Defensa de EEUU (DoD) de que algunos de sus valiosos hosts, routers y puertas de enlace de interredes fueran atacadas, surge el objetivo de que la red pudiera sobrevivir a la pérdida de hardware de la subred, sin que las conversaciones existentes se interrumpieran. En otras palabras, el DoD quería que las conexiones se mantuvieran intactas en tanto las máquinas de origen y destino estuvieran funcionando, aunque algunas de las máquinas o líneas de transmisión intermedias quedaran fuera de operación repentinamente. Además se necesitaba una arquitectura flexible debido a que se preveían aplicaciones con requerimientos diversos, desde transferencia de archivos a transmisión en tiempo real. Las capas son:

- 1- **Capa de Host a Red:** No está definida, y varía de una red a otra.
- 2- **Capa de interred:** Es no orientada a la conexión. Permite que los hosts inyecten paquetes dentro de cualquier red y estos viajen a su destino de manera independiente. Pueden llegar desordenados, en cuyo caso las capas superiores deberán ordenarlos. Esta capa define un paquete de formato y protocolo oficial llamado **IP (Protocolo de Internet)**.
- 3- **Capa de Transporte:** Permite una conexión extremo a extremo, como en OSI. Se definen 2 protocolos:
 - a. **TCP:** Orientado a la conexión, confiable, permite entregar un flujo de bytes sin errores a cualquier otra máquina de la interred. También maneja control de flujo
 - b. **UDP:** No orientado a la conexión y no confiable, para aplicaciones que no desean la secuencialización o control de flujo que TCP provee.
- 4- **Capa de Aplicación:** Contiene protocolos como Telnet, FTP, SMTP, DNS, http, etc.

OSI vs TCP/IP

- OSI hace distinción entre protocolos, servicios e interfaces. TCP/IP no.
- OSI se vislumbró antes que se inventaran los protocolos correspondientes. Con TCP/IP, los protocolos llegaron primero y a partir de ahí se ajustó el modelo. El problema fue que el modelo no aceptaba nuevas pilas de protocolos
- TCP/IP tiene 4 capas, y OSI tiene 7
- OSI soporta en la capa de red comunicación orientada y no orientada a la conexión, y sólo orientada a la conexión en la capa de transporte. TCP/IP en capa de red soporta sólo no orientada a la conexión, y ambos modos en la capa de transporte.
- El modelo OSI se usa, pero sus protocolos no. Lo contrario sucede con TCP/IP

Críticas al modelo OSI

1. Aparición Inoportuna
2. Mala Tecnología: 7 capas mal diseñadas, repetición de funciones y capas vacías
3. Malas Implementaciones
4. Malas Políticas: lo creó el Departamento de Defensa

Críticas al modelo TCP/IP

1. No distingue entre servicio, interfaz y protocolos, por lo que no es una guía para diseñar redes nuevas mediante tecnologías nuevas
2. TCP/IP no es general: sólo permite describir la pila de protocolos TCP/IP
3. La capa de host a red en realidad no es una capa, es una interfaz.
4. No distingue ni menciona capa física y de enlace de datos

REDES DE EJEMPLO

ARPANET

A fines de la década de 1950, durante el auge de la Guerra Fría, el DoD (Departamento de Defensa de EEUU) quería una red de control y comando que pudiera sobrevivir a una guerra nuclear. En esta época, todas las comunicaciones militares usaban la red telefónica pública. La vulnerabilidad de este sistema estaba en que la destrucción de alguna de las oficinas interurbanas clave podía fragmentar el sistema en muchas islas incomunicadas.

Hacia 1960, el DoD firmó un contrato con RAND Corp. para encontrar una solución. Paul Baran presentó un diseño de un sistema de conmutación distribuida. Este diseño le agradó al DOD, y pidieron a AT&T que construyera un prototipo, pero esta última compañía desechó la idea diciendo que era imposible de realizar.

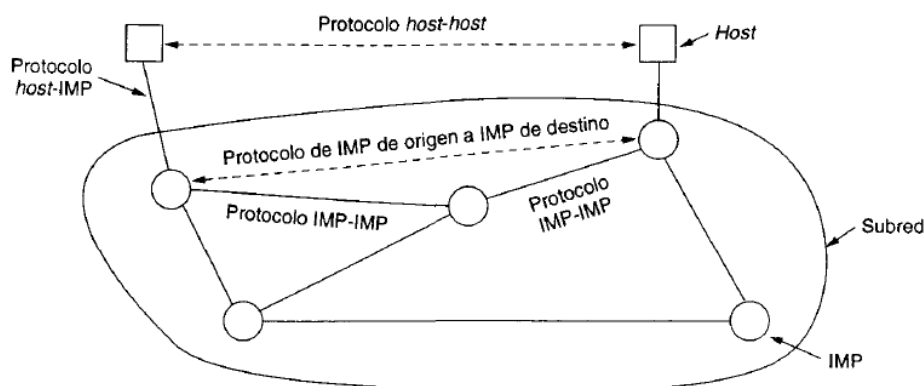
Cuando la URSS lanza el Sputnik, el Pentágono, para no quedarse atrás, crea una organización única de investigación para la defensa, ARPA (Agencia de Proyectos de Investigación Avanzada). ARPA hacía su trabajo otorgando subvenciones y contratos a universidades y empresas cuyas ideas le parecían prometedoras.

Para 1967, Wesley Clark, experto, sugirió la construcción de una subred de conmutación de paquetes, dando a cada host su propio enrutador. Roberts, director de ARPA, aceptó la idea, luego

de enterarse de que un sistema similar en Inglaterra llamado NPL funcionara, y citara las ideas de Baran.

La subred contaría de minicomputadoras llamadas **IMPs (Procesadores de Mensajes de Interfaz)**, conectadas por líneas de transmisión de 56kbps. Para alta confiabilidad, cada IMP estaría conectado al menos a otros 2 IMPs. La subred iba a ser de datagramas, almacenamiento y reenvío.

Cada nodo de la red iba a constar de un IMP y un host, conectados por un cable corto. Un host tendría la capacidad de enviar mensajes a su IMP, que fragmentaría y los reenviaría de manera independiente al destino. Los protocolos que se usaron se muestran en la figura.



Como los hosts también necesitaban software, se convocó a estudiantes de universidades, que poco a poco fueron conectando sus campus entre sí.

Este experimento demostró que los protocolos usados en ARPANET no eran los adecuados para ejecutarse a través de varias redes. Luego de investigaciones, se culminó en TCP/IP.

Durante la década de 1980, varias redes se unieron a ARPANET, por lo que encontrar hosts se volvió muy complicado. Esto dio pie al surgimiento de DNS (Servidor de Nombres de Dominio).

NFSNET

A fines de 1970, la NFS (Fundación Nacional para las Ciencias de EEUU) vio el enorme impacto de ARPANET. Sin embargo, para estar en ARPANET, una universidad debía tener un contrato de investigación con el DoD, lo cual muchas no tenían. La respuesta de la NFS fue diseñar un sucesor de ARPANET, abierto a todos los grupos de investigación de universidades.

Se construyó una red dorsal (o troncal) para conectar 6 centros de supercomputadoras. Cada supercomputador tenía un "hermano menor", **fuzzball**. Estas computadoras estaban conectadas a líneas alquiladas de 56 kbps y formaban una subred, con el mismo hardware que ARPANET, pero con TCP/IP. Toda la red (dorsal y regionales conectadas a la dorsal) se llamó **NFSNET**. Este se conectó a ARPANET mediante un enlace entre un IMP y una **fuzzball**.

NFSNET fue un éxito y pronto se saturó. El gobierno se dio cuenta de que no podía financiar por siempre el uso de redes. Además, las empresas comerciales se querían unir, pero los estatutos de la NFS les prohibían utilizar las redes. Por esto, NFS alentó a formar una corporación no lucrativa, ANS (Redes y Servicios Avanzados), como el 1^{er} paso a la comercialización. En 1990 ANS adquirió a NFSNET, escaló los enlaces a 45 Mbps, y formó **ANSNET**.

Para permitir la interconexión de redes regionales, NFS concedió contratos a 4 diferentes operadores de redes para establecer un **NAP (Punto de Acceso a la Red)**. Todo operador de red que quisiera proporcionar servicio de red dorsal a las redes regionales, debía conectarse a todos los NAPs. Con esto, se dio pie a la comercialización, competencia por calidad de servicio y costos.

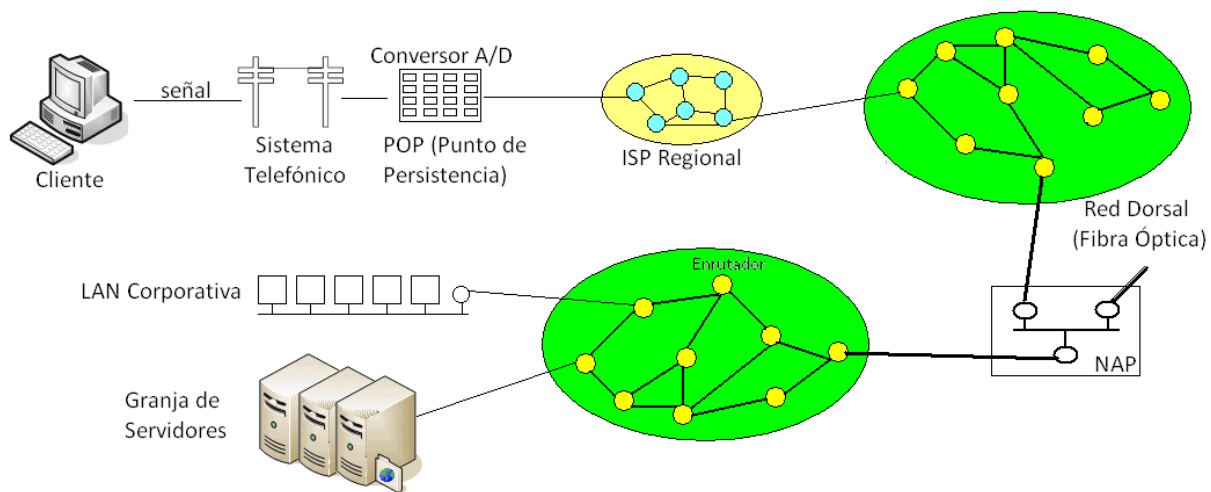
INTERNET

TPC/IP se convirtió en el protocolo oficial en 1983. A mediados de 1980, las personas empezaron a ver el conjunto de redes como una interred y más tarde como Internet, aunque no hubo una inauguración oficial.

Una máquina está en internet si ejecuta la pila de protocolos TCP/IP, tiene una dirección IP y puede enviar paquetes IP a todas las demás máquinas en Internet.

Tradicionalmente, Internet tenía 4 aplicaciones: e-mail, noticias, login remoto y transferencia de archivos. En 1990, el surgimiento de WWW atrajo a millones de usuarios no académicos a la web, ya que hizo a internet más fácil de usar. También ayudó a esto el surgimiento de varios ISP.

INTERNET HOY



Redes Orientadas a la Conexión: X.25, Frame Relay y ATM

X.25 fue la 1ª red de datos pública. Para utilizarla, una computadora establecía primero una conexión con la computadora remota, es decir, hacía una llamada telefónica. Esta conexión daba un nº de conexión para utilizarlo en los paquetes al transferir datos. En 1980, se reemplazó con **Frame Relay**, una red de entrega de paquetes en orden, sin control de errores ni de flujo.

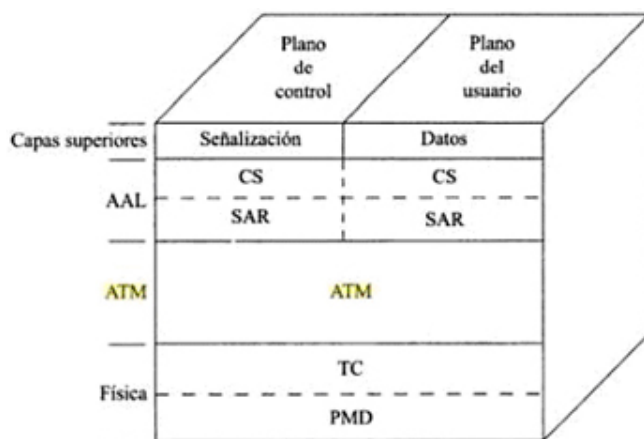
ATM (Modo de Transferencia Asincrónica) se diseñó a fines de 1990. Iba a resolver todos los problemas de conectividad y telecomunicaciones fusionando voz, datos, TV, etc., en un solo sistema integrado. Eso no sucedió, por motivos similares a los por que el OSI no funcionó. Actualmente, ATM tiene un uso profundo dentro del sistema telefónico, con frecuencia en el transporte de paquetes IP, de manera interna, por lo que los usuarios no se percatan de su uso.

A las conexiones ATM se las conoce como **circuitos virtuales**, y tienen una variante, **circuitos virtuales permanentes**, que son conexiones permanentes entre 2 hosts distantes. Cada conexión, temporal o permanente, tiene un identificador de conexión.

Una vez establecida la conexión, se puede comenzar a transmitir datos, en paquetes pequeños de tamaño fijo (53 bytes) llamados **celdas**. Dado su pequeño tamaño, la conmutación de celdas se puede hacer en hardware a gran velocidad, garantizando la QoS. Los paquetes de longitud variable IP se deben enrutar por software, algo mucho más lento. Todas las celdas siguen la misma ruta al destino. La entrega de celdas no está garantizada, pero el orden sí (no es una garantía perfecta, pero es mejor que Internet).

Modelo de Referencia ATM

1. **Capa Física:** ATM sólo especifica que las celdas se puede enviar tal cual por cable o fibra, pero también se pueden empacar dentro de la carga útil de otros sistemas de transporte. En otras palabras, ATM es independiente del medio de transmisión. Tiene 2 subcapas
 - a) PMD (Dependiente del Medio Físico): Interactúa con el cable real
 - b) TC (Convergencia de Transmisión): Maneja todos los aspectos relacionados con las indicaciones de dónde empiezan y terminan las celdas en el flujo de bits
2. **Capa ATM:** Se encarga de las celdas y su transporte. Define la disposición de una celda y su encabezado. Tiene que ver con el establecimiento y liberación de circuitos virtuales, y control de congestión
3. **Adaptación ATM (AAL):** Permite que los usuarios envíen paquetes más grandes que una celda, para luego segmentarlos y enviarlos
 - a) SAR (Segmentación y Reensambla): Fragmenta/reensambla paquetes en celdas
 - b) CS (Subcapa de Convergencia): Permite que los sistemas ATM ofrezcan diversos servicios a diferentes aplicaciones



El modelo ATM se define como si fuera tridimensional. El **plano de usuario** trata con el transporte de datos, control de flujo, corrección de errores y otras funciones de usuario. El **plano de control** se ocupa de la administración de la conexión

LAN's Inalámbricas: 802.11 (WiFi)

El estándar propuesto debía trabajar en 2 modos: en presencia de una estación base (punto de acceso), y sin ella (ad-hoc). Había que enfrentar varios retos: encontrar una banda adecuada de preferencia mundial, enfrentar el hecho de que las ondas tienen rango finito, mantener la privacidad de los usuarios, tener en cuenta la vida limitada de las baterías, etc.

El comité decidió hacer que 802.11 fuera compatible con Ethernet en la capa de Enlace de datos. Sin embargo, en la capa física y de enlace, surgieron varias diferencias:

1. Una PC en Ethernet siempre escucha el medio antes de transmitir. Si está inactivo, transmite. Esto no funciona en LANs Inalámbricas (Problema Estación Oculta/Expuesta)
2. Desvanecimiento por múltiples trayectorias: los objetos sólidos pueden reflejar una señal de radio, haciendo que se reciba múltiples veces
3. Gran cantidad de software no considera la movilidad ([Ej: Lista de impresoras activas](#))
4. Si una computadora portátil se mueve lejos de la estación base que está usando y dentro del rango de una estación base diferente, se requiere algún tipo de manejo.

La conexión entre 802.11 y el mundo exterior se conoce como **portal**.

UNIDAD 2: La Capa Física

Esta capa es la más baja del modelo, y define las interfaces mecánica, eléctrica y de temporización de la red.

ANÁLISIS DE FOURIER

Mediante la variación de algunas propiedades físicas, como el voltaje o la corriente, es posible transmitir información a través de los cables.

Fourier descubrió que cualquier función periódica de comportamiento razonable, $g(t)$ con un periodo T , se puede construir sumando la cantidad (posiblemente infinita) de senos y cosenos, donde $f=1/T$ es la frecuencia fundamental, a_n y b_n son las amplitudes de seno y coseno de los n -ésimos (términos) armónicos, y c es una constante. Tal descomposición se conoce como serie de Fourier. Esta serie permite representar señales.

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \cdot \text{sen}(2\pi nft)$$

Cualquier instalación para la transmisión disminuye las componentes de Fourier en distinto grado, lo que provoca distorsión. Por lo general, las amplitudes se transmiten sin ninguna disminución desde 0 hasta cierta frecuencia f_c [Hz], y todas las frecuencias que se encuentren por arriba de esta frecuencia de corte serán atenuadas. El rango de frecuencias que se transmiten sin atenuarse con fuerza se conoce como **ancho de banda**.

TASA DE DATOS MÁXIMA DE UN CANAL

En 1924, Nyquist se dio cuenta de que incluso un canal perfecto tiene una capacidad de transmisión finita. Derivó una ecuación que expresa la tasa de datos máxima para un canal sin ruido de ancho de banda finito:

$C = 2 \cdot B \cdot \log_2 M$, donde C =capacidad del canal; B =Ancho de banda; $M=2^n$ = Tasa señalización (bits)

En 1948, Shannon continuó el trabajo de Nyquist y lo extendió al caso de un canal sujeto a ruido aleatorio:

$C = B \cdot \log_2 (1+S/N)$, donde S/N = Relación señal a ruido (dB)

MEDIOS DE TRANSMISIÓN GUIADOS

1. **Medios Magnéticos:** Es el almacenamiento en cintas magnéticas o medios extraíbles (Ej: DVD). Es más rentable para aplicaciones en las que un ancho de banda alto o el costo por bit transportado es un factor clave. Tiene un gran ancho de banda, pero un retardo y tiempo de transmisión muy pobre.
2. **Par Trenzado Sin Blindaje (UTP):** Son 2 alambres de cobre aislados, por lo general de 1mm de grueso, que se trenzan en forma helicoidal. La distancia que se puede recorrer con estos cables es de varios kilómetros sin la necesidad de amplificar las señales. Se pueden usar tanto para la transmisión analógica como digital. El ancho de banda depende del grosor del cable y la distancia que recorre. Hay varias categorías de UTP:
 - a. Categoría 3: 2 alambres aislados en una envoltura protectora. Se usa en teléfonos.
 - b. Categoría 5: Similares a los CAT3, pero con más vueltas/cm, lo que mejora la calidad de la señal al tener una menor diafonía. Maneja señales de 100 MHz.
 - c. Categoría 6, 7: Manejan señales de 250/600 MHz.

3. **Cable Coaxial:** Tiene mejor blindaje que el par trenzado, por lo que puede alcanzar tramos más largos a mayores distancias. Has 2 clases, el de 50 ohms usado para transmisión digital, y el de 75 ohms usado para la transmisión analógica y TV por cable.



4. **Fibra Óptica:** Ésta tecnología permite alcanzar anchos de banda por encima de los 50 Tbps, y se sigue mejorando. El límite actual de señalización está impuesto por la rapidez para convertir señales eléctricas a ópticas, aunque se han alcanzado velocidades de 100 Gbps en una sola fibra.



Un sistema de transmisión óptico tiene 3 componentes: la fuente de luz, el medio de transmisión y el detector. El sistema de transmisión tendría pérdidas, de no ser por el principio físico que hace que los haces de luz se “doblen” (refracten) en la frontera entre el sílice y el aire, dentro de la fibra. La longitud de los pulsos de luz transmitidos por una fibra aumenta conforme se propagan (dispersión cromática). La atenuación de la luz dentro del vidrio depende de la longitud de onda de la luz, y de algunas propiedades físicas del vidrio.

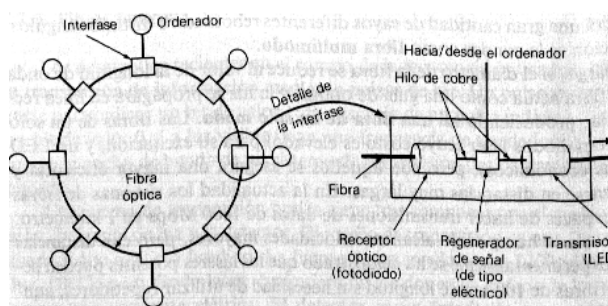
Hay 3 tipos de Fibras: multimodo de índice gradual y discontinuo, y monomodo.

Para las comunicaciones, se utilizan 3 bandas de longitud de onda, las cuales tienen muy baja atenuación: 0.85, 1.30 y 1.55 micras. Tienen una anchura entre 25.000 y 30.000 GHz.

Long. Onda [mm]	850	1300	1550
Tipo Fibra Óptica	Multimodo	Multimodo	Monomodo
Aplicaciones	LAN	FDDI	WAN
Emisor (Electricidad a luz)	Led	Led	Laser
Receptor (Fotodiodo)	PIN	APD	APD
Empalmes	Mecánico	Mecánico	Fusión
Distancia	< 1 Km	< 3 Km	> 50 Km
Velocidad	Mbps	Mbps	Gbps, Tbps
Protocolo	Ethernet	FR	SONET, ATM
\$	<< (Económico)	< (Medio)	>> (Muy Caro)

Interfaces en redes de Fibra óptica

- Interfaz Pasiva: Consiste en 2 derivaciones fusionadas a la fibra principal. Una derivación se usa para transmitir y la otra para recibir. Un LED/fotodiodo decompuesto no rompe el anillo, sólo deja afuera a una computadora.
- Repetidor Activo:



- Estrella Pasiva: Es análogo a conectar varias PC a un hub central

FIBRA VS COBRE	Fibra	Cobre
Ancho Banda	Mayores	Menores
Atenuación (repetidores cada)	50km	5km
Afectado por interferencia EM, cortes de energía, corrosión	No	Si
Peso y tamaño	Bajo	Alto
Es fácil "colgarse" / interferirlas	No	Si
Dificultad de Instalación	Alta	Baja
Forma Transmisión	Unidireccional	Bidireccional
Costo	>>>	>

MEDIOS DE TRANSMISIÓN NO GUIADOS (INALÁMBRICOS)

Frecuencia ($f = [\text{Hz}]$): Es la cantidad de ciclos por unidad de tiempo que tiene una onda sinusoidal.

$$\lambda f = C \text{ (en el vacío)}$$

Longitud de onda ($\lambda = [\text{m}]$): longitud entre 2 máximos o mínimos consecutivos.

Velocidad de la Luz (c) = 3×10^8 m/s

Las porciones del Espectro Electromagnético, radio, microondas, infrarrojo y luz visible, pueden servir para transmitir información modulando la amplitud, frecuencia o fase de las ondas. La luz UV, rayos X y gamma serían todavía mejores, debido a sus frecuencias más altas, pero son difíciles de producir y modular, no se propagan bien entre edificios, y son peligrosos para los seres vivos.

La cantidad de información que puede transportar una onda electromagnética se relaciona con su ancho de banda. Con la tecnología actual, es posible codificar varios bits/Hertz a frecuencias bajas, pero a frecuencias altas sólo se puede llegar a 8.

La mayoría de las transmisiones ocupa una banda de frecuencias estrecha ($\Delta f/f \ll 1$) a fin de obtener la mejor recepción (muchos Watts/Hz). Sin embargo, en algunos casos se utiliza una banda ancha con 2 variaciones:

- A) Espectro disperso con salto de frecuencia:** El transmisor salta de frecuencia cientos de veces por segundo. Es popular en la comunicación militar, para evitar interferencias. Las señales reflejadas siguen una trayectoria más larga, y llegan más tarde, lo que es bueno para evitar el desvanecimiento por múltiples trayectorias. Es usado por 802.11 y Bluetooth
- B) Espectro disperso de Secuencia directa:** Dispersa la señal a través de una banda de frecuencia ancha

Supondremos que en todas las transmisiones se utiliza una banda de frecuencias estrecha.

- 1. Radio-transmisión:** Las ondas de radio son fáciles de generar, omnidireccionales, pueden viajar distancias largas y penetrar edificios. Las propiedades de las ondas de radio dependen de la frecuencia:
 - a. Frecuencias Bajas: Cruzan cualquier obstáculo, pero la potencia se reduce drásticamente a medida que nos alejamos de la fuente.
 - b. Frecuencias Altas: Las ondas viajan en línea recta, rebotan en obstáculos y son absorbidas por la lluvia.

En todas las frecuencias, las ondas de radio están sujetas a interferencias por equipos eléctricos y motores. El problema principal al usar bandas para comunicación de datos es su ancho de banda bajo.

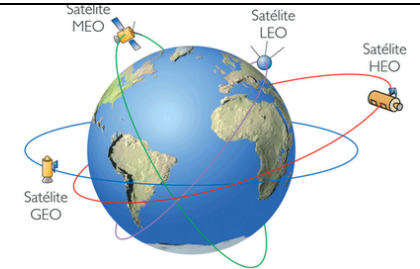
2. **Microondas:** Por encima de los 100 MHz, las ondas viajan en línea recta, lo que requiere a las antenas alineadas. Cuanto más alto sean las torres, más separadas pueden estar. Las microondas no atraviesan bien los edificios, y sufren desvanecimiento por múltiples trayectorias, en especial cuando hay mal clima. Se ha usado tanto para la comunicación a larga distancia, celulares, TV, etc., que el espectro se ha vuelto muy escaso. Las ventajas respecto a la fibra, es que no necesita derecho de paso y es relativamente barato.

Para evitar el caos, hay acuerdos nacionales e internacionales respecto a la asignación del espectro. También se puede subastar el espectro, o dejar usar a voluntad, pero regulando la potencia utilizada para evitar interferencias.

3. **Ondas Infrarrojas y Milimétricas:** Son usados por equipos como controles de TV, estéreos. Son direccionales, económicos y fáciles de construir. Sin embargo, no atraviesan objetos sólidos (lo cual también puede ser una ventaja). No requiere de licencia para operarlo.
4. **Ondas de Luz (Láser):** Es unidireccional, de ancho de banda muy alto, muy bajo costo, fácil de instalar y no requiere licencia. Sin embargo, al ser un haz muy estrecho, se limita el alcance, por lo que suelen utilizar lentes para desenfocar el rayo. No pueden penetrar la lluvia ni niebla densa.

SATÉLITES DE COMUNICACIONES (Difusión)

Altitud (km)	Tipo	Latencia (ms)	Satélites necesarios para abarcar toda la tierra
35.800	GEO	270	3
20000-GEO	MEO	35-85	10
1000-20000	HEO	Variable por órbita elíptica	
100-1000	LEO	1-7	50



Los **satélites GEO (Órbita Terrestre Geoestacionaria)**, por su posición en una órbita ecuatorial circular, aparentarían estar fijos en el cielo, por lo que se les da preferencia. Con la tecnología actual es aconsejable no usar satélites GEO espaciados a menos de 2° en el plano ecuatorial para evitar interferencias; esto implica que puede haber 180 satélites a la vez en el cielo. Sin embargo, se pueden usar múltiples frecuencias y polarizaciones para incrementar el ancho de banda disponible. Los satélites duran unos 10 años, ya que utilizan combustible para ajustar su posición.

Las posiciones orbitales las asigna la ITU, al igual que las bandas de frecuencia.

Banda	Enlace Descendente	Enlace Ascendente	Ancho Banda (BW)	Problemas
L	1.5 GHz	1.6 GHz	15 MHz	Bajo Ancho de Banda; Saturada
S	1.9 GHz	2.2 GHz	70 MHz	
C	4.0 GHz	6.0	500 MHz	Interferencia terrestre
Ku	11 GHz	14 GHz	500 MHz	Lluvia
Ka	20 GHz	30 GHz	3500 MHz	Lluvia; Valor Equipo

Se han desarrollado micro-estaciones de bajo costo, **VSATs**, de 1m o menos ([Ej: Antena Direct TV](#)). En muchos sistemas VSAT, las micro-estaciones no tienen suficiente potencia para comunicarse entre ellas (a través del satélite). En vez de ello, es necesaria una estación central (en tierra) que retransmite el tráfico entre VSATs. La desventaja, es el retardo producido.

Los satélites son un verdadero desastre en cuanto a seguridad y privacidad. Otra propiedad es que el costo de transmitir un mensaje es independiente de la distancia que se recorra.

Los **Satélites MEO** se desplazan lentamente y tardan aproximadamente 6hs para dar la vuelta a la tierra, por lo que se necesita rastrearlos conforme se desplazan. [Ej: Los 24 Satélites GPS](#)

Los **Satélites LEO**, debido a la rapidez de su movimiento, se necesitan grandes cantidades de ellos para cubrir la tierra. Como están cerca de la tierra, las estaciones terrestres no necesitan mucha potencia, y el retardo de las señales es de unos pocos milisegundos. Algunos ejemplos de LEO son:

- 1- Iridium: El propósito era que tan pronto como un satélite se perdiera de vista, otro lo reemplazara. Los socios los lanzaron en 1997; en 1999 quebró, y se retomó en el 2001. El negocio de Iridium es ofrecer servicio de telecomunicaciones en todo el mundo a través de dispositivos de bolsillo. Sus clientes son industrias marítimas, aviación, etc. Están a 750km, en órbitas polares circulares. Hay satélites cada 32º de latitud, con 48 celdas y 3840 canales cada uno. La comunicación de los satélites tiene lugar en el espacio
- 2- GlobalStar: Son 48 satélites, pero la comunicación entre ellos es en tierra, por lo que mucha complejidad se queda en tierra, donde es más sencillo manejarla. Permite la utilización de teléfonos de baja potencia.
- 3- Teledesic (proyecto): Está destinada a usuarios de Internet. Tiene un canal ascendente de hasta 100 Mbps, y uno descendente de hasta 720 Mbps mediante antenas VSAT. Funciona en Banda Ka. Los 30 satélites, están dispuestos a 1350km, con grandes huellas cada uno.

Los satélites tienen unos nichos de mercado importantes a los cuales la Fibra óptica no se dirige:

- A pesar de que la FO tiene más ancho de banda potencial que los satélites, este ancho de banda no está disponible para la mayoría de los usuarios (es compartido)
- Comunicación móvil
- Situaciones en las que se requiere difusión
- Comunicaciones en lugares agrestes o con una infraestructura terrestre pobremente desarrollada
- Lugares donde no se puede conseguir derecho para instalar fibra óptica
- Cuando se necesita un despegue rápido, como por ejemplo, una guerra.

LA RED TELEFÓNICA PÚBLICA CONMUTADA (PSTN)

Debido a que el costo de conectar mediante un cable varias computadoras distantes es prohibitivo, sin mencionar que tender líneas de transmisión privadas bajo propiedades públicas o ajenas es un delito, surge la red telefónica pública como medio para interconectar redes.

Sin embargo, la diferencia de velocidades entre la PSTN y una red UTP es de varios órdenes de magnitud (56kbps vs 10^9 bps).

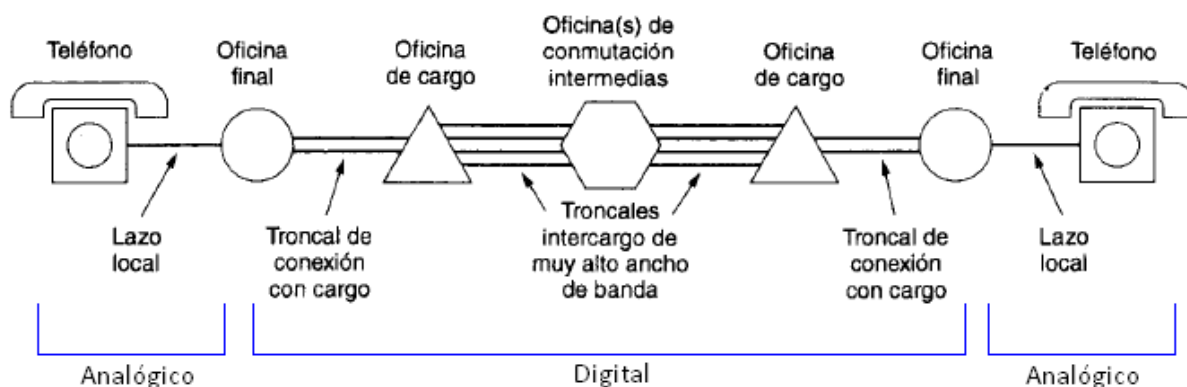
Estructura del Sistema Telefónico



Para 1890, las 3 partes principales del Sistema Telefónico ya estaban en su lugar: las oficinas de conmutación, los cables entre los clientes, y las conexiones de larga distancia entre oficinas de conmutación. El modelo de Bell se ha variado un poco, pero en esencia es el mismo.

Cada teléfono tiene 2 alambres de cobre que van directamente a la **oficina central local** de la compañía telefónica (1-10km). En el ámbito de las comunicaciones, las conexiones de 2 alambres entre el teléfono de cada suscriptor y la oficina central se conocen como **circuito local**.

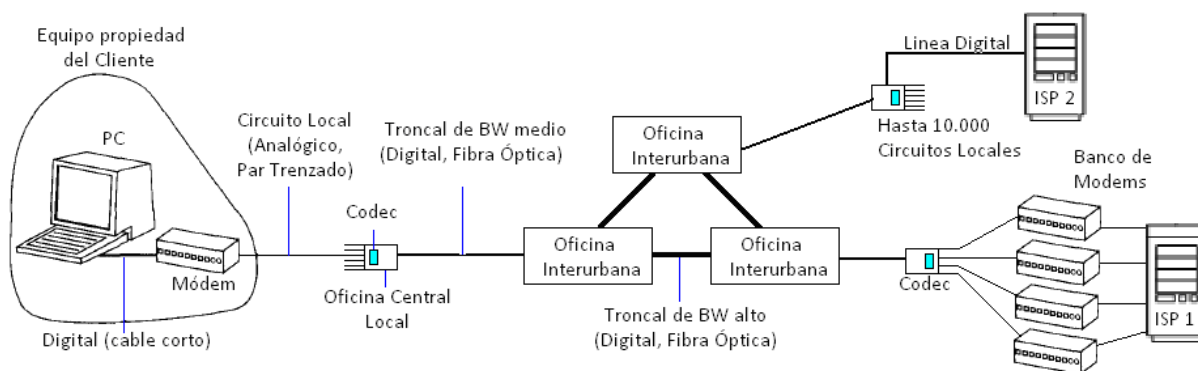
Si un suscriptor conectado a una oficina central determinada llama a otro conectado a la misma oficina central, el mecanismo de conmutación dentro de la oficina establece una conexión directa entre los 2 circuitos locales. Esta conexión permanece intacta mientras dure la llamada.



En Síntesis, el sistema telefónico tiene 3 componentes principales:

1. **Circuitos locales:** cables de par trenzado que van hacia las casas y las empresas
2. **Troncales:** Fibra Óptica Digital que conecta oficinas de conmutación
3. **Oficinas de Conmutación:** donde las llamadas pasan de una troncal a otra

El Circuito Local: módems, ADSL e Inalámbrico



Cuando una computadora desea enviar datos digitales sobre una línea analógica de acceso telefónico, es necesario convertir primero los datos a formato analógico para transmitirlos sobre el circuito local. Esta tarea la realiza el **MODEM**. Los datos se convierten a digital en la oficina central de la compañía telefónica para transmitirlos sobre las troncales.

Si en el otro extremo hay una PC con un módem, es necesario realizar la conversión inversa (D/A) para recorrer el circuito local en el destino.

Modular vs Codificar: Modular es para adaptar una señal digital a la red telefónica, modificando la amplitud, frecuencia o fase; Codificar es tomar muestras de una señal y representar su valor con pulsos.

Las líneas de transmisión tienen 3 problemas principales:

- **Atenuación:** es la pérdida de energía conforme la señal se propaga hacia su destino. Depende de la frecuencia y se expresa en dB/km
- **Distorsión:** es el efecto producido por la diferencia de velocidad de propagación de las distintas componentes de Fourier
- **Ruido**

Modems

La señalización de corriente alterna (CA) se utiliza para superar los problemas asociados a la señalización de CC, en especial en las líneas telefónicas. Se introduce un tono continuo en el rango de 1000-2000 Hz, llamado **portadora de onda sinusoidal**, cuya amplitud, frecuencia o fase se pueden modular para transmitir la información.

En **MODEM** es un dispositivo que acepta un flujo de bits en serie como entrada y que produce una portadora modulada mediante uno (o más) de estos métodos, o viceversa. El MODEM se conecta entre la PC (digital) y el sistema telefónico (analógico).

Tasa de baudios (o de símbolos): es el número de unidades de señal por segundo. Un baudio puede contener varios bits. Es importante resaltar que no se debe confundir el baud rate o velocidad en baudios con el bit rate o velocidad en bits por segundo, ya que cada evento de señalización (símbolo) transmitido puede transportar uno o más bits. Sólo cuando cada evento de señalización (símbolo) transporta un solo bit coinciden la velocidad de transmisión de datos baudios y en bits por segundo.

Tasa De Bits: Define el número de bits que se transmiten por unidad de tiempo a través de un sistema de transmisión digital. Así pues, es la velocidad de transferencia de datos.

Diagramas de Constelación: Son diagramas que muestran las combinaciones permitidas de amplitud y fase. Cuando hay muchos puntos en un diagrama de constelación, incluso la cantidad mínima de ruido en la amplitud o fase puede dar como resultado un error, y potencialmente, muchos bits malos. Para reducir errores, se desarrollaron esquemas TCM (Modulación por Codificación de malla)

Líneas de Suscripción

Conforme el acceso a Internet se tornaba una parte importante de su negocio, las **compañías telefónicas (LECs)** se dieron cuenta de que necesitaban un producto más competitivo. En principio, había muchas ofertas que se traslapaban, todas bajo el nombre de **xDSL (Línea Digital de Suscriptor)**, de las cuales **ADSL** fue la más popular.

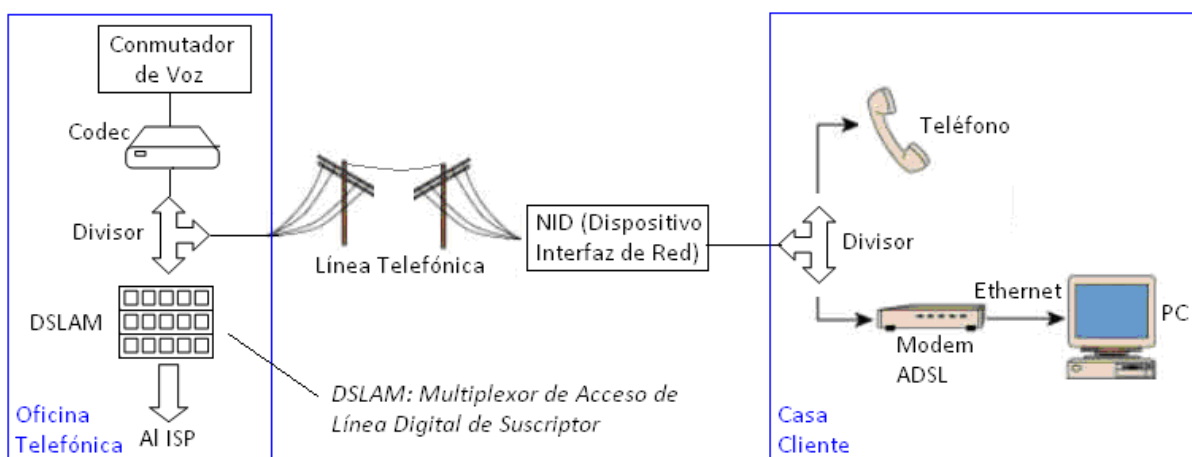
En el lugar donde cada circuito local termina en la oficina central, el cable pasa a través de un filtro que atenúa todas las frecuencias bajo los 300 Hz y sobre los 3400 Hz. Cuando un cliente se suscribe al servicio, la línea de entrada se conecta a un tipo distinto de conmutador, que no cuenta con el filtro, por lo que toda la capacidad del circuito local queda disponible. En esta situación, el ancho de banda artificial de 3100 Hz generado por el filtro ya no es el factor limitante, sino el medio físico del circuito local.

Debido a la atenuación de la señal en el cable UTP-3, las compañías, cuando eligen la velocidad que ofrecerán, al mismo tiempo eligen un radio a partir de sus oficinas centrales mas allá del cual no pueden proporcionar servicio. Aquí es donde se encuentran los negocios y la tecnología.

Todos los servicios xDSL se diseñaron para que cumplieran unos objetivos. Primero, debían funcionar en los circuitos locales existentes UTP3. Segundo, no debían afectar los teléfonos ni fax. Tercero, debían superar por mucho los 56 kbps. Cuarto, siempre debían funcionar con solo una tarifa mensual, no por minuto.

AT&T hizo la oferta inicial de ADSL, dividiendo en espectro en 3 bandas de frecuencia: **POTS (Servicio Telefónico Convencional)**, canal ascendente y canal descendente. Se usó multiplexión por división de frecuencia.

Existió otro enfoque, **DMT (MultiTono Discreto)**, que tenía 256 canales: el 0 para POTS, 1-5 de banda de resguardo, uno para control de flujo ascendente, otro para descendente, y el resto para datos de usuario. Queda a cargo del proveedor determinar cuántos canales se utilizarán para flujo ascendente y cuantos para descendente. La mayoría de los proveedores asigna un 80% al canal descendente, debido a que la mayoría de los usuarios descargan más de lo que suben. Esto dio lugar a la "A" (Asimétrica) de ADSL.



El problema del diseño anterior, es que requiere la instalación del NID y del divisor en la residencia del cliente. Hay un diseño alternativo sin divisor, en el cual se instala en cada teléfono de la casa un microfiltro ([Ej: Speedy](#))

Circuitos Locales Inalámbricos (WLL)

Consiste en instalar una antena direccional en el techo de la vivienda, a la cual llegan las señales provenientes de una antena colocada en una torre del ISP. Hay 2 principales:

1. **MMDS (Servicio de Distribución Multipunto y Multicanal):** Posee un rango de 50km y puede propagarse por vegetación y lluvia moderadamente bien. El equipo es fácil de conseguir y económico, pero tiene un bajo ancho de banda
2. **LMDS (Servicio Local de Distribución Multipunto):** Posee un rango de 2-5km, trabaja con ondas milimétricas, lo que requiere línea visual entre la antena del ISP y la del cliente. Además, le cuesta propagarse a la señal por entre la lluvia y hojas. Se estandarizó en 802.16 como **MAN Inalámbrica**.

Multiplexión en las troncales

- **FDM (Multiplexión por división de Frecuencia)**
- **WDM (Multiplexión por división de longitud de Onda):** Se usa en Fibra óptica. Aquí, n fibras se juntan en un combinador óptico, cada una con su energía presente a diferentes longitudes de onda. Los n haces se combinan en una sola fibra compartida para transmisión a un destino distante. En el extremo distante, el haz se divide en tantas fibras como hayan entrado. Siempre que cada canal tenga su propio rango de frecuencias y todos los intervalos estén separados, se pueden multiplexar juntos y ser muy confiable. Al ejecutar muchos canales en paralelo sobre diferentes longitudes de onda, el ancho de banda agregado se incrementa de manera lineal de acuerdo con el nº de canales. Hay una variante cuando el nº de canales es muy grande, **DWDM (WDM Densa)**.
- **TDM (Multiplexión por división del Tiempo):** Se usa sólo para datos digitales, ya que no se puede usar en circuitos locales salvo previa conversión a digital. Esta conversión se realiza con un dispositivo llamado **codec**, con lo que se produce una serie de nº de 8 bits, a 8000 muestras/segundo, capturamos los 4 KHz de ancho de banda del canal telefónico. Otras técnicas de modulación son:
 - **PCM (Modulación por Codificación de Impulsos):** Se muestrea una señal, y se transmite la amplitud de la misma
 - **Modulación Delta:** Transmite la diferencia entre la amplitud actual y la siguiente
 - **Codificación por predicción**

SONET/SDH

SONET define una tecnología para transportar muchas señales de diferentes capacidades a través de una jerarquía óptica síncrona y flexible. Esto se logra por medio de un esquema de multiplexado por interpolación de bytes. La interpolación de bytes simplifica la multiplexación y ofrece una administración de la red extremo a extremo.

CONMUTACIÓN

- **De Mensajes:** No se establece una trayectoria por adelantado. Cuando el emisor tiene datos que enviar, se envían de centrales en centrales, un salto a la vez, donde se verifica si tiene errores o se puede retransmitir. Ya no se utiliza.
- **De Paquetes**
- **De Circuitos**

Elemento	C. Circuitos	C. Paquetes
Establecimiento de la llamada	Requerido	No es Necesario
Trayectoria física detallada	Si	No
Cada paquete puede seguir la misma trayectoria	Si	No
Los paquetes llegan en orden	Si	No
Es una falla de conmutación fatal	Si	No
Ancho de Banda disponible	Fijo	Dinámico
Cuando puede haber congestión	Durante Establecimiento	En cada paquete
BW potencialmente desperdiciado	Si	No
Almacenamiento y Reenvío	No	Si
Transparencia	Si	No
Cargos, Tarifa	Por minuto	Por paquete

EL SISTEMA TELEFÓNICO MÓVIL

Los teléfonos inalámbricos se dividen en 2 categorías básicas: **teléfonos inalámbricos** y **teléfonos móviles (celulares)**. Los primeros son dispositivos que consisten en una estación base y un teléfono que pueden usarse dentro de una casa. Los segundos son los que trataremos

Teléfonos Móviles de 1ª Generación: Voz Analógica

En 1946, había un sistema que utilizaba un solo transmisor grande ubicado en la parte superior de un edificio y tenía un solo canal que servía para enviar y recibir. Para hablar, el usuario tenía que oprimir un botón que habilitaba al transmisor e inhabilitaba al receptor. Ej: [Radio Taxi](#)

En 1960, surge **IMTS (Sistema Mejorado de Telefonía Móvil)**, que también usaba un transmisor grande de alta frecuencia en una colina, pero tenía 2 frecuencias, una para enviar y otra para recibir. Los usuarios no se escuchaban unos a otros. No fue práctico debido a su capacidad limitada.

En 1982 surge **AMPS (Sistema Avanzado de Telefonía Móvil)**. Dividía una región en **celdas** (de aquí surge el concepto “celular”). Cada celda utiliza un conjunto de frecuencias que no es utilizada por ninguno de sus vecinos. La idea es el uso de celdas pequeñas, y la reutilización de frecuencias en celdas vecinas no adyacentes. Por lo tanto, el diseño celular incrementa la capacidad del sistema a medida de que las celdas se hacen más pequeñas. Además, al ser más pequeñas las celdas, se requiere menor potencia. Las celdas sobrecargadas pueden dividirse en microceldas.

En el centro de cada celda está la estación base a la cual transmiten todos los teléfonos de la celda. La estación base es una PC y un transmisor/receptor conectado a una antena. Las estaciones base se conectan a un mismo (o varios) MTSO o MSC (Oficina/Centro de conmutación móvil). Las MTSO se comunican mediante una red de conmutación de paquetes.

Cuando un teléfono móvil está en una celda específica y sale de su celda, su estación base nota que la señal se desvanece o pregunta a las estaciones vecinas cuánta potencia están recibiendo de ella. A continuación, se transfiere el control a la celda que recibe mayor potencia. Se informa al teléfono cuál es su nueva estación base, y si se está efectuando una llamada, se le pide que cambie de canal. Esto se llama **transferencia de celda**, y tarda aprox. 300 mseg. Hay 2 formas:

- **Transferencia suave:** El teléfono es adquirido por la nueva estación base antes de que la anterior finalice, por lo que no hay pérdida de continuidad. Sin embargo, el teléfono debe poder sintonizar 2 frecuencias a la vez (sólo pueden los teléfonos de 3ª Generación)
- **Transferencia Dura:** La estación antigua deja al teléfono antes de que se adquiera. Si hay una llamada, se corta. Es abrupto.

Teléfonos Móviles de 2ª Generación: Voz Digital

Hay 3 sistemas en uso:

- 1) **DAMPS (Sistema Avanzado de Telefonía Móvil Digital):** Se diseñó para coexistir con AMPS. La señal de voz capturada por el micrófono se digitaliza y comprime en el teléfono, en lugar de la estación base, para reducir el tráfico. Esto permite que hasta 3 usuarios puedan compartir un solo par de frecuencias que utilicen la TDM. En la transferencia de celdas, al contrario que en los teléfonos 1G, interviene el teléfono, en una técnica llamada **MAHO (Transferencia Asistida Móvil de Celda)**. Se usa en EEUU y Japón
- 2) **GSM (Sistema Global para Comunicaciones Móviles):** Se usa en el resto del mundo. Es similar a DAMPS. Los 2 son sistemas celulares. En ambos se usa FDM. En los 2 se usa TDM

para dividir un solo par de frecuencia en ranuras de tiempo compartida por múltiples celulares. Sin embargo, los canales GSM son mucho más anchos que los DAMPS (200 vs 30 KHz) y almacenan relativamente pocos usuarios (3 vs 8), lo que da a GSM una tasa de datos mucho mas grande por usuario.

- 3) CDMA (Acceso Múltiple por División de Código):** Es la base de los sistemas 3G. CDMA permite que cada estación transmita todo el tiempo a través de todo el espectro de frecuencia. Se utiliza la teoría de codificación para separar múltiples transmisiones simultáneas. CDMA no supone que las tramas que colisionan son totalmente distorsionadas, sino que asume que se agregan múltiples señales en forma lineal. La clave de CDMA es tener la capacidad de extraer la señal deseada y rechazar todo lo demás como ruido aleatorio.

Teléfonos Móviles de 3ª Generación: Voz y Datos Digitales

El tráfico de datos ha comenzado a exceder al de voz, que es en esencia plano. Las industrias han adoptado el formato digital, y se pretende que todo tenga conectividad inalámbrica a Internet con un BW alto.

En 1992 la ITU trató de llevar a cabo este sueño, creando la **IMT-2000**. Se le agregó el 2000 por:

- a. Era el año en el que se suponía que debía funcionar
- b. Era la frecuencia que se suponía que trabajaría
- c. Era el ancho de banda que el servicio debería tener

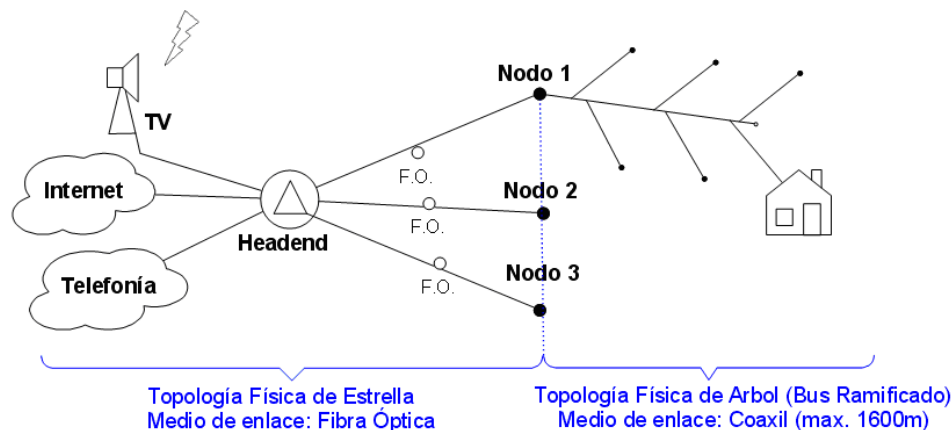
No cumplió con nada de lo anterior. Se realizaron varias propuestas, entre las más destacadas:

- a. W-CDMA (CDMA de banda ancha): Propuesta por Ericsson. Utilizaría espectro disperso de secuencia directa. Se implementó en Europa bajo el nombre UMTS
- b. CDMA2000: Propuesta de Qualcomm

El problema real no fue de ingeniería, sino de políticas (es o no compatible hacia atrás, con qué tecnologías, etc.). Finalizó la disputa cuando Ericsson compro Qualcomm. Sin embargo, la tecnología 3G aún está en desarrollo.

SISTEMA DE TV POR CABLE

Un sistema con fibra para distancias considerables y cable coaxial para las casas se conoce como sistema **HFC (Red Híbrida de Fibra Óptica y Cable Coaxial)**.



Para instalarlo, los amplificadores telefónicos de 1 vía se deben reemplazar por los de 2 vías. Otro punto a eliminar, es que los vecindarios por lo general comparten un mismo cable. Esto se soluciona conectando los cables de cada casa directamente al nodo de fibra, ya que el ancho de banda proporcionado por el amplificador *head end* al nodo de fibra es infinito.

Respecto a la asignación del espectro:

1. Canales Descendentes: Modulados en QAM-64 o 256 si la calidad del cable es muy buena
2. Canales Ascendentes: QPSK.

Respecto a los **modems de cable**, ha surgido un estándar DOCSIS, por la cual los comerciantes no estuvieron muy contentos. Es una interfaz a la PC vía USB, siempre activos. Cuando el MODEM se conecta, explora los canales descendentes en busca de un paquete especial que el amplificador *head end* transmite periódicamente para proporcionar parámetros del sistema a los modems que se acaban de conectar. Al encontrar el paquete, el MODEM anuncia su presencia en uno de los canales ascendentes, y el amplificador le asigna canales. Luego el MODEM se “alinea” con el amplificador, para obtener los parámetros correctos de temporización y las mini ranuras para transmitir (*ver ALOHA ranurado con retroceso exponencial binario*)

UNIDAD 3: La Capa de Enlace

Esta capa tiene que ver con los algoritmos para lograr una comunicación confiable y eficiente entre 2 máquinas adyacentes en la capa de enlace de datos. Con adyacente, queremos decir que las 2 máquinas están conectadas por un canal de comunicaciones que actúa conceptualmente como un alambre, es decir, que los bits se entregan con exactitud en el mismo orden que fueron enviados.

La capa de Enlace tiene que desempeñar varias funciones

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

Para cumplir con esas metas, la capa de enlace de datos toma de la capa de red los paquetes y los encapsula en **tramas** para transmitirlos. El manejo de las tramas es tarea primordial en esta capa.

Servicios Proporcionados a la capa de red

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

La capa de enlace puede diseñarse para proporcionar varios servicios. Los servicios reales pueden variar de sistema a sistema. 3 posibilidades razonables son:


1. Servicio No orientado a la conexión Sin confirmación de recepción: Consisten en hacer que la máquina de origen envíe tramas independientes a la máquina de destino sin pedir que ésta confirme la recepción. No se establece la conexión de antemano ni se libera después. Si se pierde una trama, en la capa de enlace no se hace ningún intento por detectar la pérdida ni por recuperarse de ella. Es apropiada cuando la tasa de errores es muy baja, o para tráfico en tiempo real. La mayoría de las LANs la utilizan.
2. Servicio No orientado a la conexión CON confirmación de recepción: Se confirma de manera individual la recepción de cada trama enviada. Es útil en canales inestables. Vale poner énfasis en que proporcionar confirmaciones de recepción en la capa de enlace es una optimización, nunca un requisito, ya que la capa de red siempre puede enviar un paquete y esperar que se confirme su recepción. Si las tramas se confirman y retransmiten de manera individual, los paquetes completos pasan con mayor rapidez.
3. Servicio orientado a la conexión CON confirmación de recepción: Se establece una conexión antes de transmitir. Cada trama enviada a través de la conexión está numerada, y la capa de enlace de datos garantiza que cada trama enviada llegará a su destino. Además garantiza que cada trama será recibida sólo una vez y en el orden adecuado. Este servicio proporciona a la capa de red el equivalente de un flujo de bits confiable. En un servicio orientado a la conexión hay 3 fases: establecimiento de la conexión, transferencia de datos y liberación de la conexión.

Entramado

Es responsabilidad de la capa de enlace detectar, y de ser necesario, corregir los errores. El método común es que la capa de enlace de datos divida el flujo de bits en tramas separadas y que calcule la suma de verificación de cada trama. Cuando una trama llega a destino, se recalcula su suma de verificación. Si la nueva suma es distinta de la contenida en la trama, la capa de enlace sabe que ha ocurrido un error y toma las medidas para manejarlo. Hay 4 métodos

1. Conteo de Caracteres: Se vale de un campo en el encabezado para especificar el nº de caracteres en la trama. El problema de este algoritmo es que la cuenta puede alterarse por un error de transmisión. Incluso si el destino sabe que está mal la trama porque la suma de verificación es incorrecta, no tiene forma de saber donde comienza la siguiente trama. Regresar una trama a la fuente solicitando la retransmisión no ayuda ya que el destino no sabe cuantos caracteres tiene que saltar para pedir la retransmisión. No se utiliza.

5	1	2	3	4	5	6	7	8	9	8	0	1	2	3	4	5	6	8	7	8	9	0	1	2	3
Trama 1: 5 caract.					Trama 2: 5 caract.					Trama 3: 8 caracteres								Trama 4: 8 caracteres							

 Cuenta de Caracteres

2. Banderas con relleno de caracteres: Este método evita el problema de tener que sincronizar nuevamente después de un error, haciendo que cada trama inicie y termine con bits especiales, llamados **banderas**. Si el receptor pierde la sincronía, simplemente puede buscar la bandera para encontrar el final e inicio de la trama actual. Dos banderas consecutivas señalan el final de una trama y el comienzo de la siguiente. Se puede dar el caso de que el patrón de bits de la bandera aparezca en los datos. Esto se soluciona insertando un byte ESC antes de la bandera “accidental” en los datos. La capa receptora lo detecta, lo elimina, y lo toma como un carácter de datos. Una desventaja de esta técnica, es que está fuertemente atada a caracteres de 8 bits.

FLAG	Encabezado	DATOS	Terminador	FLAG
------	------------	-------	------------	------

3. Banderas de inicio y fin, con relleno de bits: Permite que las tramas de datos contengan un nº arbitrario de bits y admite códigos de caracteres con un nº arbitrario de bits por carácter. Cada trama comienza y termina con un patrón de bits especial, 0111110 (es una bandera). Cada vez que la capa de enlace del emisor encuentra 5 unos consecutivos en los datos, automáticamente inserta un bit 0 en el flujo de datos saliente (relleno de bits). Esto es análogo al carácter ESC. Del lado del receptor, se hace lo opuesto. Con el relleno de bits, si el receptor se pierde, busca el último patrón y solicita retransmisión.
4. Violaciones de codificación en la capa física: Sólo se aplica en redes en las que la codificación en el medio físico contiene cierta redundancia, como con la codificación Manchester y Manchester Diferencial.

Control de Errores

La manera normal de asegurar la entrega confiable de datos es proporcionar retroalimentación al emisor sobre lo que está ocurriendo en el otro lado de la línea. El protocolo exige que el receptor regrese **tramas de control** especiales que contengan confirmaciones de recepción positivas (**ACK**) o negativas (**NAK**) de las tramas que llegan.

Una complicación adicional surge de la posibilidad de que los problemas de hardware causen la desaparición de una trama completa. El emisor se quedaría esperando eternamente si se pierde por completo una trama. Esto puede solucionarse mediante el agregado de **temporizadores** en la capa de enlace de datos.

Existe el peligro de que el receptor acepte la misma trama 2 o más veces y que la pase a la capa de red más de una vez. Para evitar esto, generalmente se asignan **números de secuencia** a las tramas.

Control de Flujo

La idea es qué hacer con un emisor que quiere transmitir tramas de manera sistemática y a mayor velocidad que aquella con que puede aceptarlas el receptor. Esto satura por completo al receptor, no será capaz de manejar las tramas conforme lleguen, y comenzará a perder algunas.

Para solucionarlo, por lo general se utilizan 2 métodos:

1. Control de flujo basado en retroalimentación: el receptor regresa información al emisor autorizándolo para enviar más datos o indicándole su estado
2. Control de flujo basado en tasa: el protocolo tiene un mecanismo integrado que limita la tasa a la que el emisor puede transmitir los datos, sin recurrir a retroalimentación.

DETECCIÓN Y CORRECCIÓN DE ERRORES

Los errores de transmisión van a ser inevitables durante muchos años más. Tendremos que lidiar con ellos. Los errores tienden a aparecer en ráfagas. Respecto a los errores aislados, tienen como ventaja de que los datos de computadora siempre se envían en bloques de bits. Si los errores fueran independientes, todos los bloques tendrían errores. La desventaja es que son más difíciles de detectar y corregir que los errores aislados.

Códigos de Corrección de Errores

Hay 2 estrategias básicas:

1. Códigos de corrección de errores: Se incluye suficiente información redundante en cada bloque de datos transmitido para que el receptor pueda deducir lo que debió ser el carácter transmitido. Ej: [Código de Hamming](#)
2. Códigos de detección de errores: Se incluye información suficiente para detectar el error y así poder pedir la retransmisión. Ej: [CRC](#)

En los canales rápidos y confiables, como los de fibra, es mejor utilizar códigos de detección. En los canales con muchos errores, como los inalámbricos, convienen los de corrección.

Una trama consiste en m bits de datos, y r de redundancia. $N=m+r$ es la **palabra codificada**.

PROTOCOLOS ELEMENTALES DE ENLACE DE DATOS

Para este modelo, supondremos:

- Que en las capas físicas, de enlace y de red hay procesos independientes que se comunican pasando mensajes de un lado a otro.
- Que la máquina A desea mandar un flujo de datos considerable a la máquina B, usando un servicio confiable orientado a la conexión
- A tiene un suministro infinito de datos listos para ser enviados, y no debe esperar por ellos
- Las máquinas no fallan. Los protocolos manejan errores de comunicación, no de hardware.
- Existen procedimientos para pasar de capa a capa: Ej: `A_Capa_1()`; `De_Capa_1()`;
- El hardware emisor calcula y agrega la suma de verificación (y así crea el terminador) por lo que el software de la capa de enlace no necesita preocuparse por ello
- Inicialmente el receptor está esperando un evento
- Si llega una trama errónea, NUNCA la pasa a la capa de red.

*“Es importante entender la relación entre un **paquete** y una **trama**. La capa de red construye un paquete tomando un mensaje de la capa de transporte y agregándole el encabezado de la capa de red. Este paquete se pasa a la capa de enlace de datos para incluirlo en el campo *info* de una trama saliente. Cuando ésta llega a su destino, la capa de enlace de datos extrae de ella el paquete y a continuación lo pasa a la capa de red. De esta manera, esta capa puede actuar como si las máquinas pudieran intercambiar paquetes directamente”*

1) Protocolo Simplex sin restricciones

Los datos se transmiten en una sola dirección. Las capas de red siempre están listas. El tiempo de procesamiento siempre puede ignorarse. Hay espacio infinito en buffer. El canal de comunicación nunca pierde tramas. No se usan nº de secuencia ni ACK/NAK. El único evento posible es la llegada de una nueva trama.

1. <code>void emisor() {</code>	11. <code>void receptor() {</code>
2. <code>frame s;</code>	12. <code>frame r;</code>
3. <code>packet buffer;</code>	13. <code>tipo_evento evento;</code>
4. <code>while (true) {</code>	14. <code>while (true) {</code>
5. <code>de_capa_red (&buffer);</code>	15. <code>esperar_evento (&evento);</code>
6. <code>s.info = buffer;</code>	16. <code>de_capa_fisica (&r);</code>
7. <code>a_capa_fisica (&s);</code>	17. <code>a_capa_red (&r.info);</code>
8. <code>}</code>	18. <code>}</code>
9. <code>}</code>	19. <code>}</code>

2) Protocolo Simplex de Parada y espera

Omitiremos ahora el hecho de la capacidad de la capa de red receptora para procesar datos de entrada con una rapidez infinita. Aún se supone un canal de comunicaciones libre de errores y tráfico de datos simplex. Debemos resolver cómo evitar que el emisor sature al receptor enviando datos a mayor velocidad de la que este último puede procesarlos. Lo haremos mediante

retroalimentación. Tras haber pasado un paquete a su capa de red, el receptor regresa al emisor una trama ficticia que autoriza al emisor a transmitir la siguiente.

Los protocolos en los que el emisor envía una trama y luego espera una confirmación de recepción antes de continuar se denominan de **parada y espera**.

```
1. void emisor2() {
2.     frame s;
3.     packet buffer;
4.     tipo_evento evento;
5.     while (true) {
6.         de_capa_red (&buffer);
7.         s.info = buffer;
8.         a_capa_fisica (&s);
9.         esperar_evento (&evento);
10.    }
11. }

12. void receptor2() {
13.     frame r, s;
14.     tipo_evento evento;
15.     while (true) {
16.         esperar_evento (&evento);
17.         de_capa_fisica (&r);
18.         a_capa_red (&r.info);
19.         a_capa_fisica (&s);
20.     }
21. }
```

3) Protocolo Simplex para un canal con ruido

Si una trama se daña en camino, el hardware receptor detectará esto cuando calcule la suma de verificación. Si la trama está dañada de manera que la suma de verificación esté correcta, un caso improbable, este protocolo puede fallar y entregar el paquete a la capa de red.

Se podría agregar un temporizador. El emisor podría enviar una trama, pero el receptor sólo enviaría una trama de confirmación de recepción si los datos llegaran correctamente. Si llegara una trama dañada al receptor, el temporizador del emisor expiraría y se enviaría una trama de nuevo. Esto se repetiría hasta que la trama llegue intacta. Pero esto tiene un defecto mortal. Si la trama se demora y expira el temporizador del emisor, se reenviaría, y podrían llegar 2 tramas iguales a B, que las tomaría como correctas.

La forma de solucionar lo anterior es agregando un nº de secuencia en cada trama. La cantidad mínima de bits para un nº de secuencia es 1 (0 o 1) para este caso. En cada instante, el receptor espera un nº de secuencia en particular. Cualquier trama que llegue con otro nº de secuencia, se descarta como duplicado. Cuando llega la trama esperada, se pasa a la capa de red, y el nº de secuencia esperado se incrementa en módulo 2.

```
1. # define MAXSEQ = 1
2. typedef enum{llego_trama, error_checksum, expira} event_type;
3.
4.
5. void receptor3() {
6.     frame r, s;
7.     tipo_evento evento;
8.     seq_num trama_esperada = 0;
9.
10.    while (true) {
11.        esperar_evento (&evento);
12.        if (evento == llego_trama) {
13.            de_capa_fisica (&r);
14.            if (r.seq == trama_esperada) {
15.                a_capa_red (&r.info);
16.                inc (trama_esperada);
17.            }
18.            s.ACK = 1- trama_esperada;
19.            a_capa_fisica (&s);
20.        }
21.    }
22. }

23. void emisor3() {
24.     frame s;
25.     packet buffer;
26.     tipo_evento evento;
27.     seq_num prox_trama_a_enviar = 0;
28.
29.     de_capa_red (&buffer);
30.     while (true) {
31.         s.info = buffer;
32.         s.seq = prox_trama_a_enviar;
33.         a_capa_fisica (&s);
34.         start_timer (s.seq);
35.         esperar_evento (&evento);
36.         if (evento == llego_trama) {
37.             de_capa_fisica (&s);
38.             if (s.ACK == prox_trama_a_enviar) {
39.                 stop_timer (s.ACK);
40.                 a_capa_red (&buffer);
41.                 inc (prox_trama_a_enviar);
42.             }
43.         }
44.     }
45. }
```

Los protocolos en los que el emisor espera una confirmación de recepción positiva antes de avanzar al siguiente elemento de datos se llaman **PAR (Confirmación de Recepción Positiva con Retransmisión)** o **ARQ (Solicitud Automática de Repetición)**.

PROTOCOLOS DE VENTANA CORREDIZA

Una manera de lograr una transmisión de datos duplex total es tener 2 canales de comunicaciones separados y utilizar cada uno para tráfico de datos simplex. En ambos casos, el ancho de banda de canal usado para confirmaciones de recepción se desperdicia casi por completo. En efecto, el usuario está pagando por 2 circuitos, pero solo se usa la capacidad de uno.

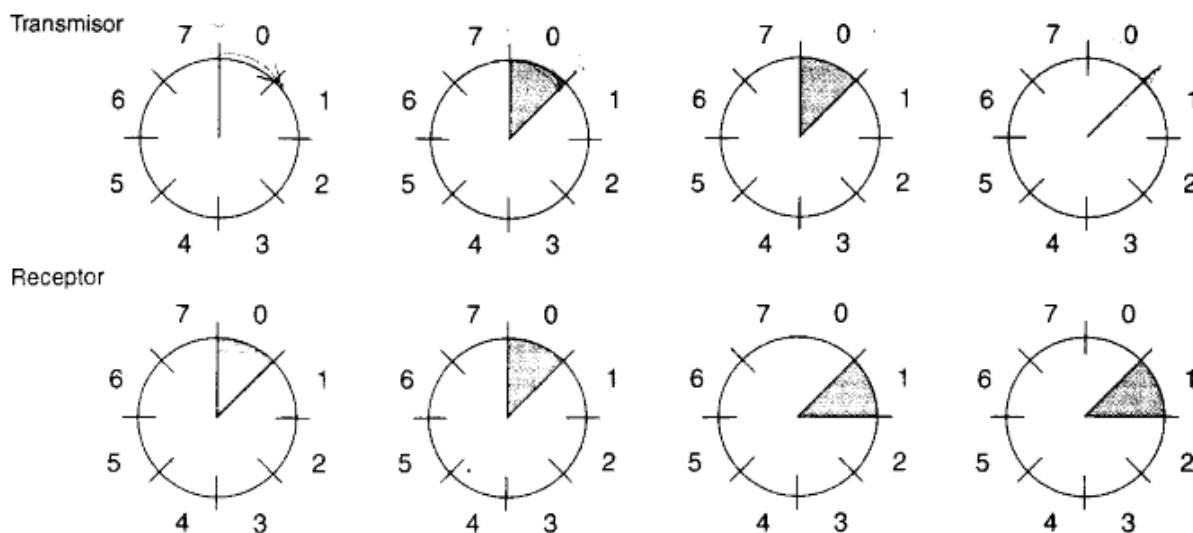
Una mejor idea es usar el mismo circuito para datos en ambas direcciones. Es este modelo, las tramas de datos de A a B se mezclan con las tramas de confirmación de recepción de A a B.

Cuando llega una trama de datos, en lugar de enviar automáticamente una trama de control independiente, el receptor se aguanta y espera hasta que la capa de red le pase el siguiente paquete. El ACK/NAK se anexa a la trama de datos de salida (usando el campo *ack* del encabezado de la trama). En efecto, la confirmación viaja gratuitamente en la siguiente trama de datos de salida. Esta técnica se conoce como **superposición**.

¿Cuánto tiempo debe esperar la capa de enlace de datos un paquete al cual superponer la confirmación de recepción? Si la capa de enlace de datos espera más tiempo del que tarda en terminar el temporizador del emisor, la trama será retransmitida, frustrando el propósito de enviar confirmaciones de recepción. Si ve que va a expirar, la envía en una trama vacía.

Los siguientes protocolos son bidireccionales y corresponden a una clase llamada **ventana corrediza**. En ellos, cada trama de salida contiene un nº de secuencia que va de 0 hasta $2^n - 1$, siendo n el tamaño en bits de la ventana.

En cualquier instante, el emisor mantiene un grupo de nº de secuencia que corresponde a las tramas que tiene permitido enviar (**ventana emisora**). De manera semejante, el receptor mantiene una **ventana receptora**, que son las que puede aceptar. En algunos protocolos el tamaño de las ventanas es estático, y en otros dinámicos.



Los nº de secuencia en la ventana del emisor representan tramas enviadas, o que pueden ser enviadas, pero cuya recepción aún no ha sido confirmada. Cuando llega un paquete nuevo de la capa de red, se le da el siguiente nº de secuencia mayor, y el extremo superior de la ventana avanza en 1. De tal manera, la ventana mantiene una lista de tramas sin confirmación de recepción.

Si el tamaño de la ventana es n , el emisor necesita n búferes para contener las tramas sin confirmación. Si la ventana llega a crecer hasta su tamaño máximo, la capa de enlace de datos emisora deberá hacer que la capa de red se detenga hasta que se libere otro búfer.

En la capa de enlace de datos receptora, cuando se recibe la trama cuyo nº sea igual al extremo inferior de la ventana, se pasa a la capa de red, genera una confirmación y avanza la ventana en 1. La ventana del receptor siempre conserva su tamaño inicial. Un tamaño de ventana de 1 significa que la capa de enlace de datos sólo acepta tramas en orden, pero con ventanas más grandes esto no es así. La capa de red, en contraste, siempre recibe los datos en el orden correcto, independientemente del tamaño de la ventana.

Retroceso N y Repetición Selectiva

Con una ventana de tamaño 1, el emisor necesita una confirmación de recepción antes de enviar otra trama. Se relajamos esta restricción, podemos lograr mayor eficiencia. Básicamente la solución está en permitir que el emisor envíe hasta w tramas antes de bloquearse, en vez de sólo 1. Con una selección adecuada de w , el emisor podrá transmitir tramas continuamente durante un tiempo igual al tiempo de tránsito de ida y vuelta sin llenar la ventana.

El producto del ancho de banda por el retardo del viaje ida y vuelta, indica básicamente cuál es la capacidad del canal, y el emisor necesita la capacidad de llenarlo sin detenerse para poder funcionar con eficiencia máxima.

Esta técnica se conoce como **canalización**. Si la capacidad del canal es b bits/seg, el tamaño de la trama de L bits, y el tiempo de propagación de ida y vuelta es de R segundos, la utilización de la línea será de:

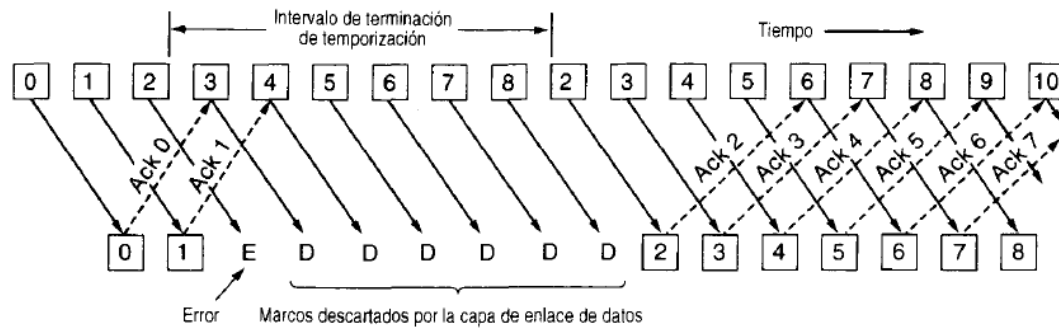
$$\text{Utilización de la línea} = \frac{L}{L + bR}$$

El envío de tramas por canalización por un canal de comunicación inestable presenta problemas serios. Por ejemplo, si una trama en la mitad de una serie larga se pierde o daña, llegarán grandes cantidades de tramas sucesivas al receptor antes de que el emisor se entere de que algo anda mal.

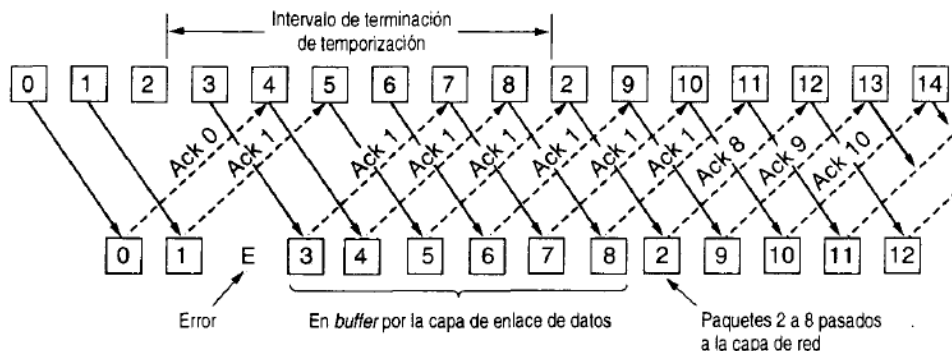
Hay 2 métodos básicos para manejar los errores durante la canalización:

1. **Retroceso N:** El receptor descarta todas las tramas subsecuentes, sin enviar confirmaciones de recepción para las tramas descartadas. Esta estrategia corresponde a una ventana de recepción de tamaño 1. Si la ventana del emisor se llena antes de terminar el temporizador, el canal comenzará a vaciarse. En algún momento, el emisor terminará de esperar y retransmitirá en orden todas las tramas cuya recepción aún no se ha confirmado,

comenzando por la dañada o perdida. Esta estrategia puede desperdiciar gran ancho de banda si la tasa de errores es alta. ([Ver protocolo de Ejemplo en el libro, página 220](#))



- Repetición Selectiva:** Cuando se utiliza, se descarta una trama dañada recibida, pero las tramas en buen estado recibidas después de esa se almacenan en búfer. Cuando el emisor termina, solo la última trama sin confirmación se retransmite. La repetición selectiva con frecuencia se combina con el hecho de que el receptor envíe una confirmación de recepción negativa (NAK) cuando detecta un error. Las NAK estimulan la retransmisión antes de que el temporizador correspondiente expire, por lo que mejoran el rendimiento. La repetición selectiva corresponde a una ventana del receptor mayor que 1. Las terminaciones de temporización pendientes forman una lista enlazada, en la que cada nodo de la lista indica la cantidad de pulsos de reloj que faltan para que expire el temporizador, el nº de trama temporizada y el puntero al siguiente nodo. Cada vez que se pulsa el reloj de hardware, se actualiza el tiempo real y el contador de pulsos a la cabeza de la lista se decrementa. ([Ver protocolo de Ejemplo en el libro, página 224](#))



Problema con los nº de Secuencia en Ventanas de tamaño > 1

Una vez que el receptor ha avanzado su ventana, el nuevo intervalo de nº de secuencia válidos se traslapa con el anterior. En consecuencia, el siguiente grupo de tramas podría ser de tramas duplicadas (si se perdieron los ACK) o nuevas (si se recibieron los ACK). El receptor no tiene forma de distinguir estos 2 casos.

La salida de esto, es asegurarse que, una vez que el emisor haya avanzado su ventana, no haya traslape con la ventana original, por lo que el tamaño máximo de la ventana debe ser al menos la mitad del intervalo de nº de secuencia. Ej: Si tengo 8 nº secuencia, la ventana puede ser hasta 4. El receptor deberá tener tantos búferes y temporizadores como sea el tamaño de la ventana.

VERIFICACIÓN DE LOS PROTOCOLOS

Máquinas de Estado Finito

Cada **máquina de protocolo** (emisor o receptor) siempre está en un estado específico en cualquier instante. Por lo general los **estados** se escogen para que sean aquellos instantes en que la máquina de protocolo está esperando que ocurra el siguiente evento (Ej: **wait (&estado)**). El estado del sistema completo es la combinación de todos los estados de las 2 máquinas de protocolos y del canal. El estado del canal está determinado por su contenido. Una trama permanece “en el canal” hasta que la máquina de protocolo ejecuta *DeCapaFísica()* y la procesa.

De cada estado hay 0 o más **transiciones** posibles a otros estados, que ocurrirán cuando suceda un evento. Dada una descripción completa de las máquinas de protocolo y las características del canal, es posible dibujar un grafo dirigido que muestre todos los estados como nodos, y las transiciones como arcos dirigidos.

Un estado en particular se designa como **estado inicial**, desde el cual pueden alcanzarse algunos, o quizás todos los demás estados mediante una secuencia de transiciones. Es posible determinar los estados alcanzables y los que no, mediante una técnica llamada **análisis de asequibilidad**. Este análisis también permite detectar errores o bloqueos que no se tuvieron en cuenta en el diseño.

Formalmente, un modelo de máquina de estados finitos de un protocolo se puede considerar como un cuádruple (S, M, I, T), donde:

- S es el conjunto de estados que pueden estar en los procesos y el canal
- M es el conjunto de tramas que pueden intercambiarse a través del canal
- I es el conjunto de estados iniciales de los procesos
- T es el conjunto de transiciones entre los estados.

Modelos de Red de Petri

Una red de Petri tiene 3 elementos básicos. **Lugar** es un estado en el que puede estar parte del sistema. Aparecen como círculos. Las **Transiciones** se indican con una barra horizontal. Los **arcos** salen de las transiciones. Cada transición tiene 0 o más **arcos de entrada**, que llegan de sus lugares de entrada, y 0 o más **arcos de salida** que van a sus lugares de salida. Los **tokens** se representan con un punto grueso.

Se **habilita una transición** si hay cuando al menos un token de entrada en cada uno de sus lugares de entrada. Cualquier transición habilitada puede **dispararse** a voluntad, quitando un token de cada lugar de entrada y depositando un token en cada lugar de salida. Si el nº de arcos de entrada es distinto al nº de arcos de salida, no se conservan los tokens. La decisión de disparo de una transición es indeterminada, por lo que las redes Petri son útiles para modelar protocolos.

Las redes Petri pueden expresarse matemáticamente.

EJEMPLOS DE PROTOCOLOS DE ENLACE DE DATOS

HDLC (Control de Enlace de Datos de Alto Nivel)

Los protocolos siguientes fueron la evolución de los protocolos de enlace de datos usados en los mainframes de IBM: SDLC ► ADCPP ► HDLC ► LAP ► LAPB

Todos son orientados a bits y usan el relleno de bits para lograr la transparencia de los datos.

Bits	8	8	8	≥ 0	16	8
	01111110	Dirección	Control	Datos	Checksum	01111110

El campo **control** se usa para nº de secuencia, ACK/NAK y otros propósitos. El campo **datos** es de longitud arbitraria, aunque la eficiencia de la suma de verificación disminuye conforme el tamaño de la trama aumenta, debido a la mayor probabilidad de múltiples errores en ráfaga.

El protocolo utiliza una ventana corrediza, con un nº de secuencia de 3 bits, por lo que en cualquier momento pueden estar pendientes hasta 7 tramas sin confirmación de recepción.

Hay 3 tipos de tramas: de **información**, **supervisión** y **no numeradas**.

Campo de Control de cada tipo de trama					
Bits	1	1	1	1	3
Trama de Información	0	Secuencia		S/F	Siguiente
Trama de Supervisión	1	0	Tipo		S/F Siguiente
Trama no numerada	1	1	Tipo		S/F Siguiente

El campo **secuencia** es el nº de secuencia. El campo **siguiente** es una confirmación de recepción superpuesta: lleva el nº de la próxima trama a recibir. El bit **S/F (Sondeo Final)** se utiliza cuando una computadora está sondeando un grupo de terminales. Cuando se usa como P, la PC está invitando a la terminal a enviar datos. Todas las tramas enviadas por la terminal, excepto la última, tienen el bit en P, salvo el último que se establece en F.

Los distintos tipos de tramas de supervisión se distinguen por el campo **tipo**: 0 es un ACK; 1 es un NAK (el campo siguiente aquí indica cual trama es la errónea); el 2 indica receptor no listo, para que el emisor detenga el envío; el 3 solicita la retransmisión de una trama determinada.

Protocolo PPP (Punto a Punto): usado en Internet

PPP proporciona 3 características:

1. Un método de entramado que delimita sin ambigüedades el final de una trama y el inicio de la siguiente. El formato de la trama maneja también la detección de errores
2. Un protocolo de control de enlace para activar líneas, probarlas, negociar opciones y desactivarlas ordenadamente cuando ya no son necesarias (**LCP – Protocolo de Control de Enlace**). Admite circuitos síncronos y asíncronos, y codificaciones orientadas a bits y caracteres.
3. Un mecanismo para negociar opciones de capa de red, con independencia del protocolo de capa de red. Consiste en tener un **NCP (Protocolo de Control de Red)** para cada protocolo de Red soportado.

Para ver la manera en que encajan estas piezas, consideremos la situación típica de un usuario casero llamando al proveedor de servicios de Internet para convertir una PC casera en un *host* temporal de Internet. La PC llama inicialmente al enrutador del proveedor a través de un módem. Una vez que el módem del enrutador ha contestado el teléfono y ha establecido una conexión física, la PC manda al enrutador una serie de paquetes LCP en el campo de carga útil de uno o más marcos PPP. Estos paquetes, y sus respuestas, seleccionan los parámetros PPP por usar.

Una vez que se han acordado estos parámetros, se envía una serie de paquetes NCP para configurar la capa de red. Típicamente, la PC quiere ejecutar una pila de protocolos *TCP/IP*, por lo que necesita una dirección de IP. No hay suficientes direcciones de IP para todos, por lo que normalmente cada proveedor de Internet tiene un bloque de ellas y asigna dinámicamente una a cada PC que se acaba de conectar para que la use durante su sesión. Si un proveedor posee n direcciones de IP, puede tener hasta n máquinas conectadas simultáneamente, pero su base total de clientes puede ser muchas veces mayor. Se utiliza el NCP para IP para asignar la dirección de IP.

En este momento, la PC es ya un *host* de Internet y puede enviar y recibir paquetes de IP, igual que los *host* permanentes. Cuando el usuario ha terminado, se usa NCP para dismantelar la conexión de la capa de red y liberar la dirección de IP. Luego se usa LCP para cancelar la conexión de la capa de enlace de datos. Finalmente, la computadora indica al módem que cuelgue el teléfono, liberando la conexión de la capa física.

El formato de marco de PPP se escogió de modo que fuera muy parecido al formato de marco de HDLC. La diferencia principal entre PPP y HDLC es que el primero está orientado a caracteres, no a bits. En particular, PPP, usa el relleno de bytes.

Formato de una trama PPP

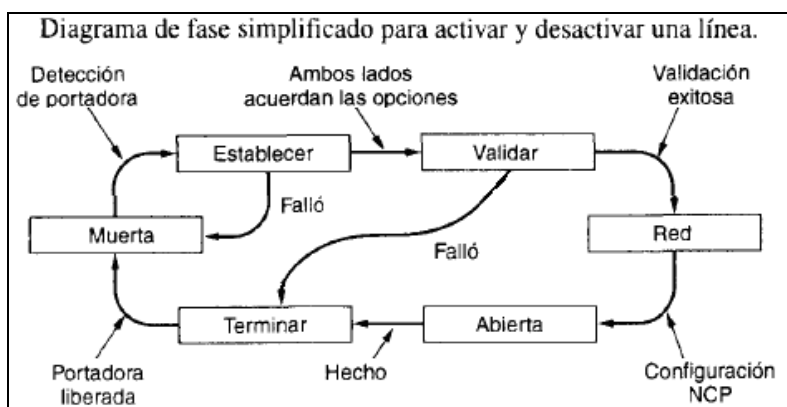
Bytes	1	1	1	1 o 2	Variable	2 o 4	1
	Bandera	Dirección	Control	Protocolo	Carga útil	Checksum	Bandera
	01111110	11111111	00000011				01111110

El campo **dirección** se establece en 11111111 para indicar que todas las estaciones deben aceptar la trama. Esto evita tener que asignar direcciones en la capa de enlace de datos.

Dado que los campos de **dirección** y de **control** son constantes en la configuración predeterminada, LCP proporciona los mecanismos necesarios para que las dos partes negocien una opción que simplemente los omita por completo y ahorre dos bytes por marco.

El cuarto campo PPP es el de **protocolo**. Su tarea es indicar la clase de paquete que está en el campo de **carga**. Se definen códigos para LCP, NCP, IP, IPX, AppleTalk y otros protocolos

El tamaño por omisión del campo de **protocolo** es de 2 bytes, pero puede negociarse su disminución a 1 byte usando LCP. **Checksum**, que es de 2 bytes, puede negociarse en 4.



UNIDAD 4: La Subcapa MAC

Este capítulo trata las redes de difusión y sus protocolos. En cualquier red de difusión, el asunto clave es la manera de determinar quién puede utilizar el canal cuando hay competencia por él.

Cuando únicamente hay un canal, determinar quién debería tener el turno es muy complicado. En la literatura, los canales de difusión a veces se denominan **canales multiacceso** o **canales de acceso aleatorio**.

Los protocolos usados para determinar quién sigue en un canal multiacceso, pertenecen a una subcapa de enlace de datos llamada **subcapa MAC (Control de Acceso al Medio)**. La subcapa MAC tiene especial importancia en las LANs, casi todas las cuales usan un canal multiacceso como base de su comunicación. Las WANs, en contraste, usan enlaces punto a punto, excepto las redes satelitales.

EL PROBLEMA DE LA ASIGNACIÓN DEL CANAL

El problema es la forma de asignar un solo canal entre varios usuarios competidores.

1. **Asignación estática de canal en LANs y MANs:** Si hay N usuarios, el ancho de banda se divide en N partes de igual tamaño, y a cada usuario se le asigna una parte (FDM). Dado que cada usuario tiene una banda de frecuencia privada, no hay interferencia entre los usuarios. Si el espectro se divide en N regiones, y hay menos de N usuarios interesados en comunicarse actualmente, se desperdiciará una buena parte de espectro valioso. Si hay más de N usuarios interesados, a alguno de ellos se les negará el permiso por falta de ancho de banda, aún cuando algunos de los usuarios que tengan asignada una banda de frecuencia apenas transmitan o reciban algo. Con TDM sucede lo mismo.
2. **Asignación dinámica de canales en LANs y MANs:** Tiene 5 supuestos clave:
 - a. Modelo de Estación: Consiste en N estaciones independientes, cada una con un programa o usuario que genera tramas para transmisión. La probabilidad de que una trama se genere en un intervalo Δt , es de $\lambda \Delta t$, donde λ es una constante (la tasa de llegada de tramas nuevas). Una vez que se ha generado una trama, la estación se bloquea y no hace nada sino hasta que la trama se ha transmitido con éxito.
 - b. Supuesto de Canal único: Hay un solo canal disponible para todas las comunicaciones. Todas las estaciones pueden transmitir en él y recibir de él. En lo referente al hardware, todas las estaciones son equivalentes, aunque el software del protocolo puede asignarle prioridades
 - c. Supuesto de Colisión: Si dos tramas se transmiten en forma simultánea, se traslapan en el tiempo y la señal resultante se altera. Este evento se llama **colisión**. Todas las estaciones pueden detectar colisiones. Una trama en colisión debe transmitirse nuevamente después. No hay otros errores excepto aquellos generados por las colisiones.
 - d. Tiempo:
 - i. Tiempo continuo: La transmisión de una trama puede comenzar en cualquier momento. No hay reloj maestro que divida el tiempo en intervalos discretos.
 - ii. Tiempo ranurado: El tiempo se divide en intervalos discretos (ranuras). La transmisión de los marcos siempre comienza al inicio de una ranura. Una ranura puede contener 0, 1 o más marcos, correspondientes a una ranura inactiva, una transmisión con éxito, o una colisión, respectivamente.

e. Detección de Portadora

- i. Con Detección de portadora: Las estaciones pueden saber si el canal está en uso antes de intentar usarlo. Si se detecta que el canal está en uso, ninguna estación intentará usarlo hasta que regrese a la inactividad.
- ii. Sin detección de portadora: Las estaciones no pueden detectar el canal antes de intentar usarlo. Simplemente transmiten. Sólo después pueden determinar si la transmisión tuvo éxito.

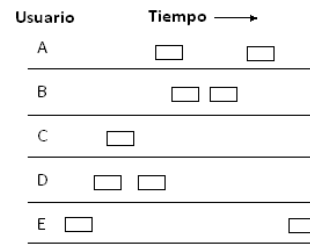
PROTOCOLOS DE ACCESO MÚLTIPLE

ALOHA

En 1970, en Hawaii, se diseñó un sistema llamado ALOHA, que usó la radiodifusión basada en tierra, para determinar el acceso a un canal compartido. Analizaremos 2 versiones de ALOHA: puro y ranurado. Difieren en si se divide o no el tiempo en ranuras discretas en las que deben caer todas las tramas. El ALOHA puro no requiere sincronización global del tiempo, el ranurado sí.

ALOHA PURO

La idea básica es permitir que los usuarios transmitan cuando tengan datos por enviar. Un emisor siempre puede saber si la trama fue o no destruida, escuchando el canal, de la misma manera que los demás usuarios. Si por alguna razón no es posible escuchar cuando se transmite, se necesitan usar ACK. Si la trama fue destruida, el emisor espera un tiempo aleatorio y la reenvía. El tiempo de espera debe ser aleatorio, o las mismas tramas chocarán una y otra vez, en sincronía.



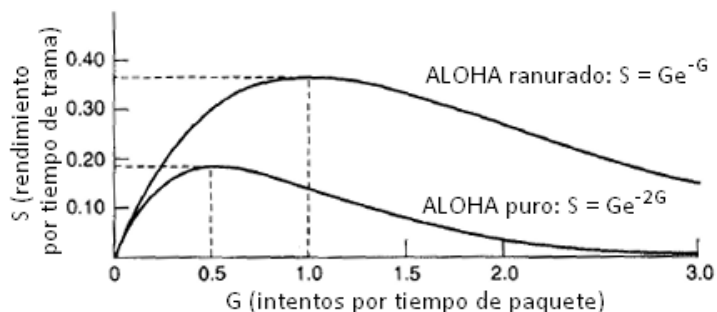
Los sistemas en los cuáles varios usuarios comparten un canal común, de modo tal que puede dar pie a conflictos, se conocen como **sistemas de contención**.

La velocidad real de transporte (*throughput*) de los sistemas ALOHA se maximiza al tener tramas con un tamaño uniforme en lugar de tramas de longitud variable.

Cada vez que 2 tramas traten de ocupar el canal al mismo tiempo, habrá una colisión y ambas se dañarán. Si el primer bit de una trama nueva se traslapa con el último bit de una trama casi terminada, ambas tramas se destruirán por completo, y ambas tendrán que retransmitirse. La suma de verificación no puede (y no debe) distinguir entre una pérdida total y un error ligero.

ALOHA RANURADO

En 1972, Roberts publicó un método para duplicar la capacidad de un sistema ALOHA. Su propuesta fue dividir el tiempo en intervalos discretos, cada uno de los cuales correspondía a una trama. Este enfoque requiere que los usuarios acuerden límites de ranura.



No se permite que una computadora envíe cada vez que se pulsa "enviar". En cambio, se obliga a esperar el comienzo de la siguiente ranura. Dado que el periodo vulnerable ahora es de la mitad, la probabilidad de que no haya más tráfico durante la misma ranura que nuestra trama de prueba es de e^{-G} , lo que conduce a $S=Ge^{-G}$. Como resultado de la dependencia exponencial de e respecto a G , pequeños aumentos en la carga del canal pueden reducir drásticamente su desempeño.

Protocolos de Acceso Múltiple con detección de portadora

Con el ALOHA ranurado, el mejor aprovechamiento de canal que puede lograrse es $1/e$. A continuación analizaremos algunos protocolos que mejoran el desempeño, en los cuales las estaciones escuchan una portadora (una transmisión) y actúan de acuerdo con ello.

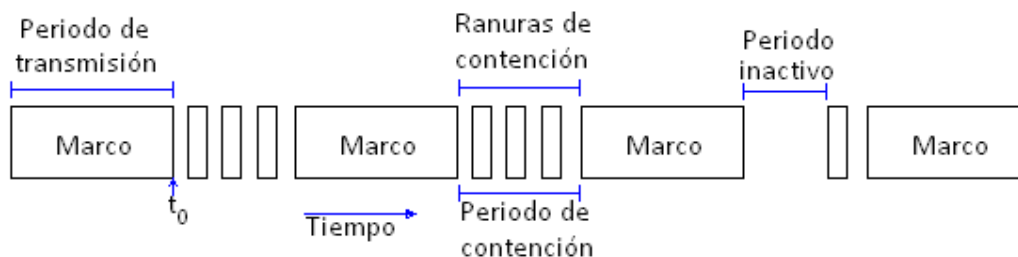
CSMA (Acceso Múltiple con detección de portadora) persistente y no Persistente

1. CSMA persistente-1: Cuando una estación tiene datos por transmitir, primero escucha el canal para ver si otra está transmitiendo en ese momento. Si el canal está ocupado, la estación espera hasta que se desocupa. Cuando la estación detecta un canal en reposo, transmite una trama. Si ocurre una colisión, la estación espera una cantidad aleatoria de tiempo y comienza de nuevo. El protocolo se llama persistente-1 porque la estación transmite con una probabilidad de 1 cuando encuentra el canal inactivo. El retardo de propagación tiene un efecto importante en el desempeño del protocolo. Hay una pequeña posibilidad de que, justo después de que una estación comienza a transmitir, otra estación esté lista para enviar y detectar el canal. Si la señal de la primera estación no ha llegado aún a la segunda, esta última detectará un canal inactivo y comenzará a enviar también, resultando una colisión. Cuanto mayor sea el tiempo de propagación, más importante será este efecto, y peor el desempeño del protocolo. Aun si el retardo de propagación es de cero, habrá colisiones.
2. CSMA no persistente: Antes de enviar, una estación detecta el canal. Si nadie más está transmitiendo, la estación comienza a hacerlo. Sin embargo, si el canal ya está en uso, la estación no observa continuamente el canal a fin de tomarlo de inmediato al detectar el final de la transmisión previa. En cambio, espera un periodo de tiempo aleatorio y repite el algoritmo. Intuitivamente este algoritmo deberá conducir a una utilización mejor del canal y a mayores retardos que el CSMA persistente-1.
3. CSMA persistente-p: Se aplica a canales ranurados. Cuando una estación está lista para enviar, escucha el canal. Si el canal está en reposo, la estación transmite con una probabilidad p . Con una probabilidad $q = 1 - p$, se espera hasta la siguiente ranura. Si esa ranura también está inactiva, la estación transmite o espera nuevamente, con probabilidades p y q . Este proceso se repite hasta que el marco ha sido transmitido o hasta que otra estación ha comenzado a transmitir. En el segundo caso, la estación actúa como si hubiera habido una colisión (espera un tiempo aleatorio y comienza de nuevo). Si la estación detecta inicialmente que el canal está ocupado, espera hasta la siguiente ranura y aplica el algoritmo anterior.

CSMA/CD (Acceso Múltiple con Detección de Portadora y Detección de Colisiones)

Otra mejora es que las estaciones aborten sus transmisiones tan pronto como detecten una colisión. En lugar de terminar de transmitir sus tramas, que de todos modos están irremediabilmente alteradas, deben detener de manera abrupta la transmisión tan pronto como detecten la colisión. Esto ahorra tiempo y ancho de banda. Este protocolo, CSMA/CD, es la base de la popular LAN Ethernet.

Las colisiones pueden detectarse observando la potencia o el ancho de pulso de la señal recibida y comparándola con la señal transmitida. Una vez que una estación detecta una colisión, aborta la transmisión, espera un periodo de tiempo aleatorio e intenta de nuevo, suponiendo que ninguna otra estación ha comenzado a transmitir durante ese lapso. Por tanto, nuestro modelo de CSMA/CD consistirá en periodos alternantes de contención y transmisión, ocurriendo periodos de inactividad cuando todas las estaciones están en reposo. El tiempo mínimo para detectar la colisión, es sólo el tiempo que tarda la señal en propagarse de una estación a otra.



Sea t el tiempo que tarda una señal en propagarse entre las 2 estaciones más lejanas. En el peor caso, una estación no puede estar segura que ha tomado el canal hasta que ha transmitido durante $2t$ sin detectar una colisión.

La detección de colisiones es un proceso analógico. El hardware de la estación debe escuchar el cable mientras transmite. Si lo que lee es distinto de lo que puso en él, sabe que está ocurriendo una colisión. La implicación es que la codificación de la señal debe permitir que se detecten colisiones (por ejemplo, una colisión de dos señales de 0 volts bien podría ser imposible de detectar). Por esta razón, comúnmente se usa una codificación especial.

También vale la pena mencionar que una estación emisora debe monitorear de manera continua el canal en busca de ráfagas de ruido que puedan indicar una colisión. Por esta razón, CSMA/CD con un solo canal, es inherentemente un sistema semiduplex.

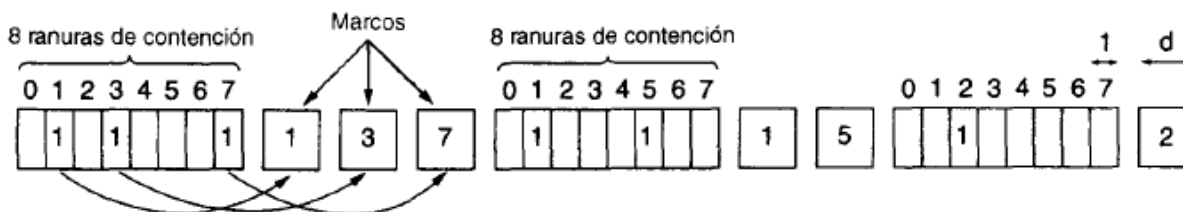
Es importante hacer notar que ningún protocolo de subcapa MAC garantiza la entrega confiable.

Protocolos Libres de Colisiones

Las colisiones en CSMA/CD pueden ocurrir durante el periodo de contención. En los protocolos que describiremos, suponemos que hay N estaciones, cada una con una dirección única de 0 a $N-1$ incorporada en hardware. Damos por hecho que el retardo de propagación no importa.

Protocolo de mapa de Bits

Cada periodo de contención consiste en exactamente N ranuras. Si la estación 0 tiene una trama por enviar, transmite un bit 1 durante la ranura 0. No está permitido a ninguna otra estación transmitir durante esa ranura. En general, la estación j puede anunciar que tiene una trama por enviar introduciendo un bit 1 en la ranura j . Una vez que han pasado las N ranuras, cada estación tiene conocimiento completo de las estaciones que quieren transmitir. En ese punto, las estaciones comienzan a transmitir en orden numérico



Dado que todos están de acuerdo en quién continúa, nunca habrá colisiones. Una vez que la última estación lista haya transmitido su marco, evento que pueden detectar fácilmente todas las estaciones, comienza otro periodo de contención de N bits.

Protocolos como éste, en los que el deseo de transmitir se difunde antes de la transmisión, se llaman **protocolos de reservación**. El problema de este protocolo, es la sobrecarga de 1 bit por estación, por lo que no escala bien en redes con miles de estaciones.

Conteo Descendente Binario

Podemos tener mejores resultados usando direcciones de estación binarias. Una estación que quiere utilizar el canal ahora difunde su dirección como una cadena binaria de bits. Se supone que todas las direcciones tienen la misma longitud. A los bits en cada posición de dirección de las distintas estaciones, se les aplica un OR booleano a todos juntos. Asume retardos insignificantes.

Para evitar conflictos, debe aplicarse una regla de arbitraje: una vez que una estación ve que una posición de bit de orden alto, que en su dirección es 0, ha sido sobrescrita con un 1, se da por vencida. Tras ganar la contienda, ahora puede transmitir una trama, después de lo cual, comienza otro ciclo de contienda.

Este protocolo tiene la propiedad de que las estaciones con nº grandes tienen una prioridad mayor que las estaciones con nº pequeños, lo cual puede ser bueno o mal según el contexto.

Mok y Ward sugieren el uso de nº virtuales de estación, que se permutan en forma circular tras cada transmisión a fin de darle mayor prioridad a las estaciones que han estado en silencio durante mucho tiempo.



Protocolos de Contención Limitada

Cada estrategia puede ser clasificada según lo bien que funciona en relación con las 2 medidas de desempeño: el retardo con carga baja y la eficiencia del canal con carga alta. En condiciones de carga baja, la contención (es decir, ALOHA puro o ranurado) es preferible debido a su bajo retardo. A medida que aumenta la carga, la contención paulatinamente se vuelve menos atractiva, debido a que la información extra asociada al arbitraje del canal se vuelve mayor. Lo inverso se cumple para los protocolos libres de colisiones. Con carga baja, tienen un retardo alto, pero a medida que aumenta la carga mejora la eficiencia del canal en lugar de empeorar, como ocurre con los protocolos de contención.

Obviamente, sería agradable si pudiéramos combinar las mejores propiedades de los protocolos de contención y los libres de colisiones, desarrollando un protocolo nuevo que usara contención cuando la carga es baja para así tener un retardo bajo, y una técnica libre de colisiones cuando la carga es alta para lograr una buena eficiencia de canal. Estos son los **protocolos de contención limitada**.

Hasta ahora, los únicos protocolos de contención que hemos estudiado han sido simétricos, es decir, cada estación intenta adquirir el canal con alguna probabilidad p , usando todas las estaciones la misma p . Resulta interesante que el desempeño general del sistema a veces puede mejorarse usando un protocolo que asigne diferentes probabilidades a diferentes estaciones.

La probabilidad de que una estación adquiera el canal sólo puede aumentar disminuyendo la cantidad de competencia. Los protocolos de contención limitada hacen precisamente eso. Primero dividen las estaciones en grupos (no necesariamente excluyentes). Sólo los miembros del grupo 0 pueden competir por la ranura 0. Si uno de ellos tiene éxito, adquiere el canal y transmite su marco. Si la ranura permanece desocupada o si hay una colisión, los miembros del grupo 1 contienden por la ranura 1, etc.

El truco está en cómo asignar las estaciones a las ranuras. Lo que necesitamos es una manera de asignar las estaciones a las ranuras de manera dinámica, con muchas estaciones por intervalo cuando la carga es baja y pocas estaciones (o incluso sólo una) por ranura cuando la carga es alta.

Protocolo de recorrido de árbol adaptable

Una manera particularmente sencilla de llevar a cabo la asignación necesaria es usar el algoritmo desarrollado por el ejército de Estados Unidos para hacer pruebas de sífilis a los soldados durante la Segunda Guerra Mundial. En pocas palabras, el ejército tomaba una muestra de sangre de N soldados. Se vaciaba una parte de cada muestra en un solo tubo de ensayo. Luego, esta muestra mezclada era examinada en busca de anticuerpos. Si no se encontraban, todos los soldados del grupo eran declarados sanos. Si se encontraban anticuerpos, se preparaban dos muestras mezcladas nuevas, una de los soldados 1 a $N/2$ y otra de los demás. El proceso se repetía recursivamente hasta que se determinaban todos los soldados infectados.

Para la versión de computadora de este algoritmo es conveniente pensar en las estaciones como las hojas de un árbol binario. En la primera ranura de contención después de la transmisión satisfactoria de una trama, ranura 0, se permite que todas las estaciones intenten adquirir el canal. Si una de ellas lo logra, que bueno. Si hay una colisión, entonces, durante la ranura 1, sólo aquellas estaciones que queden bajo el nodo 2 del árbol podrán competir. Si una de ellas adquiere el canal, la ranura que sigue a la trama se reserva para las estaciones que están bajo el nodo 3. Por otra parte, si dos o mas estaciones bajo el nodo 2 quieren transmitir, habrá una colisión durante la ranura 1, en cuyo caso es el turno del nodo 4 durante la ranura 2.

En esencia, si ocurre una colisión durante la ranura 0, se examina todo el árbol, primero en profundidad, para localizar todas las estaciones listas. Cada ranura de bit está asociada a un nodo en particular del árbol. Si ocurre una colisión, continúa la búsqueda recursivamente con los hijos izquierdos y derechos del nodo. Si una ranura de bit está inactiva, o si sólo hay una estación que transmite en ella, puede detenerse la búsqueda de su nodo, pues se han localizado todas las estaciones listas. (Si hubiera más de una, habría habido una colisión.)

¿En qué nivel del árbol debe comenzar la búsqueda? Es obvio que, a mayor carga, la búsqueda debe comenzar más abajo en el árbol.

Protocolos de Acceso Múltiple por división de longitud de onda (WDMA)

Un método diferente para la asignación del canal es dividir el canal en subcanales, usando FDM, TDM, o ambas, y asignarlos de manera dinámica según se necesite. Se usa en las LANs de Fibra.

Para permitir múltiples transmisiones al mismo tiempo, se divide el espectro en canales (bandas de longitud de onda). En este protocolo, se asignan dos canales a cada estación. Se proporciona un canal estrecho como canal de control para enviar señales a la estación, y se proporciona un canal ancho para que la estación pueda enviar tramas de datos. Todos los canales se sincronizan con un solo reloj global. El protocolo reconoce tres clases de tráfico: tráfico orientado a conexión con tasa de datos constante (Ej: [video sin comprimir](#)) tráfico orientado a conexión con tasa de datos variable (Ej: [transferencia de archivos](#)), y tráfico de datagramas (Ej: [paquetes UDP](#)).

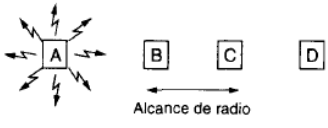
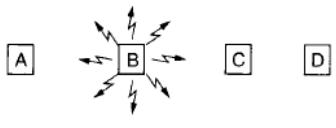
Cada estación tiene dos transmisores y dos receptores, como sigue:

1. Un receptor de longitud de onda fija para escuchar su propio canal de control.
2. Un emisor sintonizable para enviar por el canal de control de otra estación.
3. Un emisor de longitud de onda fija para la salida de marcos de datos.
4. Un receptor sintonizable para seleccionar el transmisor de datos a escuchar.

En otras palabras: cada estación escucha en su propio canal de control las solicitudes que llegan, pero tiene que sintonizarse a la longitud de onda del transmisor para obtener los datos. La sintonización de la longitud de onda se realiza con un interferómetro que filtra todas las longitudes de onda excepto la banda de longitud de onda deseada.

Protocolos de LANs Inalámbricas

Un enfoque inocente para usar una LAN inalámbrica podría ser intentar el CSMA; escuchar si hay otras transmisiones y sólo transmitir si nadie más lo está haciendo. El problema radica en que este protocolo no es realmente adecuado, porque lo que importa es la interferencia en el receptor, no en el emisor. Hay 2 problemas principales:

- 1. Problema de la estación oculta:** Es el problema por el cual una estación no puede detectar a un competidor potencial por el medio, puesto que este competidor está demasiado lejos. Si A está transmitiendo hacia B, como muestra la figura. Si C detecta el medio, no podrá escuchar a A porque está fuera de su alcance, y por tanto, deducirá falsamente que puede transmitir a B. Si C comienza a transmitir, interferirá en B, eliminando la trama de A.
- 2. Problema de la estación inversa (expuesta):** Es lo opuesto: B transmitiendo hacia A. Si C detecta el medio, escuchará una transmisión, y concluirá falsamente que no puede enviar a D, cuando de hecho, tal transmisión causaría una mala recepción solo en la zona entre B y C, en la que no está localizado ninguno de los receptores pretendidos.

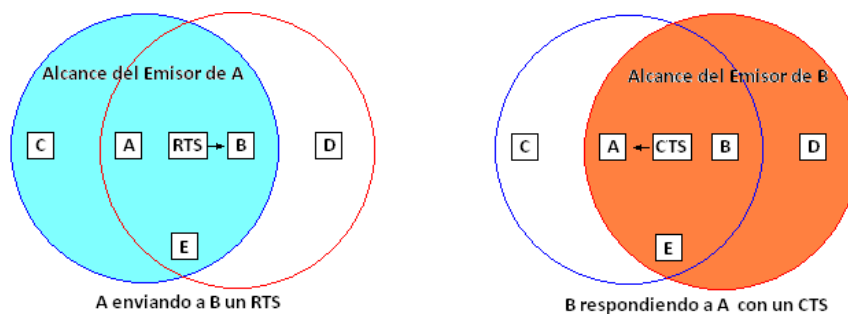
El problema es que antes de comenzar una transmisión, una estación realmente necesita saber si hay actividad o no alrededor del receptor. El CSMA simplemente le indica si hay o no actividad alrededor de la estación que está detectando la portadora.

MACA (Acceso Múltiple con Prevención de Colisiones) y MACAW (MACA Inalámbrico)

MACA se basa en que el transmisor estimule al receptor a enviar una trama corta, de manera que las estaciones cercanas puedan detectar esta transmisión y eviten ellas mismas interferir durante el siguiente marco de datos (grande).

Consideremos ahora la manera en que A envía una trama a B. A comienza por enviar una trama **RTS (Solicitud de envío)** a B. Esta trama corta (30 bytes) contiene la longitud de la trama de datos que seguirá posteriormente. Entonces B contesta con una trama **CTS (Libre para envío)**. La trama CTS contiene la longitud de los datos (copiada de la trama RTS). A una vez que se recibe la trama CTS, A comienza la transmisión.

Ahora veamos cómo reaccionan las estaciones que escuchan cualquiera de estas tramas. Cualquier estación que escuche el RTS evidentemente está bastante cerca de A, y debe permanecer en silencio durante el tiempo suficiente para que el CTS se transmita de regreso a A sin conflicto. Cualquier estación que escuche el CTS evidentemente está bastante cerca de B y debe permanecer en silencio durante el siguiente tiempo de transmisión de datos, cuya longitud puede determinar examinando el marco CTS.



A pesar de estas precauciones, aún pueden ocurrir colisiones. Por ejemplo, B y C pueden enviar tramas RTS a A al mismo tiempo. Estas chocarán y se perderán. Para evitar esto, se usa el algoritmo de retroceso exponencial binario.

Luego de estudios, se desarrolló **MACAW**. Expertos notaron que, sin confirmación de recepción en la capa de enlace de datos, las tramas no eran retransmitidas sino hasta que la capa de transporte notaba su ausencia, mucho después. Resolvieron este problema introduciendo una trama ACK tras cada trama de datos exitosa. Se agregó la detección de portadora, y decidieron ejecutar el algoritmo de retroceso por separado para cada flujo de datos, en lugar para cada estación. Este cambio mejora la equidad del protocolo. Por último, agregaron un mecanismo para que las estaciones intercambiaran información sobre congestionamientos, y una manera de hacer que el algoritmo de retroceso reaccionara menos violentamente a problemas pasajeros, con lo que mejoraron el desempeño del sistema.

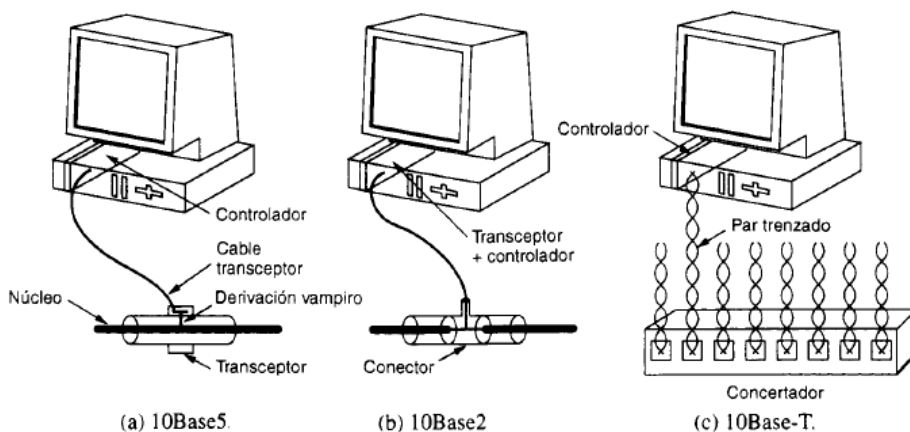
ETHERNET

El IEEE ha estandarizado varias redes LAN y MAN bajo el nombre de IEEE 802. Los sobrevivientes más importantes son el 802.3 (Ethernet) y el 802.11 (LAN Inalámbrica). Ambos tienen diferentes capas físicas y subcapas MAC, pero convergen en la misma subcapa de control lógico de enlace (LLC), por lo que tienen la misma interfaz a la capa de red.

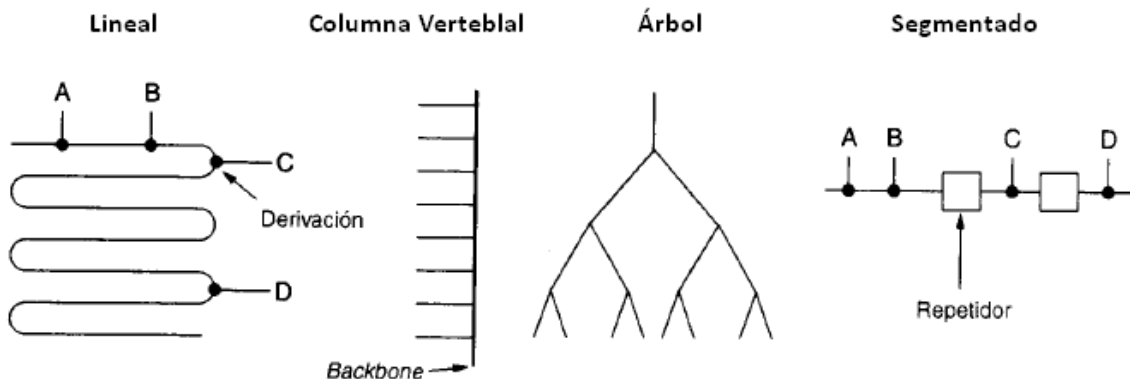
Ethernet es un estándar de redes de computadoras de área local con acceso al medio por contienda CSMA/CD. El nombre viene del concepto físico de ether. Ethernet define las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI.

Cableado Ethernet

Nombre	Cable	Segmento Máximo	Nodos /seg.	Ventajas	Conector	Desventajas
10Base5 (10 Mbps) (~5*100 m)	Coaxial Grueso	500 m	100	Cable original. Ahora obsoleto	Derivaciones vampiro	Difícil detectar rupturas en el cable.
10Base2	Coaxial Delgado	185 m	30	No necesita hub. Se dobla con facilidad	BNC. Más fáciles de usar y mas confiables	Se detecta mediante reflectometría en el dominio del tiempo
10BaseT	Par Trenzado	100 m	1024	Económico. Fácil de configurar	Al hub	Esquema concentrado
10BaseF	Fibra Óptica	2000 m	1024	Inmune al ruido, seguridad, distancia	Convertor óptico-digital	Costosa



Topologías de Cableado



Codificación Manchester

Se necesita un mecanismo para que los receptores determinen sin ambigüedades el comienzo, final o mitad de cada bit sin referencia a un reloj externo. Dos de tales enfoques son la codificación Manchester y Manchester diferencial. Tienen como desventaja que requieren el doble de ancho de banda que la codificación binaria directa, pues los pulsos son de la mitad de ancho.

Todos los sistemas Ethernet usan codificación Manchester debido a su sencillez. La señal alta es de +0.85 voltios, y la señal baja es de -0.85 voltios, dando un valor de Corriente Continua de 0 voltios.

El protocolo de subcapa MAC de Ethernet

Trama	8	6	6	2	0-1500	0-46	4
DIX	Preámbulo	Dir. Destino	Dir. Origen	Tipo	Datos	Relleno	Checksum
Trama	8	6	6	2	0-1500	0-46	4
802.3	Preámbulo+sof	Destino	Origen	Longitud	Datos	Relleno	Checksum

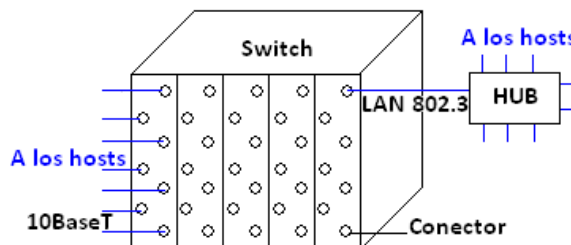
A medida que aumente la velocidad de la red, la longitud mínima de la trama debe aumentar, o la longitud máxima del cable debe disminuir, de manera proporcional (debido al problema de los 2t).

Ethernet Utiliza el algoritmo de retroceso exponencial binario comentado anteriormente.

Ethernet conmutada

A medida que se agregan más y más estaciones a una Ethernet, aumenta el tráfico. Se puede aumentar la velocidad, pero tarde o temprano se saturará igual.

Existe una solución diferente para tratar con el aumento de carga: una Ethernet Conmutada. El corazón de este sistema es un **conmutador (switch)** que contiene una matriz de conmutación de alta velocidad, y espacio para varias tarjetas de línea, cada una de las cuales contiene de uno a 8 conectores. Lo más común es que cada conector tenga una conexión de par trenzado 10BaseT a una sola computadora host.



Cuando una estación quiere transmitir una trama Ethernet, envía una trama estándar al switch. La tarjeta que recibe la trama la revisa para ver si está destinada a una de las otras estaciones conectadas a la misma tarjeta. De ser así, la trama se copia ahí; si no, se envía a través del canal en segundo plano de alta velocidad a la tarjeta de la estación de destino. El canal de alta velocidad funciona a más de 1 Gbps usando un protocolo patentado. Con este diseño, **cada tarjeta forma su propio dominio de colisión**, independiente de los demás.

Con el otro tipo de tarjeta, cada puerto de entrada se maneja con un buffer, por lo que las tramas de entrada se almacenan en la RAM de la tarjeta según llegan. Este diseño permite que todos los puertos de entrada reciban (y transmitan) tramas al mismo tiempo, para operación dúplex integral paralela. Una vez que se ha recibido por completo la trama, la tarjeta puede determinar si el marco está destinado a otro puerto de la misma tarjeta, o a un puerto distante. En el primer caso, puede transmitirse directamente al destino. En el segundo, debe transmitirse a través del canal de alta velocidad a la tarjeta apropiada. Con este diseño, **cada puerto es un dominio de colisión independiente, por lo que no ocurren colisiones**.

Dado que el conmutador sólo espera marcos 802.3 en cada puerto de entrada, es posible usar algunos de los puertos como concentradores. Los concentradores son más baratos que los switches, pero debido a los precios de los últimos, se están haciendo obsoletos rápidamente.

Fast Ethernet (802.3u)

En 1992 la IEEE convocó al comité 802.3 para hacer una LAN más rápida. La idea era mantener compatibilidad hacia atrás, y sólo reducir el tiempo de bits de 100 nseg a 10 nseg.

Las ventajas del cableado 10BaseT eran tan abrumadoras, que Fast Ethernet se basa por completo en este diseño, por lo que también se usan switches y hubs.

Sin embargo, aún se debía determinar que tipos de cables soportar (además de 10BaseT). El cable par trenzado UTP3 tenía la ventaja que ya estaba instalado (red telefónica), pero como desventaja, no podía llevar señales de 100 Mbps en Manchester hasta 100 metros. En contraste, UTP5 puede manejar esto con facilidad, al igual que la fibra. Se decidió usarlo también, pero para alcanzar esta velocidad, en vez de utilizar codificación Manchester, se usó 8B/6T (8bits-6trits).

Nombre	Cable	Seg. Max.	Ventajas	Codificación
100Base-T4	Par Trenzado	100 m	UTP3 (telefónico)	8B/6T
100Base-TX	Par Trenzado	100 m	Full Duplex 100 Mbps	Manchester
100Base-FX	Fibra Óptica	2000 m	Full Duplex 100 Mbps. No permite Hubs. Distancias Largas.	4B/5B

Con 100BaseT son posibles 2 tipos de dispositivos de interconexión: hubs y switches. Con los **hubs**, todas las líneas entrantes se conectan lógicamente, formando un solo dominio de colisión. Se aplican todas las reglas estándar, por lo que el sistema funciona como la Ethernet antigua: una sola estación por vez puede transmitir (semidúplex). Con los **switches**, cada trama entrante se almacena en el búfer de una tarjeta de conexión y se para a través de una matriz de conmutación de alta velocidad de la tarjeta de origen a la de destino, si es necesario.

Gigabit Ethernet (802.3z)

Se busco, al igual que con Fast Ethernet, aumentar 10 veces la velocidad, y que fuera compatible hacia atrás. En particular, tiene que ofrecer servicio de datagramas sin confirmación de recepción con difusión y multidifusión, utilizar el mismo esquema de direccionamiento y de trama.

Todas las configuraciones de Gigabit Ethernet son de punto a punto. Soporta 2 modos:

- **Full Duplex:** Se utiliza cuando hay un switch central conectado a otros hosts periféricos. Todas las líneas se almacenan en el búfer a fin de que cada computadora y switch pueda enviar tramas siempre que lo desee. El emisor no tiene que detectar el canal ya que no es posible una situación de contención. Por esto, no se usa CSMA/CD y la longitud máxima del cable se determina con base a la fuerza de la señal. Los conmutadores son libres de igualar y mezclar velocidades
- **Half Duplex:** Se utiliza hay un hub central. El hub no almacena en búfer las tramas entrantes. En su lugar, conecta en forma eléctrica todas las líneas internamente, simulando un cable entre el origen y el destino. Aquí son posibles las colisiones, por lo que se usa CSMA/CD.

Como la trama ahora puede transmitirse 100 veces más rápido que en Ethernet clásica, la longitud del cable se achica 4 veces, a 25 metros. Esto era inaceptable, por lo que se desarrollaron 2 técnicas para extender esto:

- Extensión de portadora: se extiende la longitud de la trama a 512 bytes. No requiere cambios al software
- Ráfagas de Trama: Permite enviar muchas tramas en una sola. Se prefiere este método.

Nombre	Cable	Seg. Max.	Ventajas
1000Base-SX	Fibra Óptica	550 m	Fibra multimodo (50 – 62.5 micras)
1000Base-LX		5000 m	Sencilla o multimodo
1000Base-CX	2 pares STP	25 m	Par Trenzado Blindado
1000Base-T	2 pares UTP	100 m	UTP Cat. 5 estándar

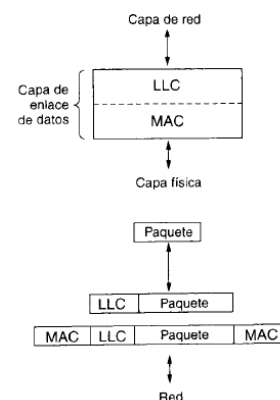
802.2: Control Lógico de Enlace (LLC)

En capítulos anteriores vimos la manera en que dos máquinas se podían comunicar de manera confiable a través de una línea inestable usando varios protocolos de enlace de datos, que proporcionaban control de errores (mediante acuses) y control de flujo (usando una ventana corrediza).

Todo lo que ofrecen las LAN y las MAN 802 es un servicio de datagramas que hace su mejor esfuerzo. Hay sistemas en los que se desea un protocolo de enlace de datos con control de errores y control de flujo. El IEEE ha definido uno que puede operar encima de todos los protocolos LAN y MAN 802, **LLC**, que esconde las diferencias entre los diferentes tipos de redes 802, proporcionando un formato único y una interfaz con la capa de red.

El uso típico del LLC es el siguiente. La capa de red de la máquina transmisora pasa un paquete al LLC usando las primitivas de acceso del LLC. La subcapa LLC agrega una cabecera LLC que contiene los números de secuencia y acuse. La estructura resultante se introduce entonces en el campo de carga útil de un marco 802.x y se transmite. En el receptor ocurre el proceso inverso.

El LLC proporciona tres opciones de servicio: servicio no confiable de datagramas, servicio reconocido de datagramas y servicio confiable orientado a conexión. El encabezado LLC tiene 3 campos: un punto de acceso de destino, un punto de acceso de origen, y un campo de control. Los puntos de acceso indican de cuál proceso proviene la trama y a donde se va a enviar.



LANS INALÁMBRICAS (802.11)

En el estándar 802.11, la subcapa MAC determina la forma en que se asigna el canal. Arriba de esta subcapa, se encuentra la subcapa LLC, cuyo trabajo es ocultar las diferencias entre las variantes 802, de modo que sean imperceptibles en la capa de red.

En la **capa física**, hay 5 técnicas permitidas: Infrarrojos, FHSS, DSSS, OFDM, HR-DSSS. Cada una de estas técnicas difieren en la tecnología utilizada y en las velocidades alcanzables:

1. **Infrarrojos:** Utiliza transmisión difusa (no requiere línea visual). Trabaja a 1 o 2 Mbps. Estas señales no pueden penetrar paredes. No es muy popular
2. **FHSS (Espectro Disperso con Salto de Frecuencia):** Utiliza 79 canales de 1MHz. Para producir la secuencia de frecuencias a saltar, se utiliza un generador de nº pseudo aleatorios. El tiempo de permanencia en cada frecuencia es ajustable. Es un método seguro, relativamente insensible a la interferencia, pero de bajo ancho de banda.
3. **DSSS (Espectro Disperso de Secuencia Directa):** 1 o 2 Mbps. Usa modulación por fase.
4. **OFDM (Multiplexión por División de Frecuencias Ortogonales), 802.11a:** Permite hasta 54 Mbps. Se utilizan 52 frecuencias. Esta división permite obtener mejor inmunidad a la interferencia y la posibilidad de usar bandas no contiguas. Usa modulación por desplazamiento de fase hasta 18 Mbps y QAM para velocidades mayores
5. **HR-DSSS (DSSS de alta velocidad):** Alcanza 11 Mbps en la banda de 2.4 GHz, y también soporta 1, 2 y 5.5 Mbps. La tasa de datos es adaptable. Tiene más alcance que OFDM.

El protocolo de subcapa MAC del 802.11

Con Ethernet, una estación simplemente espera hasta que el medio está en silencio y comienza a transmitir. Si no recibe una ráfaga de ruido dentro de los primeros 64 bytes, con seguridad la trama se ha entregado exitosamente. Esto no funciona en los sistemas inalámbricos.

Recordemos que está el problema de la estación oculta y la estación expuesta. Además, la mayoría de los radios son semiduplex, lo que significa que no pueden transmitir y escuchar ráfagas de ruido al mismo tiempo en una sola frecuencia. Como resultado de esto, 802.11 no usa CSMA/CD.

Para solucionar estos problemas, 802.11 soporta 2 modos de funcionamiento:

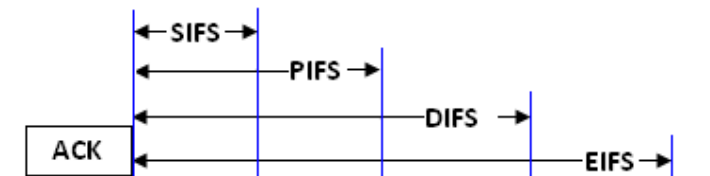
- **DCF (Función de Coordinación Distribuida):** No usa control central (similar a Ethernet). Cuando se usa DCF, se utiliza un protocolo llamado CSMA/CA (CSMA con evitación de colisiones). Este protocolo se utiliza en 2 métodos:
 - **Detección del canal físico:** Cuando una estación desea transmitir, detecta el canal. Si está inactivo, empieza a transmitir. NO detecta el canal mientras transmite, pero emite su trama completa, la cual podría ser destruida en el receptor debido a interferencia. Si el canal está ocupado, el emisor espera hasta que esté inactivo. Si ocurre una colisión, las estaciones involucradas en ella esperan un tiempo aleatorio, mediante el algoritmo de retroceso exponencial, y lo intentan más tarde.
 - **Detección del canal virtual:** Se basa en MACAW. La estación que detecta el RTS o CTS, puede estimar cuánto tardará la secuencia, incluido el ACK final, por lo que impone a sí misma un tipo de canal ocupado, indicado por **NAV (Vector de Asignación de Red)**. Esta es una señal interna, no se transmite, e indica silencio.

Como las redes inalámbricas son ruidosas e inestables, si una trama es muy grande, hay grandes probabilidades que deba retransmitirse por daños en el camino. Esto se solucionó

permitiendo dividir las tramas en fragmentos, cada uno con su suma de verificación. Una vez adquirido un canal mediante CTS y RTS, pueden enviarse ráfagas de fragmentos.

- **PCF (Función de Coordinación Puntual):** Utiliza la estación base para controlar toda la actividad en su celda. La estación base sondea a las demás estaciones, preguntándoles si tienen tramas para enviar. No hay colisiones ya que la estación base determina el orden. La frecuencia de sondeo no está definida. Lo único que se define es el mecanismo, que consiste en que la estación base difunda una **trama de beacon** de manera periódica, que contiene parámetros del sistema. También indica a las nuevas estaciones a suscribirse al servicio de sondeo. Una vez que una estación se suscribe al sondeo, se le garantiza cierta fracción del ancho de banda y se hace posible garantizarle calidad de servicio.

PCF y DCF pueden coexistir en una celda. Funciona definiendo cuidadosamente el intervalo de tiempo entre tramas. Después de que se ha enviado una trama, se necesita cierta cantidad de tiempo muerto antes de que cualquier estación pueda enviar una trama. Se definen 4 intervalos diferentes, cada uno con un propósito específico.



- SIFS (Espaciado Corto entre Tramas): Se permiten CTS, RTS y ACK
- PIFS (Espaciado entre tramas PCF): Permite distribuir tramas beacon
- DIFS (Espaciado entre tramas DCF): Cualquier estación puede intentar adquirir el canal
- EIFS (Espaciado entre tramas Extendido): Se usa para reportar errores (NAK).

Servicios

Cada LAN Inalámbrica que se apegue al estándar debe proporcionar:

- **Servicios de Distribución:** Son proporcionados por las estaciones base, y tienen que ver con la movilidad de la estación, conforme entran y salen de las celdas
 1. Asociación: Es utilizado por las estaciones móviles para conectarse ellas mismas a las estaciones base. Se utiliza después de que una estación entra a una celda. Una vez que llega, anuncia su identidad y capacidades. La estación base puede aceptar o rechazar la estación móvil. Si la acepta, debe autenticarse.
 2. Disociación: Es cuando la estación se disocia de la estación base
 3. Reasociación: Se usa cuando una estación cambia de celda (y de estación base)
 4. Distribución: Determina como enrutar las tramas enviadas a la estación base
 5. Integración: Maneja la traducción del formato 802.11 al requerido por la red destino.
- **Servicios de Estación:** Se relacionan con la actividad dentro de una celda
 1. Autenticación: Una estación debe autenticarse antes de que pueda transmitir
 2. Desautenticación
 3. Privacidad: Son temas referidos a la codificación de la información
 4. Entrega de datos: Es la transmisión de datos. No se garantiza que sea confiable.

BANDA ANCHA INALÁMBRICA

El problema es que el tendido de fibra óptica, cable coaxial, o incluso UTP5, a 5 millones de casas y oficinas es extremadamente costoso. ¿Qué debe hacer un competidor? Usar banda ancha inalámbrica. Construir una antena enorme en una colina en las afueras del pueblo e instalar antenas que se dirijan a dicha antena en los techos de los clientes. Esto es más barato que cavar zanjas y colocar cables. Sin embargo, hasta hace poco, cada ISP diseñaba su propio sistema. Esta falta de estándares significaba que el hardware y software no se podía producir en masa, por lo que los precios eran altos, y la aceptación baja.

Por esto, se diseñó el estándar **802.16**, llamado **MAN inalámbrica** o **circuito local inalámbrico**.

¿Por qué no usar 802.11?

Principalmente, porque 802.11 y 802.16 resuelven diferentes problemas. 802.16 proporciona servicio a edificios, y éstos no son móviles. No migran de celda a celda con frecuencia. La mayor parte de 802.11 tiene que ver con la movilidad, y nada de eso es relevante aquí. Además, los edificios pueden tener más de una computadora en ellos, lo cual no ocurre cuando la estación final es una sola computadora portátil.

Debido a que los dueños de edificios por lo general están dispuestos a gastar mucho más dinero en artículos de comunicación que los dueños de computadoras portátiles, hay mejores radios disponibles. Esta diferencia significa que 802.16 puede utilizar comunicación de duplex total, algo que 802.11 evita para mantener bajo el costo de los radios.

Como 802.16 se usa en la ciudad, las distancias involucradas pueden ser de varios kilómetros, lo que significa que la energía detectada en la estación base puede variar considerablemente de estación en estación. Además, la comunicación abierta a través de la ciudad significa que la seguridad y privacidad son esenciales y obligatorias.

Es probable que una celda tenga muchos más usuarios que una celda 802.11 típica, y se espera mucho más uso de ancho de banda, con lo que se obliga al estándar 802.16 a funcionar en el rango de frecuencia más alto, de 10 a 66 GHz.

Estas ondas milimétricas son absorbidas por completo por el agua. En consecuencia, el control de errores es más importante que en un entorno interno. Las ondas milimétricas pueden enfocarse en rayos direccionales, por lo que no se requiere la propagación en múltiples ondas como 802.11.

Otro aspecto es la calidad del servicio, ya que 802.16 soporta tráfico en tiempo real.

En resumen, 802.11 se diseñó para ser una Ethernet móvil, mientras que 802.16 se diseñó para ser TV por cable inalámbrica, pero estacionaria. Estas diferencias son tan grandes que los estándares resultantes son muy diferentes, debido a que tratan de optimizar cosas distintas.

La pila de protocolos del estándar 802.16

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

A diferencia de otras subcapas MAC, ésta es completamente orientada a la conexión, para proporcionar garantías de QoS para la comunicación de telefonía y multimedia.

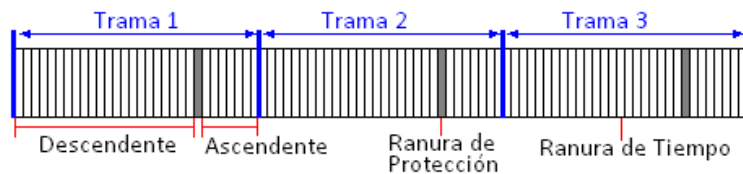
La Capa Física de 802.16

La banda ancha inalámbrica necesita mucho espectro, y el único lugar disponible es de 10 a 66 GHz. Estas ondas milimétricas viajan en líneas rectas. Cada sector tiene sus propios usuarios, y es completamente independiente de los sectores adyacentes.

Debido a que la fuerza de señal en la banda milimétrica desciende drásticamente con la distancia a partir de la estación base, la relación señal a ruido también desciende con la distancia a partir de la estación base. Por esta razón, el estándar 802.16 utiliza 3 esquemas de modulación diferentes, dependiendo de la distancia entre la estación suscriptora y la estación base:

- QAM-64 para sectores cercanos, con 6 bits/baudio
- QAM-16 para distancias medias, con 4 bits/baudio
- QPSK para sectores distantes

En cuanto al ancho de banda ascendente y descendente, se asigna de una forma flexible. Se utilizan 2 esquemas: **FDD (Duplexación por División de Frecuencia)** y **TDD (Duplexación por División de Tiempo)**. En TDD, la estación base envía tramas periódicamente. Cada trama contiene ranuras de tiempo. Las primeras son para el tráfico descendente. Después se encuentra el tiempo de protección o guarda, el cual es usado por las estaciones para cambiar de dirección. Por último están las ranuras para el tráfico ascendente. El nº de ranuras dedicadas para cada dirección se puede cambiar dinámicamente, con el fin de que el ancho de banda en cada dirección coincida con el tráfico. La estación base asigna el tráfico descendente, mientras que el ascendente depende de la QoS.



Otra característica interesante de esta capa, es la capacidad para empaquetar múltiples tramas MAC consecutivas en una sola transmisión física

Protocolo de subcapa MAC del 802.16

Hay 4 clases de servicio definidas:

1. Servicio de tasa de bits constante: para transmitir voz descomprimida. Este servicio necesita enviar una cantidad predeterminada de datos en intervalos de tiempo predeterminados
2. Servicio de tasa de bits variable en tiempo real: destinado para multimedia comprimida y otras aplicaciones que requieran ancho de banda dinámico. Es ajustada por la estación base sondeando al suscriptor a un intervalo fijo para saber cuánto ancho de banda se necesita esta vez
3. Servicio de tasa de bits variable no en tiempo real: es para las transmisiones grandes de archivos. Para este servicio, la estación base sondea al suscriptor con mucha frecuencia
4. Servicio de mejor esfuerzo: es para todo lo demás. No se realiza sondeo, y el suscriptor debe competir con el resto de los suscriptores por ancho de banda.

Todos los servicios son orientados a la conexión. Hay 2 formas de asignar ancho de banda, por estación (edificio) y por conexión.

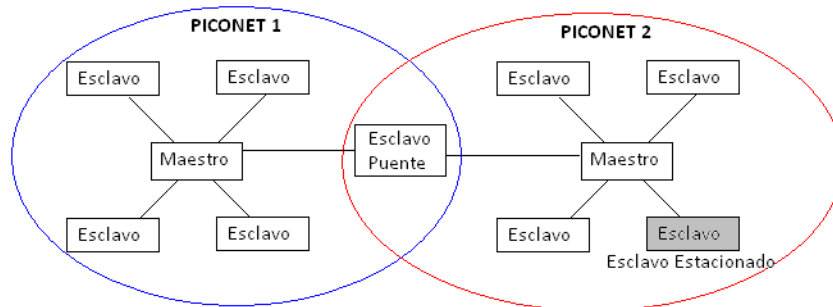
BLUETOOTH

En 1994, Ericsson se interesó en interconectar computadoras, dispositivos de comunicaciones y accesorios a través de radios inalámbricos de bajo consumo de energía, corto alcance y baratos.

Aunque la idea original era tan solo evitar cables entre dispositivos, su alcance se propagó al área de las LANs inalámbricas, compitiendo con 802.11. Para empeorar las cosas, los 2 sistemas interfieren entre sí en el ámbito eléctrico.

Arquitectura de Bluetooth

La unidad básica de un sistema Bluetooth es una **piconet**, que consta de un nodo maestro y hasta 7 nodos esclavos activos a una distancia de 10 metros. En una misma sala pueden encontrarse varias piconets, y se pueden conectar mediante un nodo puente. Un conjunto de piconets interconectadas se denomina **scatternet**.



Además de los 7 nodos esclavos activos de una piconet, puede haber hasta 255 nodos estacionados en la red. Estos son dispositivos que el nodo maestro ha cambiado a un estado de bajo consumo de energía para reducir el desgaste innecesario de sus pilas.

La razón para el diseño maestro-esclavo es que los diseñadores pretendían facilitar la implementación de chips Bluetooth completos por debajo de 5 USD. La consecuencia de esta decisión es que los esclavos son sumamente pasivos y realizan todo lo que los maestros les indican.

Aplicaciones de Bluetooth

La especificación v1.1 especifica el soporte a 13 aplicaciones, las cuales se denominan **perfiles**. Estos los diseñaron distintos grupos de trabajo, y cada uno se enfocó en su problema específico, creando su propia pila de protocolos para cada perfil. Quizá 2 pilas de protocolos habrían sido suficientes en lugar de 13, una para la transferencia de archivos y otra para posibilitar el flujo continuo de la comunicación en tiempo real.

La pila de protocolos de Bluetooth

La capa inferior es la de **radio física**, que se ocupa de la transmisión y la modulación de radio. Aquí, gran parte del interés se enfoca en el objetivo de lograr que el sistema tenga un costo bajo para que pueda entrar al mercado masivo.

La **capa de banda base** tiene algunos puntos en común con la subcapa MAC y de la física. Se encarga de la manera en que el maestro controla las ranuras de tiempo y de que éstas se agrupen en tramas.

Luego viene una capa con un grupo de protocolos de control, audio, etc. Le sigue la de middleware, y luego las aplicaciones.

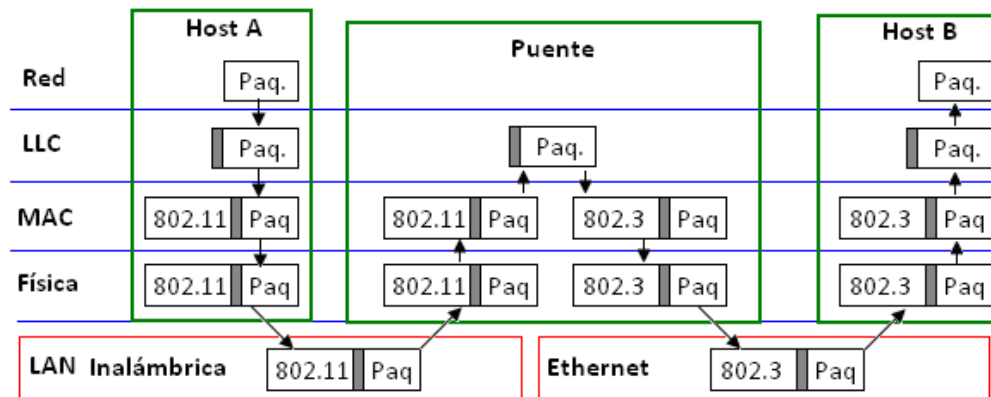
CONMUTACIÓN EN LA CAPA DE ENLACE DE DATOS

Cuando se quieren interconectar LANs, se utilizan dispositivos llamados **puentes**. Los mismos examinan las direcciones de la capa de enlace de datos para enrutar los datos. En contraste, los **enrutadores** examinan las direcciones de los paquetes y realizan su trabajo de enrutamiento con base en ellas.

Hay varias situaciones comunes en las cuales se utilizan los puentes. Por ejemplo:

1. Universidades y departamentos. Hay interacción y se requieren puentes
2. La organización puede estar distribuida geográficamente en varios edificios. Puede ser más barato hacer una LAN por edificio, e interconectarlas mediante puentes
3. Puede ser necesario dividir una LAN en varias LANs individuales para manejar la carga
4. Para dividir la distancia física entre las LAN
5. Para aislar errores y problemas
6. Para mejorar la seguridad

Puentes de 802.x a 802.y



Observe que un puente que conecta k LANs diferentes, tendrá k subcapas MAC diferentes y k capas físicas diferentes, una para cada tipo.

Pareciera que fuese sencillo desplazar una trama de una LAN a otra, pero no lo es, ya que:

1. Cada LAN utiliza un formato de trama distinto. En consecuencia, cualquier copia que se realice entre LANs distintas requiere de volver a dar formato, lo que lleva tiempo de CPU, una nueva suma de verificación, y se presenta la posibilidad de errores sin detectar debido a bits erróneos en la memoria fuente.
2. Las LANs interconectadas no necesariamente operan a la misma tasa de datos, por lo que deben tenerse previstos búferes
3. Las longitudes de tramas varían de LAN a LAN
4. Seguridad. Tanto 802.11 como 802.16 soportan encriptación en la capa de enlace, pero Ethernet no. Por lo que los diversos servicios de encriptación de una red inalámbrica se pierden cuando el tráfico pasa por una Ethernet. Peor aún, si una estación inalámbrica encripta en la capa de enlace, será imposible desencriptarlo en una red Ethernet. Una solución a este problema es realizar encriptación en capas superiores, pero esto requiere que una estación 802.11 sepa con que tipo de estación se está comunicando, lo que viola la transparencia del protocolo.
5. Calidad de servicio. En Ethernet no existe el concepto de calidad de servicio, por lo que cualquier tráfico que pase por una Ethernet perderá su QoS.

Interconectividad Local

En algunas organizaciones, aún cuando todas sus LAN sean Ethernet, la interconexión puede dar muchos problemas. En un plano ideal, los puentes deberían ser transparentes (invisibles para todo el hardware y software). Sorprendentemente, esto es posible.

Al llegar una trama, un puente debe decidir si la descarta o la reenvía y, de ser lo segundo, en qué LAN debe poner la trama. Esta decisión se toma buscando la dirección de destino en una gran tabla hash contenida en el puente. La tabla puede listar cada posible destino e indicar a qué línea de salida (LAN) pertenece la trama.

Al conectarse inicialmente los puentes, todas las tablas hash están vacías. Ninguno de los puentes sabe la ubicación de ninguno de los destinos, por lo que usa el algoritmo de inundación: cada trama de entrada para un destino desconocido se envía a todas las LAN a las que está conectado el puente, excepto a aquella por la que llegó. A medida que pasa el tiempo, los puentes aprenden la ubicación de los destinos.

El algoritmo que los puentes transparentes utilizan es el de aprendizaje hacia atrás. Al examinar la dirección de origen, pueden saber cuál máquina está disponible en cuál LAN.

La topología puede cambiar a medida que las máquinas y los puentes se encienden, se apagan y se mueven. Para manejar topologías dinámicas, cada vez que se hace una entrada en la tabla de dispersión se anota la hora de llegada del marco en la entrada. Cada vez que llega un marco cuyo destino ya está en la tabla, se actualiza su entrada con la hora actual. Un proceso del puente analiza periódicamente la tabla hash y purga todas las entradas que tengan más de algunos minutos.

El proceso de enrutamiento para una trama que ingresa al puente, depende de la LAN de origen y de la LAN de destino, y es como sigue:

1. Si la LAN de destino y la LAN de origen son la misma, se descarta el marco.
2. Si la LAN de destino y la de origen son distintas, se reenvía el marco.
3. Si la LAN de destino es desconocida, se usa el proceso de inundación.

Puentes con árbol de expansión

Para incrementar la confiabilidad, algunos sitios utilizan 2 o más puentes en paralelo entre pares de LANs. Sin embargo este arreglo genera algunos problemas, como los ciclos.

Para evitar los ciclos, se debe armar un árbol de expansión, en el cual hay exactamente una trayectoria de cada LAN a cualquier otra LAN. Una vez que los puentes han acordado el árbol de expansión, todo el reenvío entre las LAN sigue al árbol de expansión. Dado que hay una trayectoria única de cada origen a cada destino, es imposible que ocurran ciclos.

Para construir el árbol de expansión, los puentes primero tienen que escoger un puente que sea la raíz del árbol; deben tomar esta decisión haciendo que cada uno difunda su número de serie, instalado por el fabricante y con garantía de ser único en el mundo. El puente con el número de serie menor se vuelve la raíz. A continuación, se construye un árbol de trayectorias mínimas de la raíz a cada puente y LAN. Este árbol es el árbol de expansión. Si falla un puente o una LAN, se calcula un árbol nuevo.

Puentes Remotos

Un uso común de los puentes es conectar 2 o más LANs distantes. Este objetivo se puede cumplir colocando un puente en cada LAN y conectando los puentes por pares con líneas punto a punto. La forma más sencilla de entender esto es considerar las 3 líneas punto a punto como LANs sin hosts.

Routers, hubs, switchs, puentes, repetidores y puertas de enlace

Todos estos dispositivos son de uso común, aunque difieren en formas sutiles y no tan sutiles. Para empezar, estos dispositivos operan en distintas capas. La capa es importante porque los distintos dispositivos utilizan diferentes partes de información para decidir su modo de operación. En un escenario común, el usuario genera algunos datos que se enviarán a una máquina remota. Estos datos se pasan a la capa de transporte, que le agrega un encabezado, y pasa la unidad a la capa de red, que agrega su propio encabezado, etc.

Aplicación	Puerta de enlace de aplicación
Transporte	Puerta de enlace de transporte
Red	Router
Enlace	Switch, puente
Física	Hub, repetidor

- Repetidor: Es un dispositivo analógico conectado a 2 segmentos de cable. Una señal que aparece en uno de ellos es amplificada y enviada al otro. No distinguen entre tramas, paquetes y encabezados. Sólo manejan voltios
- Hub: Tiene numerosos puertos de entrada que une de manera eléctrica. Las tramas que llegan a cualquiera de las líneas se envían a todas las demás. Constituye un solo dominio de colisión, por lo que si 2 tramas llegan al mismo tiempo, chocarán, al igual que en un cable coaxial. Todas las líneas que convergen deben operar a la misma velocidad. Por lo general no amplifican las señales entrantes
- Puente: Conecta 2 o más LAN. Extrae la dirección de destino del encabezado. En un puente, cada puerto constituye su propio dominio de colisión. Un puente puede tener tarjetas para conectar diferentes tipos de redes y velocidades
- Switch: Son similares a los puentes: ambos enrutan tomando como base las direcciones de las tramas. La principal diferencia consiste en que un switch se utiliza con mayor frecuencia para conectar computadoras individuales, por lo que debe tener muchos puertos. Cada puerto constituye su propio dominio de colisión, por lo que no existen las colisiones.
- Router: Utiliza el encabezado del paquete para elegir un puerto de salida
- Puertas de enlace de transporte: Conectan 2 computadoras que utilizan diferentes protocolos de transporte orientados a la conexión. La puerta de enlace de transporte puede copiar los paquetes de una conexión a otra y darles el formato que necesiten.
- Puertas de enlace de aplicación: Comprenden el formato y contenido de los datos y traducen los mensajes de un formato a otro. Ej: [mail a SMS](#)

VLANs (LAN virtuales)

El uso de hubs y switches con Ethernet hizo posible configurar las LANs con base en el aspecto lógico más que en el físico. Si una empresa necesita k LANs, compra k switches. Al elegir con cuidado qué conectores incorporar a que switches, los usuarios de una LAN se pueden seleccionar de tal manera que tenga sentido para la organización, sin tomar mucho en cuenta el aspecto geográfico.

¿Es importante quién está en qué LAN? A los administradores de red les gusta agrupar a los usuarios en LANs para reflejar la estructura de la organización, por temas de seguridad, carga y difusión (problemas de tormenta de difusión).

Para desacoplar la topología lógica de la física, surgieron las **VLAN**, que se fundamentan en los switches. El administrador de la red decide cuántas VLANs habrá, cuántas PC habrá en cada una y cómo se llamarán.

Para que las VLANs funcionen correctamente, las tablas de configuración se deben establecer en los puentes o switches. Estas tablas indican cuáles VLANs se pueden acceder a través de qué puertos (líneas).

Hemos supuesto que puentes y switches saben de alguna forma que “color” tienen las tramas que llegan. ¿Cómo lo saben?

1. A cada puerto se le asigna un color de VLAN
2. A cada dirección MAC se le asigna un color de VLAN
3. A cada protocolo de la capa 3 o a cada dirección IP se le asigna un color de VLAN

El único problema de este enfoque es que transgrede la regla más elemental de la conectividad: independencia de las capas.

Si hubiera alguna forma de identificar la VLAN en el encabezado de la trama, se desvanecería la necesidad de examinar la carga útil. Para esto se publicó el estándar 802.1Q: una etiqueta VLAN.

La clave de la solución consiste en comprender que los campos VLAN sólo son utilizados por los puentes y switches, no por las máquinas de los usuarios, por lo que las tarjetas de red no habría que desecharlas. Si el emisor no generará los campos VLAN, ¿quién lo hará?: el primer puente o switch con soporte VLAN en recibir una trama los agregará, y el último que los reciba los eliminará.

UNIDAD 5: La Capa de Red

La capa de red se encarga de llevar los paquetes desde el origen hasta el destino. Llegar al destino puede requerir muchos saltos por enrutadores intermedios. Esta función ciertamente contrasta con la de la capa de enlace de datos, que sólo tiene la meta de mover tramas de un extremo del cable al otro. Por lo tanto, la capa de red es la capa más baja que maneja la transmisión de extremo a extremo. Para lograr su cometido, la capa de red debe conocer la topología de la subred de comunicación y elegir las rutas adecuadas a través de ella.

ASPECTOS DE DISEÑO DE LA CAPA DE RED

Los servicios de la capa de red se han diseñado con los siguientes objetivos en mente.

1. Los servicios deben ser independientes de la tecnología del enrutador.
2. La capa de transporte debe estar aislada de la cantidad, tipo y topología de los enrutadores presentes.
3. Las direcciones de red disponibles para la capa de transporte deben seguir un plan de numeración uniforme, aun a través de varias LANs y WANs.

La discusión se centra en determinar si la capa de red debe proporcionar servicio orientado o no orientado a la conexión.

Un bando (representado por la comunidad de Internet) alega que la tarea del enrutador es mover bits de un lado a otro, y nada más. Los hosts deben aceptar este hecho y efectuar ellos mismos el control de errores y el control de flujo. Este punto de vista conduce directamente a la conclusión de que el servicio de red no debe ser orientado a la conexión, y debe contar tan sólo con las primitivas *SEND PACKET* y *RECEIVE PACKET*.

El otro bando (representado por las compañías telefónicas) argumenta que la subred debe proporcionar un servicio confiable, orientado a la conexión.

En un **servicio no orientado a la conexión**, los paquetes (datagramas) se colocan individualmente en la subred y se enrutan de manera independiente. No se necesita una configuración avanzada. En este contexto, por lo general los paquetes se conocen como datagramas y la subred se conoce como subred de datagramas.

En un **servicio orientado a la conexión**, antes de poder enviar cualquier paquete de datos, es necesario establecer una ruta del enrutador de origen al de destino. Esta conexión se conoce como CV (circuito virtual), y la subred se conoce como subred de circuitos virtuales. El propósito de los circuitos virtuales es evitar la necesidad de elegir una nueva ruta para cada paquete enviado. Cuando se establece una conexión, la ruta se almacena en tablas en de los enrutadores.

Comparación entre las subredes de circuitos virtuales y las de datagramas

Dentro de la subred hay varios pros y contras entre los circuitos virtuales y los datagramas. Uno de ellos tiene que ver con el espacio de memoria del enrutador y el ancho de banda. Los circuitos virtuales permiten que los paquetes contengan números de circuito en lugar de direcciones de destino completas.

Asunto	Subred de datagramas	Subred de circuitos virtuales
Configuración del circuito	No necesaria	Requerida
Análisis de Paquete	Se debe examinar la IP	Se debe examinar el nº de CV (más fácil)
Direccionamiento	Cada paquete contiene la dirección de origen y de destino	Cada paquete contiene un número de CV corto
Información de estado	Los enrutadores no contienen información de estado de las conexiones	Cada CV requiere mucho espacio de tabla del enrutador por conexión
Enrutamiento	Cada paquete se enruta de manera independiente	Ruta escogida cuando se establece el CV; todos los paquetes siguen esta ruta
Espacio de tablas en la memoria del router	Una entrada para cada destino posible	Una entrada por cada circuito virtual.
Efecto de fallas del enrutador	Ninguno, excepto para paquetes perdidos durante una caída	Terminan todos los CVs que pasan a través del enrutador
Calidad del servicio	Difícil	Fácil si se pueden asignar suficientes recursos por adelantado para cada CV
Control de congestión	Difícil	Fácil si pueden asignarse por adelantado suficientes recursos a cada CV
Tolerancia a fallas	Alta	Baja
Transacciones	Sencillas	El tiempo de establecimiento de un CV puede ser mucho mayor que la operación

ALGORITMOS DE ENRUTAMIENTO

El **algoritmo de enrutamiento** es aquella parte del software de la capa de red encargada de decidir la línea de salida por la que se transmitirá un paquete de entrada. Si la subred usa datagramas de manera interna, esta decisión debe tomarse cada vez que llega un paquete de datos. Si la subred usa circuitos virtuales internamente, las decisiones de enrutamiento se toman sólo al establecerse un circuito virtual nuevo. En lo sucesivo, los paquetes de datos simplemente siguen la ruta previamente establecida. Este último caso a veces se llama **enrutamiento de sesión**.

Algunas veces es útil distinguir entre el enrutamiento, que es el proceso consistente en tomar la decisión de cuáles rutas utilizar, y el reenvío, que es la acción que se toma al llegar un paquete.

Hay ciertas propiedades que todo algoritmo de enrutamiento debe poseer: exactitud, sencillez, robustez, estabilidad, equidad y optimización. El algoritmo de enrutamiento debe ser capaz de manejar los cambios de topología y tráfico sin requerir el aborto de todas las actividades en todos los *hosts* y el reinicio de la red con cada caída de un enrutador. Los algoritmos de enrutamiento pueden agruparse en dos clases principales:

- **Algoritmos no adaptativos (o Enrutamiento estático):** La decisión de qué ruta se usará se toma por adelantado, fuera de línea, y se carga en los enrutadores al arrancar la red. No basan sus decisiones de enrutamiento en mediciones o estimaciones del tráfico y la topología actuales.
- **Algoritmos adaptativos:** cambian sus decisiones de enrutamiento para reflejar los cambios de topología y, por lo general también el tráfico.

Principio de optimización

Si el enrutador **J** está en ruta óptima del enrutador **I** al enrutador **K**, entonces la ruta óptima de **J** a **K** también está en la misma ruta.

Como consecuencia directa del principio de optimización, podemos ver que el grupo de rutas óptimas de todos los orígenes a un destino dado forman un árbol con raíz en el destino. Tal árbol se conoce como **árbol sumidero** (o árbol divergente), donde la métrica de distancia es el número de saltos. Un árbol sumidero no necesariamente es único. La meta de todos los algoritmos de enrutamiento es descubrir y utilizar los árboles sumideros de todos los enrutadores. Este árbol no contiene ciclos, por lo que cada paquete será entregado en un número de saltos finito y limitado.

Enrutamiento por la ruta más corta (Estático)

Una manera de medir la longitud de una ruta es por la cantidad de saltos. Otra métrica es la distancia geográfica en kilómetros. Sin embargo, también son posibles muchas otras métricas además de los saltos y la distancia física, como el retardo medio de encolamiento. Cambiando la función de ponderación que asigna pesos a los arcos (enlaces), el algoritmo calcularía la ruta "más corta" de acuerdo con cualquiera de varios criterios, o una combinación de ellos. Ej: [Dijkstra](#)

Inundación (Estático)

Otro algoritmo estático es la **inundación**, en la que cada paquete de entrada se envía por cada una de las líneas de salida, excepto aquella por la que llegó. La inundación genera grandes cantidades de paquetes duplicados; de hecho, una cantidad infinita a menos que se tomen algunas medidas para limitar el proceso.

Una de estas medidas es integrar un contador de saltos en el encabezado de cada paquete, que disminuya con cada salto, y el paquete se descarte cuando el contador llegue a cero.

Una técnica alterna para ponerle diques a la inundación es llevar un registro de los paquetes difundidos, para evitar enviarlos una segunda vez. Esto se hace haciendo que el enrutador de origen ponga un número de secuencia en cada paquete que recibe de sus *hosts*. Para evitar que la lista crezca sin límites, cada lista debe incluir un contador, *k*, que indique que todos los números de secuencia hasta *k* ya han sido vistos.

Una variación de la inundación, un poco más práctica, es la **inundación selectiva**. En este algoritmo, los enrutadores no envían cada paquete de entrada por todas las líneas, sino sólo por aquellas que van aproximadamente en la dirección correcta. La inundación no es práctica en la mayoría de las aplicaciones.

Enrutamiento por vector de distancia (Dinámico)

Los algoritmos de **enrutamiento por vector de distancia** operan haciendo que cada enrutador mantenga una tabla (vector) que da la mejor distancia conocida a cada destino, y la línea que se puede usar para llegar ahí. Estas tablas se actualizan intercambiando información con los vecinos.

Una vez cada *T* mseg, cada enrutador envía a todos sus vecinos una lista de sus retardos estimados a cada destino. También recibe una lista parecida de cada vecino, y se actualiza la tabla.

El enrutamiento por vector de distancia funciona en teoría, pero tiene un problema serio en la práctica: aunque llega a la respuesta correcta, podría hacerlo lentamente. En particular, reacciona con rapidez a las buenas noticias, pero con lentitud ante las malas.

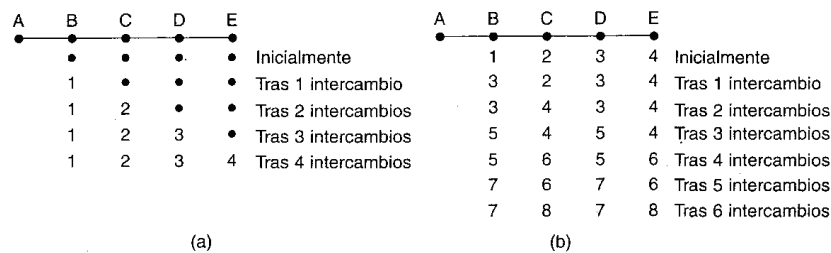


Figura 5-10. El problema de la cuenta hasta infinito.

En el primer intercambio de paquetes, *B* no escucha nada de *A*. Afortunadamente, *C* dice: "No te preocupes. Tengo una ruta a *A* de longitud 2". *B* no sabe que la ruta de *C* pasa a través de *B* mismo. Hasta donde *B* sabe, *C* puede tener 10 líneas, todas con rutas independientes a *A* de longitud 2. Como resultado, *B* ahora piensa que puede llegar a *A* por medio de *C*, con una longitud de ruta de 3. *D* y *E* no actualizan sus entradas para *A* en el primer intercambio.

En el segundo intercambio, *C* nota que cada uno de sus vecinos indica tener una ruta a *A* de longitud 3. *C* escoge una de ellas al azar y hace que su nueva distancia a *A* sea de 4, como se muestra en la tercera fila de la figura 5-10(b). Los intercambios subsecuentes producen la historia mostrada en el resto de la figura 5-10(b).

Gradualmente, todos los enrutadores elevan cuentas hacia el infinito. Por esta razón, es prudente hacer que el infinito sea igual a la ruta más larga, más 1.

El enrutamiento por vector de distancia tiene 2 problemas principales:

1. Debido a que la métrica de retardo era la longitud de la cola, no tomaba en cuenta, el ancho de banda al escoger rutas.
2. El algoritmo con frecuencia tardaba demasiado en converger

Enrutamiento por estado del enlace (Dinámico)

El concepto en que se basa el enrutamiento por estado del enlace es sencillo y puede enunciarse en cinco partes. Cada enrutador debe:

1. Descubrir a sus vecinos y conocer sus direcciones de red: Cuando un enrutador se pone en funcionamiento, envía un paquete HELLO especial a cada línea punto a punto. El enrutador del otro extremo regresa una respuesta indicando quién es. Estos nombres deben ser globalmente únicos
2. Medir el retardo para cada uno de sus vecinos: Se realiza enviando un paquete ECHO especial a través de la línea y una vez que llegue al otro extremo, éste debe regresarlo inmediatamente. Si se mide el tiempo de ida y vuelta y se divide entre dos, el enrutador emisor puede tener una idea razonable del retardo. La prueba puede llevarse a cabo varias veces y usarse el promedio. Un aspecto interesante es si se debe tomar en cuenta la carga al medir el retardo. Para considerar la carga, el temporizador debe iniciarse cuando el paquete ECHO se ponga en la cola. Para ignorar la carga, el temporizador debe iniciarse cuando el paquete ECHO alcance el frente de la cola.
3. Construir un paquete que indique todo lo que acaba de aprender: El paquete comienza con la identidad del emisor, seguida de un número de secuencia, una edad y una lista de vecinos. Se da el retardo al vecino. Es fácil construir los paquetes de estado del enlace, pero es difícil determinar cuándo. Una posibilidad es construirlos de manera periódica, a intervalos regulares. Otra posibilidad es construirlos cuando ocurra un evento significativo.

4. Enviar este paquete a todos los demás enrutadores: A medida que se distribuyen e instalan los paquetes, los enrutadores que reciban los primeros cambiarán sus rutas. En consecuencia, los distintos enrutadores podrían estar usando versiones diferentes de la topología, lo que puede conducir a inconsistencias, ciclos, máquinas inalcanzables y otros problemas.

La idea fundamental es utilizar inundación para distribuir los paquetes de estado del enlace. A fin de mantener controlada la inundación, cada paquete contiene un número de secuencia que se incrementa con cada paquete nuevo enviado. Los enrutadores llevan el registro de todos los pares (enrutador origen, secuencia) que ven. Cuando llega un paquete de estado del enlace, se verifica contra la lista de paquetes ya vistos. Si es nuevo, se reenvía a través de todas las líneas, excepto aquella por la que llegó. Si es un duplicado, se descarta. Si llega un paquete con número de secuencia menor que el mayor visto hasta el momento, se rechaza como obsoleto debido que el enrutador tiene datos más recientes.

Este algoritmo tiene algunos problemas:

- Si los números de secuencia vuelven a comenzar, reinará la confusión. La solución aquí es utilizar un número de secuencia de 32 bits. Con un paquete de estado del enlace por segundo, el tiempo para volver a empezar será de 137 años, por lo que puede ignorarse esta posibilidad.
- Si llega a caerse un enrutador, perderá el registro de su número de secuencia. Si comienza nuevamente en 0, se rechazará como duplicado el siguiente paquete.
- Si llega a corromperse un número de secuencia y se escribe 65,540 en lugar de 4 (un error de 1 bit), los paquetes 5 a 65,540 serán rechazados como obsoletos.

La solución a todos estos problemas es incluir la edad de cada paquete después del número de secuencia y disminuirla una vez cada segundo. Cuando la edad llega a cero, se descarta la información de ese enrutador.

Algunos refinamientos de este algoritmo lo hacen más robusto. Una vez que un paquete de estado del enlace llega a un enrutador para ser inundado, no se encola para transmisión inmediata. En vez de ello, entra en un área de almacenamiento donde espera un tiempo breve. Si antes de transmitirlo entra otro paquete de estado del enlace proveniente del mismo origen, se comparan sus números de secuencia. Si son iguales, se descarta el duplicado. Si son diferentes, se desecha el más viejo.

5. Calcular la ruta más corta a todos los demás enrutadores: Una vez que un enrutador ha acumulado un grupo completo de paquetes de estado del enlace, puede construir el grafo de la subred completa porque todos los enlaces están representados. De hecho, cada enlace se representa dos veces, una para cada dirección. Los dos valores pueden promediarse o usarse por separado. Ahora puede ejecutar localmente el algoritmo de Dijkstra para construir la ruta más corta a todos los destinos posibles.

Enrutamiento jerárquico

A medida que crece el tamaño de las redes, también lo hacen, de manera proporcional, las tablas de enrutamiento del enrutador. Las tablas que siempre crecen consumen memoria del enrutador, y requieren más tiempo de CPU para examinarlas y más ancho de banda para enviar informes de estado entre enrutadores. En cierto momento, la red puede crecer hasta el punto en que ya no es factible que cada enrutador tenga una entrada para cada uno de los demás enrutadores, por lo que el enrutamiento tendrá que hacerse de manera jerárquica, como ocurre en la red telefónica.

Cuando se utiliza el enrutamiento jerárquico, los enrutadores se dividen en lo que llamaremos **regiones**, donde cada enrutador conoce todos los detalles para enrutar paquetes a destinos dentro de su propia región, pero no sabe nada de la estructura interna de las otras regiones. A medida que crece la razón entre la cantidad de regiones y el número de enrutadores por región, aumentan los ahorros de espacio de tabla. Desgraciadamente, estas ganancias de espacio no son gratuitas. Se paga un precio, que es una longitud de ruta mayor.

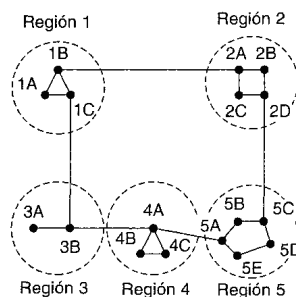


Tabla jerárquica para 1A

Dest.	Línea	Salto
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Enrutamiento por difusión

A veces, los *hosts* necesitan enviar mensajes a varios otros *hosts* o a todos los demás. El envío simultáneo de un paquete a todos los destinos se llama **difusión**. Hay varios métodos:

1. Que el origen simplemente envíe un paquete distinto a todos los destinos. Desperdicia ancho de banda, y requiere que el origen tenga una lista completa de todos los destinos.
2. Inundación
3. Enrutamiento multidestino: Cada paquete contiene una lista de destinos o un mapa de bits que indica los destinos deseados. Cuando un paquete llega al enrutador, éste revisa todos los destinos para determinar el grupo de líneas de salida que necesitará. El enrutador genera una copia nueva del paquete para cada línea de salida que se utilizará, e incluye en cada paquete sólo aquellos destinos que utilizarán la línea.
4. Árbol de expansión: es un subgrupo de la subred que incluye todos los enrutadores pero no contiene ciclos. El único problema es que cada enrutador debe tener conocimiento de algún árbol de expansión para que este método pueda funcionar.
5. Reenvío por ruta invertida: Cuando llega un paquete difundido a un enrutador, éste lo revisa para ver si llegó por la línea normalmente usada para enviar paquetes al origen de la difusión. De ser así, hay excelentes posibilidades de que el paquete difundido haya seguido la mejor ruta desde el enrutador y, por lo tanto, sea la primera copia en llegar al enrutador. Si éste es el caso, el enrutador reenvía copias del paquete a todas las líneas, excepto a aquella por la que llegó. Sin embargo, si el paquete difundido llegó por una línea diferente de la preferida, el paquete se descarta como probable duplicado. En otras palabras, este algoritmo se usa cuando no se conoce el árbol de expansión, y queremos construir una aproximación sin mucho costo. Se asume que cuando un router envía un paquete al nodo N, siempre elige un enlace que pertenece al árbol sumidero con raíz N.

Este método es razonablemente eficiente y fácil de implementar. No requiere que los enrutadores conozcan los árboles de expansión ni tiene la sobrecarga de una lista de destinos en cada paquete de difusión, como los tiene el direccionamiento multidestino. Tampoco requiere mecanismos especiales para detener el proceso, como en el caso de la inundación.

Enrutamiento por multidifusión

Algunas aplicaciones requieren que procesos muy separados trabajen juntos en grupo. Si el grupo es pequeño, simplemente se puede transmitir a cada uno de los miembros un mensaje punto a punto. Si el grupo es grande, esta estrategia es costosa. Necesitamos una manera de enviar

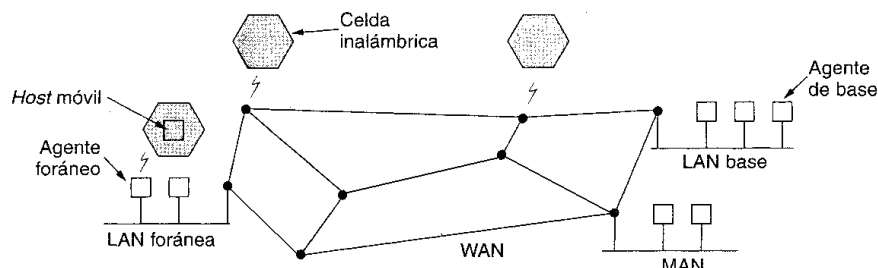
mensajes a grupos bien definidos de tamaño numéricamente grande, pero pequeños en comparación con la totalidad de la red.

Para la multidifusión se requiere administración de grupos. Se necesita alguna manera de crear y destruir grupos, y un mecanismo para que los procesos se unan a los grupos y salgan de ellos. La forma de realizar estas tareas no le concierne al algoritmo de enrutamiento. Lo que sí le concierne es que cuando un proceso se una a un grupo, informe a su *host* este hecho. De cualquier manera, los enrutadores aprenden qué *hosts* pertenecen a cuáles grupos.

Cada enrutador calcula un árbol de expansión que cubre a todos los demás enrutadores de la subred. Cuando un proceso envía un paquete de multidifusión a un grupo, el primer enrutador examina su árbol de expansión y lo recorta, eliminando todas las líneas que conduzcan a *hosts* que no sean miembros del grupo.

Enrutamiento para hosts móviles y routers fijos

El problema radica en cómo encuentra la red al *host* móvil.



Se dice que los *hosts* que nunca se mueven son estacionarios. Los **hosts migratorios** básicamente son *hosts* estacionarios que se mueven de un lugar fijo a otro de tiempo en tiempo, pero que usan la red sólo cuando están conectados físicamente a ella. Los **hosts ambulantes** hacen su cómputo en movimiento, y necesitan mantener sus conexiones mientras se trasladan de un lado a otro. Usaremos el término **hosts móviles** para referirnos a cualquiera de las dos últimas categorías. Se supone que todos los *hosts* tienen una **localidad base** que nunca cambia. Los *hosts* también tienen una dirección base permanente.

El mundo se divide (geográficamente) en unidades pequeñas, a las que llamaremos áreas. Un área por lo general es una LAN o una celda inalámbrica. Cada área tiene uno o más **agentes foráneos**, los cuales son procesos que llevan el registro de todos los *hosts* móviles que visitan el área. Además, cada área tiene un **agente de base**, que lleva el registro de todos los *hosts* cuya base está en el área, pero que actualmente están visitando otra área.

Cuando un nuevo *host* entra en un área, ya sea al conectarse a ella, o simplemente al entrar en la celda, su computadora debe registrarse con el agente foráneo de ese lugar. El procedimiento de registro funciona típicamente de esta manera:

1. Periódicamente, cada agente foráneo difunde un paquete que anuncia su existencia y dirección. Un *host* móvil recién llegado puede esperar uno de estos mensajes, pero si no llega ninguno con suficiente rapidez, el *host* móvil puede difundir un paquete que diga: "¿hay agentes foráneos por ahí?"
2. El *host* móvil se registra con el agente foráneo, dando su dirección base, su dirección actual de capa de enlace de datos y cierta información de seguridad.
3. El agente foráneo se pone en contacto con el agente de base del *host* móvil y le dice: "uno de tus *hosts* está por aquí". El mensaje del agente foráneo al agente de base contiene la

dirección de red del agente foráneo, así como la información de seguridad, para convencer al agente de base de que el *host* móvil en realidad está ahí.

4. El agente de base examina la información de seguridad, que contiene una marca de tiempo, para comprobar que fue generada en los últimos segundos. Si está conforme, indica al agente foráneo que proceda.
5. Cuando el agente foráneo recibe la confirmación de recepción del agente de base, hace una entrada en sus tablas e informa al *host* móvil que ahora está registrado.

Idealmente, cuando un *host* sale de un área, este hecho también se debe anunciar para permitir que se borre el registro, pero muchos usuarios apagan abruptamente sus computadoras cuando terminan.

Cuando un paquete se envía a un *host* móvil, se enruta a la LAN base del *host*. Los paquetes enviados al *host* móvil en su LAN base son interceptados por el agente de base que se encuentra ahí. A continuación, dicho agente busca la nueva ubicación (temporal) del *host* móvil y encuentra la dirección del agente foráneo que maneja al *host* móvil.

El agente de base entonces hace dos cosas. Primero, encapsula el paquete en el campo de carga útil de un paquete exterior y envía este último al agente foráneo. Tras obtener el paquete encapsulado, el agente foráneo extrae el paquete original del campo de carga útil y lo envía al *host* móvil como trama de enlace de datos.

Segundo, el agente de base indica al emisor que en lo futuro envíe paquetes al *host* móvil encapsulándolos en la carga útil de paquetes explícitamente dirigidos al agente foráneo, en lugar de simplemente enviarlos a la dirección base del *host* móvil. Los paquetes subsiguientes ahora pueden enrutarse en forma directa al usuario por medio del agente foráneo, omitiendo la localidad base por completo

Enrutamiento en redes ad hoc (Hosts y routers móviles)

Entre las posibilidades se encuentran: Vehículos militares en un campo de batalla sin infraestructura, una flota de barcos en el mar, una reunión de personas con computadoras portátiles en un área que no cuenta con 802.11, etc.

En todos estos casos y en otros, cada nodo consiste en un enrutador y un *host*, por lo general en la misma computadora. Las redes de nodos que están cerca entre sí se conocen como **redes ad hoc** o **MANETs (Redes ad hoc Móviles)**. En estas redes, los enrutadores pueden ir y venir o aparecer en nuevos lugares en cualquier momento, la topología cambia todo el tiempo, por lo que la necesidad o la validez de las rutas pueden cambiar en cualquier momento, sin previo aviso.

Un algoritmo de enrutamiento para las redes *ad hoc* es el **AODV (Vector de Distancia ad hoc bajo Demanda)**, que describiremos a continuación:

- 1- Descubrimiento de ruta: En cualquier instante dado, una red *ad hoc* puede describirse mediante un grafo de los nodos (enrutadores + *hosts*). Dos nodos se conectan si se pueden comunicar de manera directa mediante sus radios. Debido a que uno de los dos podría tener un emisor más poderoso que el otro, es posible que *A* esté conectado a *B*, pero *B* no está conectado a *A*. Por simplicidad, asumiremos que todas las conexiones son simétricas.

El algoritmo AODV mantiene una tabla en cada nodo, codificada por destino, que proporciona información acerca de ese destino, incluyendo a cuál vecino enviar los paquetes a fin de llegar al destino. Suponga que *A* busca en sus tablas y no encuentra una

entrada para L . Ahora tiene que descubrir una ruta a L . Debido a esta propiedad de descubrir rutas sólo cuando es necesario este algoritmo se conoce como "bajo demanda".

Para localizar a L , A construye un paquete especial de solicitud de ruta (ROUTE REQUEST) y lo difunde. Este paquete contiene los siguientes campos:

Dirección Origen	ID de Solicitud	Dirección Destino	#de secuencia De Origen	#de secuencia De Destino	Cuenta de Saltos
---------------------	--------------------	----------------------	----------------------------	-----------------------------	---------------------

Cuando un paquete de solicitud de ruta llega a un nodo intermedio (B por ejemplo), se procesa mediante los siguientes pasos.

1. El par (*Dirección de origen*, *ID de solicitud*) se busca en una tabla de historia local para ver si esta solicitud ya se había visto y procesado. Si es un duplicado, se descarta y el procesamiento se detiene. Si no es un duplicado, el par se introduce en la tabla de historia a fin de que se puedan rechazar futuros duplicados, y el procesamiento continúa.
2. El receptor busca el destino en su tabla de enrutamiento. Si se conoce una ruta reciente al destino, se regresa un paquete de respuesta de ruta (ROUTE REPLY) al origen que le indica cómo llegar al destino (básicamente: utilízame). Reciente significa que el *Número de secuencia de destino* almacenado en la tabla de enrutamiento es mayor que o igual al *Número de secuencia de destino* del paquete de solicitud de ruta. Si es menor, la ruta almacenada es más antigua que la que el origen tenía para el destino, por lo que se ejecuta el paso 3.
3. Puesto que el receptor no conoce una ruta reciente al destino, incrementa el campo *Cuenta de saltos* y vuelve a difundir el paquete de solicitud de ruta. También extrae los datos del paquete y los almacena como una entrada nueva en su tabla de rutas invertidas. Esta información se utilizará para construir la ruta invertida a fin de que la respuesta pueda regresar posteriormente al origen. También se inicia un temporizador para la nueva entrada de ruta invertida. Si expira, la entrada se borra.

El paquete se inspecciona en cada nodo intermedio del camino de regreso. Se introduce en la tabla de enrutamiento local como una ruta a L si se cumple una o más de las siguientes tres condiciones:

1. No se conoce una ruta a L .
 2. El número de secuencia para L en el paquete de respuesta de ruta es mayor que el valor en la tabla de enrutamiento.
 3. Los números de secuencia son iguales pero la nueva ruta es más corta.
- 2- Mantenimiento de rutas:** Cada nodo difunde de manera periódica un mensaje de saludo (*HELLO*). Se espera que cada uno de sus vecinos responda a dicho mensaje. Si no se recibe ninguna respuesta, el difusor sabe que el vecino se ha movido del alcance y ya no está conectado a él. Esta información se utiliza para eliminar rutas que ya no funcionan.

Cuando cualquiera de los vecinos de N se vuelve inalcanzable, N verifica su tabla de enrutamiento para ver cuáles destinos tienen rutas en las que se incluya al vecino ahora perdido. Para cada una de estas rutas, se les informa a los vecinos activos que su ruta a través de N ahora es inválida y que se debe eliminar de sus tablas de enrutamiento. A continuación los vecinos activos indican este hecho a sus vecinos activos, y así sucesivamente y de manera recursiva, hasta que las rutas que dependen del nodo perdido se eliminan de todas las tablas de enrutamiento.

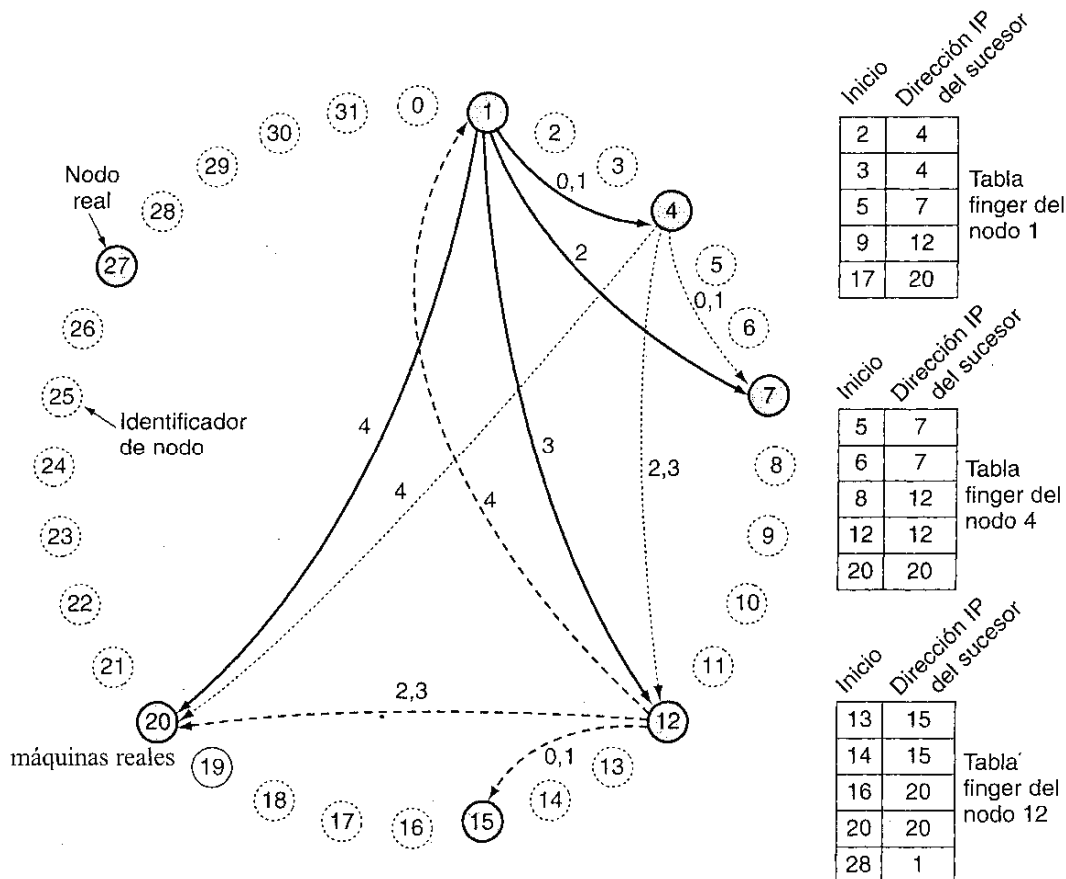
Búsqueda de nodos en redes de igual a igual

Lo que hace que los sistemas de igual a igual sean interesantes es que son totalmente distribuidos. El problema se reduce a qué debe hacer el usuario para encontrar un nodo que contiene lo que desea cuando no hay una base de datos centralizada o incluso un índice centralizado.

El sistema **Chord** consiste de n usuarios participantes, cada uno de los cuales podría contar con algunos registros almacenados y además está preparado para almacenar bits y fragmentos del índice para que otros usuarios los utilicen. Cada nodo de usuario tiene una dirección IP que puede generar un código de *hash* de m bits mediante una función de *hash*. Podemos convertir cualquier dirección IP a un número de 160 bits llamado **identificador de nodo**.

Definiremos la función $\text{sucesor}(k)$ como el identificador de nodo del primer nodo real que sigue a k alrededor del círculo en el sentido de las manecillas del reloj. Por ejemplo, $\text{sucesor}(6) = 7$, $\text{sucesor}(8) = 12$ y $\text{sucesor}(22) = 27$.

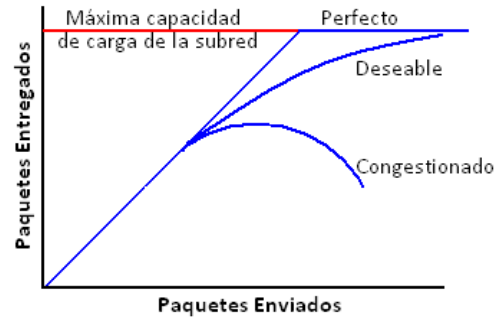
A los nombres de los registros (nombres de canciones, nombres de los predecesores, etcétera) también se les aplica la función de *hash* para generar un número de 160 bits, llamado **clave**. $\text{clave} = \text{hash}(\text{nombre})$. Si una persona que posee un registro genealógico para *nombre* desea ponerlo a disposición de todos, primero construye una tupla que consiste de $(\text{nombre}, \text{mi-dirección-IP})$ y después solicita a $\text{sucesor}(\text{hash}(\text{nombre}))$ que almacene la tupla.



Si posteriormente algún usuario desea buscar *nombre*, le aplica la función de *hash* para obtener *clave* y después utiliza $\text{sucesor}(\text{clave})$ para encontrar la dirección IP del nodo que almacena sus tuplas de índice.

ALGORITMOS DE CONTROL DE CONGESTION

Cuando hay demasiados paquetes presentes en la subred (o en una parte de ella), hay una degradación del desempeño. Esta situación se llama **congestión**. A medida que aumenta el tráfico, los enrutadores ya no pueden manejarlo y comienzan a perder paquetes. Esto tiende a empeorar las cosas. Con mucho tráfico, el desempeño se desploma por completo y casi no hay entrega de paquetes.



La congestión puede ocurrir por varias razones.

- Si llegan cadenas de paquetes y todas necesitan la misma línea de salida, habrá una cola.
- Si no hay suficiente memoria para almacenar a todos los paquetes
- Expiraciones de temporizadores cuando se han enviado duplicados
- Procesadores de los enrutadores lentos

El **control de congestión** se ocupa de asegurar que la subred sea capaz de transportar el tráfico ofrecido. Es un asunto global, en el que interviene el comportamiento de todos los *hosts*, todos los enrutadores, el proceso de almacenamiento y reenvío dentro de los enrutadores, y todos los demás factores que tienden a disminuir la capacidad de transporte de la subred.

En contraste, el **control de flujo** se relaciona con el tráfico punto a punto entre un emisor dado y un receptor dado. Su tarea es asegurar que un emisor rápido no pueda transmitir datos de manera continua a una velocidad mayor que la que puede absorber el receptor. El control de flujo casi siempre implica una retroalimentación directa del receptor al emisor.

Principios generales del control de congestión

Muchos problemas de los sistemas complejos, como las redes de computadoras, pueden analizarse desde el punto de vista de una teoría de control. Este método conduce a dividir en dos grupos todas las soluciones: de ciclo abierto y de ciclo cerrado. En esencia, las **soluciones de ciclo abierto** intentan resolver el problema mediante un buen diseño, para asegurarse en primer lugar de que no ocurra. Una vez que el sistema está en funcionamiento, no se hacen correcciones a medio camino.

Las **soluciones de ciclo cerrado** se basan en el concepto de un ciclo de retroalimentación. Este método tiene tres partes cuando se aplica al control de congestión:

1. Monitorear el sistema para detectar cuándo y dónde ocurren congestiones: se pueden usar varias métricas. Las principales son el porcentaje de paquetes descartados debido a falta de espacio de búfer, la longitud promedio de las colas, la cantidad de paquetes para los cuales termina el temporizador y se transmiten de nueva cuenta, el retardo promedio de los paquetes y la desviación estándar del retardo de paquete. En todos los casos, un aumento en las cifras indica un aumento en la congestión.
2. Pasar esta información a lugares en los que pueden llevarse a cabo acciones: es la transferencia de información relativa a la congestión desde el punto en que se detecta hasta el punto en que puede hacerse algo al respecto
3. Ajustar la operación del sistema para corregir el problema

En todos los esquemas de retroalimentación, la esperanza es que el conocimiento sobre la congestión hará que los *hosts* emprendan acciones adecuadas con miras a reducir la congestión.

Se conocen muchos algoritmos de control de congestión. Se clasifican en:

1. De ciclo abierto: Pueden actuar en el origen o en el destino
2. De ciclo cerrado:
 - a. *De retroalimentación explícita*: regresan paquetes desde el punto de congestión para avisar al origen
 - b. *De retroalimentación implícita*: el origen deduce la existencia de una congestión haciendo observaciones locales, como el tiempo necesario para que regresen las confirmaciones de recepción

Existen varias maneras de reducir la carga, como negar el servicio a algunos usuarios, degradar el servicio para algunos o todos los usuarios y obligar a los usuarios a programar sus solicitudes de una manera más predecible.

Políticas de prevención de congestión

Los sistemas de ciclo abierto están diseñados para reducir al mínimo la congestión desde el inicio, en lugar de permitir que ocurra y reaccionar después del hecho.

Capa	Políticas
Transporte	<ul style="list-style-type: none">- Política de retransmisión- Política de almacenamiento en caché de paquetes fuera de orden- Política de confirmaciones de recepción- Política de control de flujo- Determinación de terminaciones de temporizador
Red	<ul style="list-style-type: none">- Circuitos virtuales vs datagramas en la subred- Política de encolamiento y servicio de paquetes- Política de descarte de paquetes- Algoritmo de enrutamiento- Administración de tiempo de vida del paquete
Enlace de datos	<ul style="list-style-type: none">- Política de retransmisiones- Política de almacenamiento en caché de paquetes fuera de orden- Política de confirmación de recepción- Política de control de flujo

Control de congestión en subredes de circuitos virtuales (ciclo cerrado)

Un método de control dinámico de congestión en las subredes de circuitos virtuales, es el **control de admisión**. La idea es sencilla: una vez que se ha detectado la congestión, no se establecen circuitos virtuales nuevos hasta que ha desaparecido el problema. Por lo tanto, fallan los intentos por establecer conexiones nuevas de capa de transporte.

Un método alternativo es permitir el establecimiento de nuevos circuitos virtuales, pero enrutando cuidadosamente los circuitos nuevos por otras rutas que no tengan problemas.

Otra estrategia que tiene que ver con los circuitos virtuales es negociar un acuerdo entre el *host* y la subred cuando se establece un circuito virtual. De esta manera, es poco probable que ocurran congestiones en los circuitos virtuales nuevos, porque está garantizada la disponibilidad de todos los recursos necesarios.

Control de congestión en subredes de datagramas

Cada enrutador puede asociar cada línea a una variable real, u , cuyo valor, entre 0.0 y 1.0, refleja el uso reciente de esa línea. Siempre que u rebasa el umbral, la línea de salida entra en un estado de "advertencia". Cada paquete nuevo que llega se revisa para ver si su línea de salida está en el estado de advertencia. Si es así, se realiza alguna acción. Ésta puede ser una de varias alternativas:

- a. El bit de advertencia: La arquitectura DECNET antigua (y Frame Relay) señalaba el estado de advertencia activando un bit especial en el encabezado del paquete. Cuando el paquete llegaba a su destino, la entidad transportadora copiaba el bit en la siguiente confirmación de recepción que se regresaba al origen. A continuación el origen reducía el tráfico.
- b. Paquetes reguladores: el enrutador regresa un **paquete regulador** al *host* de origen, proporcionándole el destino encontrado en el paquete. Cuando el *host* de origen obtiene el paquete regulador, se le pide que reduzca en un porcentaje X el tráfico enviado al destino especificado. Si no llega ningún paquete de este tipo durante el periodo de escucha, el *host* puede incrementar el flujo otra vez. A altas velocidades o distancias grandes, la reacción del *host* origen por un paquete regulador es muy lenta.
- c. Paquetes reguladores de salto por salto: Un método alternativo es hacer que el paquete regulador ejerza su efecto en cada salto que dé, hasta llegar al origen. Cuando llegue al origen, el flujo en todo el camino estará reducido.

Desprendimiento de carga

Es una manera de decir que, cuando se inunda a los enrutadores con paquetes que no pueden manejar, simplemente los tiran. El paquete a descartar puede depender de las aplicaciones que se estén ejecutando. En la transferencia de archivos vale más un paquete viejo que uno nuevo. En contraste, en multimedia es más importante un paquete nuevo que uno viejo.

Todas estas decisiones requieren cooperación de los emisores para evitar inconvenientes. Para poner en práctica una política inteligente de descarte, las aplicaciones deben marcar sus paquetes con clases de prioridades para indicar su importancia.

Otra posibilidad es descartar paquetes antes de que se ocupe todo el espacio de búfer y sea demasiado tarde. Un algoritmo popular para realizar esto se conoce como **RED (detección temprana aleatoria)**. En algunos protocolos de transporte (entre ellos TCP), la respuesta a paquetes perdidos es que el origen disminuya su velocidad.

¿Cómo puede el enrutador informar al origen sobre el problema? Una estrategia (aparte de las ya mencionadas) es descartar el paquete seleccionado y no reportarlo. Esta forma implícita de retroalimentación sólo funciona cuando los orígenes responden a la pérdida de paquetes reduciendo su tasa de transmisión. En las redes inalámbricas, en las que la mayoría de las pérdidas se debe al ruido en el enlace de radio, no se puede utilizar este método.

Control de fluctuación

La variación en el retardo de los paquetes se conoce como **fluctuación**. La fluctuación puede limitarse calculando el tiempo de tránsito esperado para cada salto en la ruta. Cuando un paquete llega a un enrutador, éste lo examina para saber qué tan adelantado o retrasado está respecto a lo programado. Esta información se almacena en el paquete y se actualiza en cada salto. Si el paquete está adelantado, se retiene durante el tiempo suficiente para regresar lo a lo programado; si está retrasado, el enrutador trata de sacarlo rápidamente.

CALIDAD DEL SERVICIO (QoS)

Un **flujo** es un conjunto de paquetes que van de un origen a un destino. En una red orientada a la conexión, todos los paquetes que pertenezcan a un flujo siguen la misma ruta; en una red sin conexión, pueden seguir diferentes rutas. La necesidad de cada flujo se puede caracterizar por cuatro parámetros principales: confiabilidad, retardo, fluctuación y ancho de banda. Estos parámetros en conjunto determinan la **QoS (calidad del servicio)** que el flujo requiere.

Aplicación	Confiabilidad	Retardo	Fluctuación	Ancho de banda
Correo electrónico	Alta	Bajo	Baja	Bajo
Transferencia de archivos	Alta	Bajo	Baja	Medio
Acceso a Web	Alta	Medio	Baja	Medio
Inicio de sesión remoto	Alta	Medio	Media	Bajo
Audio bajo demanda	Baja	Bajo	Alta	Medio
Vídeo bajo demanda	Baja	Bajo	Alta	Alto
Telefonía	Baja	Alto	Alta	Bajo
Videoconferencia	Baja	Alto	Alta	Alto

Las redes ATM clasifican los flujos en cuatro categorías amplias con respecto a sus demandas de QoS, como se muestra a continuación:

1. Tasa de bits constante (por ejemplo, telefonía).
2. Tasa de bits variable en tiempo real (por ejemplo, videoconferencia comprimida).
3. Tasa de bits variable no constante (por ejemplo, ver una película a través de Internet).
4. Tasa de bits disponible (por ejemplo, transferencia de archivos).

Técnicas para alcanzar buena calidad de servicio

1. Sobre aprovisionamiento: Es proporcionar la suficiente capacidad de enrutador, espacio en búfer y ancho de banda como para que los paquetes fluyan con facilidad. Es costoso.
2. Almacenamiento en búfer: No afecta la confiabilidad o el ancho de banda, incrementa el retardo, y atenúa la fluctuación. Es útil para el vídeo o audio bajo demanda.
3. Modelado de tráfico: Modera el tráfico en el servidor, en lugar de en el cliente. Cuando se establece una conexión, el usuario y la subred acuerdan cierto patrón de tráfico para ese circuito. Algunas veces esto se llama **acuerdo de nivel de servicio**. Luego, el servidor debe realizar una **supervisión de tráfico** para ver si el acuerdo se está cumpliendo. La supervisión es más fácil en las subredes de circuitos virtuales que en las de datagramas.
4. Algoritmo de cubeta con goteo: Imagínese una cubeta con un pequeño agujero en el fondo. Sin importar la rapidez con que entra agua en la cubeta, el flujo de salida tiene una tasa constante, p , cuando hay agua en la cubeta, y una tasa de 0 cuando la cubeta está vacía. Además, una vez que se llena la cubeta, cualquier agua adicional que entra se derrama por los costados y se pierde (no aparece en el flujo por debajo del agujero).
5. Algoritmo de cubeta con tokens: El algoritmo de cubeta con goteo impone un patrón de salida rígido a la tasa promedio, sin importar la cantidad de ráfagas que tenga el tráfico. En muchas aplicaciones es mejor permitir que la salida se acelere un poco cuando llegan ráfagas grandes, por lo que se necesita un algoritmo más flexible, de preferencia uno que nunca pierda datos. El **algoritmo de cubeta con tokens** es uno de tales algoritmos. Este último no permite que los *hosts* inactivos acumulen permisos para enviar posteriormente

ráfagas grandes, pero sí permite el ahorro, hasta el tamaño máximo de la cubeta, n . Es decir, pueden enviarse a la vez ráfagas de hasta n paquetes, permitiendo cierta irregularidad en el flujo de salida y dando una respuesta más rápida a las ráfagas de entrada repentinas. Otra diferencia entre los dos algoritmos es que el algoritmo de cubeta con *tokens* descarta los *tokens* (es decir, la capacidad de transmisión) cuando se llena la cubeta, pero nunca descarta los paquetes. En contraste, el algoritmo de cubeta con goteo descarta los paquetes cuando se llena la cubeta.

6. Reservación de recursos: Una vez que se tiene una ruta específica para una flujo, es posible reservar recursos a lo largo de esa ruta para asegurar que la capacidad necesaria esté disponible. Se pueden reservar tres tipos de recursos: *ancho de banda*, *espacio en búfer* y *ciclos de CPU del enrutador*.
7. Control de admisión: Cuando un flujo de este tipo se ofrece a un enrutador, éste tiene que decidir, con base en su capacidad y en cuántos compromisos tiene con otros flujos, si lo admite o lo rechaza. Debido a que muchas partes pueden estar involucradas en la negociación del flujo (el emisor, el receptor y todos los enrutadores a lo largo de la ruta), los flujos deben describirse de manera precisa en términos de parámetros específicos que se puedan negociar. Un conjunto de tales parámetros se conoce como **especificación de flujo**. Por lo general, el emisor produce una especificación de flujo que propone los parámetros que le gustaría utilizar. Conforme la especificación se propague por la ruta, cada enrutador la examina y modifica los parámetros conforme sea necesario. Las modificaciones sólo pueden reducir el flujo, no incrementarlo. Cuando llega al otro extremo, se pueden establecer los parámetros.
8. Enrutamiento proporcional: Es dividir el tráfico para cada destino a través de diferentes rutas.
9. Calendarización de paquetes: Un algoritmo es el de **encolamiento justo**, en el que los enrutadores tienen colas separadas para cada línea de salida, una por flujo.

Servicios integrados (algoritmos basados en flujo)

Entre 1995 y 1997, se diseñó una arquitectura para la multimedia de flujos continuos. El nombre genérico para este trabajo es **algoritmos basados en flujo** o **servicios integrados**. Se diseñó tanto para aplicaciones de unidifusión como para multidifusión. Nos concentraremos en la multidifusión, debido a que la transmisión por unidifusión es un caso especial de multidifusión.

El principal protocolo IETF para la arquitectura de servicios integrados es **RSVP (Protocolo de reservación de recursos)**, que permite que varios emisores transmitan a múltiples grupos de receptores, permite que receptores individuales cambien de canal libremente, optimiza el uso de ancho de banda y elimina la congestión.

En su forma más sencilla, el protocolo usa enrutamiento de multidifusión con árboles de expansión, como se vio antes. A cada grupo se le asigna un grupo de direcciones. Para enviar a un grupo, un emisor pone la dirección del grupo en sus paquetes. El algoritmo estándar de multidifusión construye entonces un árbol de expansión que cubre a todos los miembros del grupo. La única diferencia con la multidifusión normal es un poco de información extra multidifundida al grupo periódicamente para indicarle a los enrutadores a lo largo del árbol que mantengan ciertas estructuras de datos en sus memorias.

En cada salto, el enrutador nota la reservación y aparta el ancho de banda necesario; si no hay suficiente ancho de banda disponible, informa de una falla. En el momento que el mensaje llega

de regreso al origen, se ha reservado el ancho de banda desde el emisor hasta el receptor que hace la solicitud de reservación a lo largo del árbol de expansión.

La razón de esta estrategia en el caso totalmente dinámico es que el ancho de banda reservado está desacoplado de la selección del origen. Una vez que un receptor ha reservado ancho de banda, puede conmutarse a otro origen y conservar la parte de la ruta existente que es válida para el nuevo origen. Por ejemplo, si el *host* 2 está transmitiendo varios flujos de vídeo, el *host* 3 puede conmutarse entre ellos a voluntad sin cambiar su reservación: a los enrutadores no les importa el programa que está viendo el receptor.

Los algoritmos basados en flujo tienen el potencial de ofrecer buena calidad de servicio a uno o más flujos debido a que reservan los recursos que son necesarios a lo largo de la ruta. Sin embargo, también tienen una desventaja. Requieren una configuración avanzada para establecer cada flujo, algo que no se escala bien cuando hay miles o millones de flujos. Además, son vulnerables a las caídas de enrutadores, Por último, involucran intercambios complejos de enrutador a enrutador para establecer los flujos. Por esto, hay pocas implementaciones de RSVP.

Servicios diferenciados (algoritmos basados en clases)

La IETF ha diseñado un método que puede implementarse ampliamente de manera local en cada enrutador sin una configuración avanzada y sin que toda la ruta esté involucrada. Este método se conoce como calidad de servicio **basada en clase** (contraria a basada en flujo). La IETF ha estandarizado una arquitectura para él, llamada **servicios diferenciados**.

Un conjunto de enrutadores que forman un dominio administrativo (Ej: un ISP) pueden ofrecer los servicios diferenciados (SD). La administración define un conjunto de clases de servicios con reglas de reenvío correspondientes. Si un cliente firma para un SD, los paquetes del cliente que entran en el dominio podrían contener un campo *Tipo de servicio*, con un mejor servicio proporcionado a algunas clases (Ej: *servicio Premium*) que a otras. Al tráfico dentro de una clase se le podría requerir que se apegue a algún modelo específico. Esto hace a SD muy fácil de implementar.

El servicio basado en clase también ocurre en otras industrias. Por ejemplo, las aerolíneas ofrecen *servicio de 1º, 2º y 3º clase*. Para los paquetes, las clases pueden diferir en términos de retardo, fluctuación y probabilidad de ser descartado en caso de congestión, entre otras posibilidades.

Reenvío asegurado

Especifica que deberá haber 4 clases de prioridades, y cada una tendrá sus propios recursos. Además, define 3 probabilidades de descarte para paquetes que están en congestión: baja, media y alta. En conjunto, estos dos factores definen 12 clases de servicios. Los pasos son:

- 1- Clasificar los paquetes en una de cuatro clases de prioridades, en el *host* emisor o en el enrutador de ingreso. La ventaja de realizar la clasificación en el *host* emisor es que hay más información disponible acerca de cuáles paquetes pertenecen a qué flujos.
- 2- Marcar los paquetes de acuerdo con su clase. (Encabezado IP – Tipo de Servicio)
- 3- Pasar los paquetes a través de un filtro modelador/eliminador que podría retardar o descartar algunos de ellos para dar una forma aceptable a los cuatro flujos,

En este ejemplo, estos tres pasos se realizan en el *host* emisor, por lo que el flujo de salida ahora se introduce en el enrutador de ingreso. Estos pasos pueden ser realizados por software especial de conectividad de redes o por el Sistema Operativo.

Reenvío expedito o acelerado

Dos clases de servicios están disponibles: regular y expedita. Se espera que la mayor parte del tráfico sea regular, pero una pequeña fracción de los paquetes son expeditos. Los paquetes expeditos deben tener la capacidad de transitar la subred como si no hubiera otros paquetes.

Una forma de implementar esta estrategia es programar los enrutadores para que tengan dos colas de salida por cada línea de salida, una para los paquetes expeditos y una para los regulares. La programación de paquetes debe utilizar algo parecido al encolamiento justo ponderado. [Por ejemplo, si 10% del tráfico es expedito y 90% es regular.](#) Se espera que los paquetes expeditos vean una red descargada, incluso cuando hay, de hecho, una carga pesada.

Conmutación de etiquetas ó MPLS (conmutación de etiquetas multiprotocolo)

La idea es agregar una etiqueta en frente de cada paquete y realizar el enrutamiento con base en ella y no con base en la dirección de destino. Al ser la etiqueta un índice de una tabla, encontrar la línea correcta de salida es una simple cuestión de buscar en una tabla.

Algunas personas hacen una distinción entre **enrutamiento** y **conmutación**. El enrutamiento es el proceso de buscar una dirección de destino en una tabla para saber a dónde enviar los paquetes hacia ese destino. En contraste, la conmutación utiliza una etiqueta que se toma de un paquete como un índice en una tabla de reenvío.

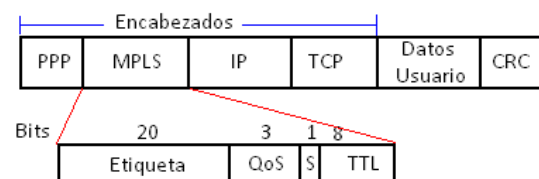
El primer problema es en dónde colocar la etiqueta. Como los paquetes IP no fueron diseñados para circuitos virtuales, su encabezado no tiene lugar. Se tuvo que agregar un nuevo encabezado MPLS enfrente del encabezado IP

El encabezado MPLS genérico tiene cuatro campos.

El campo *QoS* indica la clase de servicio. El campo

S se relaciona con colocar en una pila múltiples etiquetas en redes jerárquicas. Debido a que los encabezados MPLS no son parte del paquete de la

capa de red o de la trama del enlace de datos, MPLS es en gran medida independiente de ambas capas. De esta característica proviene la parte "multiprotocolo" del nombre MPLS.



Cuando un paquete mejorado con MPLS llega a un enrutador con capacidad MPLS, la etiqueta se utiliza como un índice en una tabla para determinar la línea de salida y la nueva etiqueta a utilizar. Esta conmutación de etiquetas se utiliza en todas las subredes de circuitos virtuales, debido a que las etiquetas sólo tienen importancia local y dos enrutadores diferentes pueden asignar la misma etiqueta a paquetes hacia diferentes destinos, es decir, la etiqueta es reasignada a la salida de cada enrutador, por lo que no se mantiene la misma etiqueta en toda la ruta.

Ciertamente es posible que cada flujo tenga su propio conjunto de etiquetas a través de la subred. Sin embargo, es más común que los enrutadores agrupen múltiples flujos que terminan en un enrutador o una LAN particulares y utilizan una sola etiqueta de ellos. Se dice que los flujos que están agrupados en una sola etiqueta pertenecen a la misma **FEC (clase de equivalencia de reenvío)**.

Con MPLS, los paquetes aún contienen su dirección de destino final, además de la etiqueta, a fin de que al final de la red de MPLS pueda eliminarse la etiqueta y que el reenvío pueda continuar de la forma normal, utilizando la dirección de destino de la capa de red. En MPLS no hay fase de configuración para cada conexión (pues eso podría romper con la operación de mucho software existente en Internet).

En su lugar, hay dos formas de crear las entradas de la tabla de reenvío. En el método **orientado a datos**, cuando un paquete llega, el primer enrutador que encuentra contacta al siguiente enrutador en el sentido descendente del flujo a donde tiene que ir el paquete, y le pide que genere una etiqueta para el flujo. Este método se aplica de manera recursiva. En efecto, ésta es una creación de circuitos virtuales por petición.

Los protocolos que hacen esta propagación son muy cuidadosos para evitar los ciclos cerrados. Por lo general, utilizan una técnica llamada **subprocesos con color**. Otra forma que se utiliza en las redes que no se basan en ATM, es el método **dirigido por control**.

INTERCONECTIVIDAD

Hasta ahora hemos supuesto de manera implícita que hay una sola red homogénea y que cada máquina usa el mismo protocolo en cada capa. Este supuesto es demasiado optimista.

Aspecto	Algunas posibilidades
Servicio ofrecido	Sin conexiones, orientado a conexiones
Protocolos	IP, IPX, SNA, ATM, MPLS, AppleTalk, etc.
Direccionamiento	Plano (802) o jerárquico (IP)
Multidifusión	Presente o ausente (también difusión)
Tamaño de paquete	Cada red tiene su propio máximo
Calidad del servicio	Puede estar presente o ausente; muchos tipos diferentes
Manejo de errores	Entrega confiable, ordenada y desordenada
Control de flujo	Ventana corrediza, control de tasa, otros o ninguno
Control de congestión	Cubeta con goteo, paquetes reguladores, etc.
Seguridad	Reglas de confidencialidad, encriptación, etc.
Parámetros	Diferentes terminaciones de temporizador, especificaciones de flujo, etc.
Contabilidad	Por tiempo de conexión, por paquete, por byte, o sin ella

Un enrutador que puede manejar múltiples protocolos se conoce como **enrutador multiprotocolo**. Nos enfocaremos en la interconectividad en la capa de red (routers). Con un conmutador (o puente), toda la trama se transporta con base en su dirección MAC. Con un enrutador, el paquete se extrae de la trama y la dirección del paquete se utiliza para decidir a dónde enviarlo.

Interconectividad Orientada a la Conexión (enfoque por Circuitos Virtuales)

En el modelo de circuitos virtuales concatenados, se establece una conexión con un *host* de una red distante. La subred ve que el destino es remoto y construye un circuito virtual al enrutador más cercano a la red de destino; luego construye un circuito virtual de ese enrutador a una **puerta de enlace externa** (enrutador multiprotocolo). Ésta registra la existencia del circuito virtual en sus tablas y procede a construir otro circuito virtual a un enrutador de la siguiente subred. Este proceso continúa hasta llegar al *host* de destino.

Una vez que comienzan a fluir paquetes de datos por la ruta, cada puerta de enlace retransmite los paquetes de entrada y hace las conversiones entre los formatos de paquete y los nº de circuito virtual, según sea necesario. Obviamente, todos los paquetes de datos deben atravesar la misma secuencia de puertas de enlace. En consecuencia, la red nunca reordena los paquetes de un flujo. Este esquema funciona mejor cuando todas las redes tienen aproximadamente las mismas propiedades.

El modelo de circuitos virtuales concatenados tiene en esencia las mismas ventajas que el uso de circuitos virtuales en una sola subred: pueden reservarse búferes por adelantado, puede garantizarse la secuencia, pueden usarse encabezados cortos y pueden evitarse los problemas causados por paquetes duplicados retrasados. El modelo también tiene las mismas desventajas: el espacio de tablas requerido en los enrutadores para cada conexión abierta, la falta de enrutamiento alternativo para evitar áreas congestionadas y la vulnerabilidad a fallas de los enrutadores a lo largo de la ruta. También tiene la desventaja de que su implementación es difícil, si no imposible, si una de las redes que intervienen es una red no confiable de datagramas.

Interconectividad no orientada a la conexión (enfoque por Datagramas)

Con un modelo de datagramas, se puede utilizar múltiples rutas y lograr de esta manera un ancho de banda mayor que el modelo de circuitos virtuales concatenados. Por otra parte, no hay garantía de que los paquetes llegarán al destino en orden, suponiendo que lleguen.

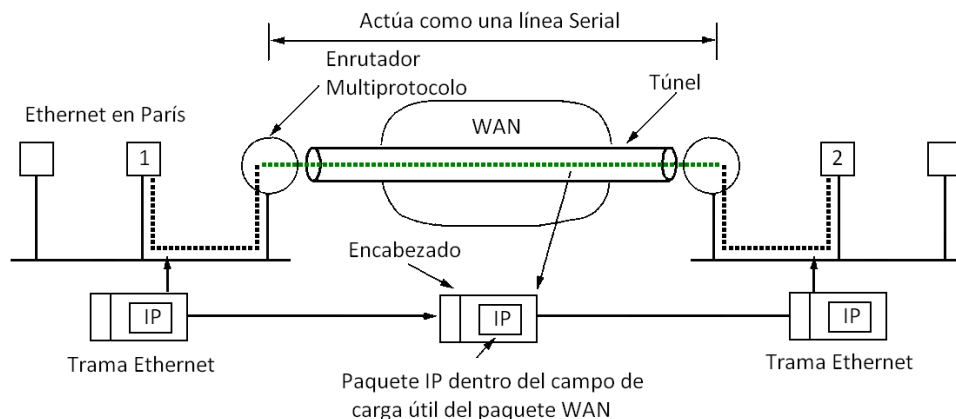
Podríamos imaginar a los enrutadores multiprotocolo tratando de traducir de un formato a otro, pero a menos que los dos formatos sean “parientes”, tales conversiones siempre serán incompletas, y destinadas al fracaso. Por esto, pocas veces se intentan las conversiones.

Un segundo problema, más serio, es el direccionamiento. Se podría diseñar un paquete universal de “interred” y hacer que todos los enrutadores lo reconozcan. Este enfoque es, precisamente, el que tiene IP: un paquete diseñado para llevarse por muchas redes.

Las propiedades del enfoque por datagramas para la interconectividad son las mismas que las de las subredes de datagramas: un mayor potencial de congestión, pero también mayor potencial para adaptarse a él, la robustez ante fallas de los enrutadores y la necesidad de encabezados más grandes. Una ventaja principal del enfoque por datagramas para la interconectividad es que puede usarse en subredes que no usan circuitos virtuales.

Entunelamiento

Lograr la interacción de dos redes diferentes es en extremo difícil. Sin embargo, hay un caso especial común que puede manejarse. Este caso es cuando el *host* de origen y el de destino están en la misma clase de red, pero hay una red diferente en medio.



La solución a este problema es una técnica llamada **entunelamiento**. Sólo el enrutador multiprotocolo tiene que entender los paquetes IP y WAN. De hecho, la distancia completa entre la mitad de un enrutador multiprotocolo y la mitad del otro actúa como una línea serial.

Enrutamiento entre redes

El enrutamiento a través de una interred es parecido al enrutamiento en una sola subred, pero con algunas complicaciones adicionales. *Cada enrutador multiprotocolo puede acceder (es decir, enviar paquetes) de manera directa a todos los demás enrutadores conectados a cualquier red a la que esté conectado.*

Una vez construido el grafo, pueden aplicarse algoritmos de enrutamiento conocidos al grupo de enrutadores multiprotocolo. Esto da un algoritmo de enrutamiento de dos niveles: en cada red se utiliza un **protocolo de puerta de enlace interior (IGP)**, pero entre ellas se usa un **protocolo de puerta de enlace exterior (EGP)** ("puerta de enlace" es un término antiguo para "enrutador").

Otra diferencia entre el enrutamiento interior y el exterior es el costo.

Fragmentación

Cada red impone un tamaño máximo a sus paquetes. Estos límites son debido a:

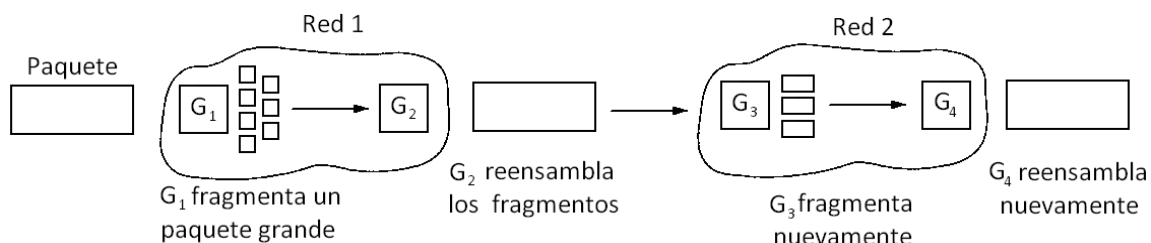
1. El hardware (*por ejemplo, el tamaño de una trama Ethernet*).
2. El sistema operativo (*por ejemplo, todos los búferes son de 512 bytes*).
3. Los protocolos (*por ejemplo, la cantidad de bits en el campo de longitud de paquete*).
4. El cumplimiento de algún estándar (inter)nacional.
5. El deseo de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

Las cargas útiles máximas van desde 48 bytes (celdas ATM) hasta 65,515 bytes (paquetes IP), aunque el tamaño de la carga útil en las capas superiores con frecuencia es más grande.

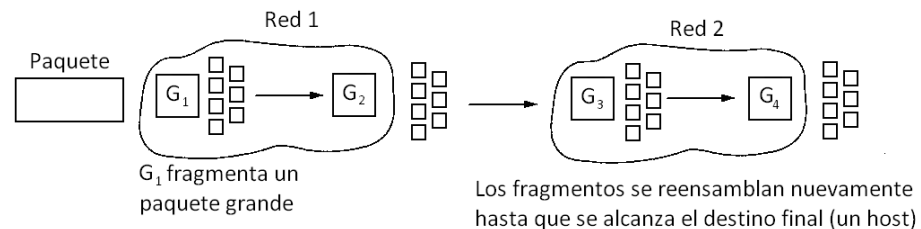
Surge un problema obvio cuando un paquete grande quiere viajar a través de una red cuyo tamaño máximo de paquete es demasiado pequeño. La única solución al problema es permitir que las puertas de enlace dividan los paquetes en **fragmentos**. Las redes de conmutación de paquetes también tienen problemas al unir nuevamente los fragmentos.

Existen dos estrategias opuestas para recombinar los fragmentos y recuperar el paquete original:

- **Fragmentación Transparente:** es hacer transparente la fragmentación causada por una red de "paquete pequeño" a las demás redes subsiguientes por las que debe pasar el paquete para llegar a su destino final. Las redes ATM, por ejemplo, tienen *hardware* especial para proporcionar fragmentación transparente de paquetes en celdas y luego reensamblar las celdas en paquetes. La fragmentación transparente es sencilla, pero tiene algunos problemas. Por una parte, la puerta de enlace de salida debe saber cuándo ha recibido todas las piezas, por lo que debe incluirse un campo de conteo o un bit de "fin de paquete" en cada paquete. Por otra parte, todos los paquetes deben salir por la misma puerta de enlace.



- Fragmentación no transparente: es abstenerse de recombinar los fragmentos en las puertas de enlace intermedias. Una vez que se ha fragmentado un paquete, cada fragmento se trata como si fuera un paquete original. La recombinación ocurre sólo en el *host* de destino. IP funciona de esta manera. Tiene algunos problemas:
 - Requiere que *"todos"* los *hosts* sean capaces de hacer el reensamble.
 - Al fragmentarse un paquete grande, aumenta la sobrecarga total, pues cada fragmento debe tener un encabezado.
 - Cuando se divide un paquete, los fragmentos deben numerarse (0.0), y si deben reenumerarse por una nueva fragmentación (0.0.1), y se pierden, surge la necesidad de retransmisiones de extremo a extremo, con efectos poco afortunados para el sistema de numeración.
 - Un sistema de numeración completamente diferente, y mejor, es que el protocolo de interred defina un tamaño de fragmento elemental lo bastante pequeño como para que el fragmento elemental pueda pasar a través de todas las redes. Este método requiere dos campos de secuencia en el encabezado de interred: el número original de paquete y el número de fragmento.



LA CAPA DE RED DE INTERNET

En la capa de red, Internet puede verse como un conjunto de subredes, o **sistemas autónomos** interconectados. No hay una estructura real, pero existen varias redes dorsales principales. Éstas se construyen a partir de líneas de alto ancho de banda y enrutadores rápidos. Conectadas a las redes dorsales hay redes regionales, y conectadas a éstas, están las LANs.

El pegamento que mantiene unida a Internet es el protocolo de capa de red, **IP (Protocolo de Internet)**. Se diseñó desde el principio con la interconexión de redes en mente. Su trabajo es proporcionar un medio de mejor esfuerzo (sin garantía) para el transporte de datagramas del origen al destino, sin importar si estas máquinas están en la misma red, o si hay otras entre ellas.

La comunicación en Internet funciona como sigue. La capa de transporte toma flujos de datos y los divide en datagramas. Cada datagrama se transmite a través de Internet, posiblemente fragmentándose en unidades más pequeñas en el camino. Cuando todas las piezas llegan finalmente a la máquina de destino, son reensambladas por la capa de red, dejando el datagrama original. A continuación este datagrama se entrega a la capa de transporte, que lo introduce en el flujo de entrada del proceso receptor.

El protocolo IP

VERS	LONG1	SERV	LONGITUD TOTAL		DIRECCIÓN ORIGEN
IDENTIFICADOR			FLAGS	OFFSET	DIRECCIÓN DESTINO
TTL		PROTO	CHECKSUM		PARTE OPCIONAL
DATOS					

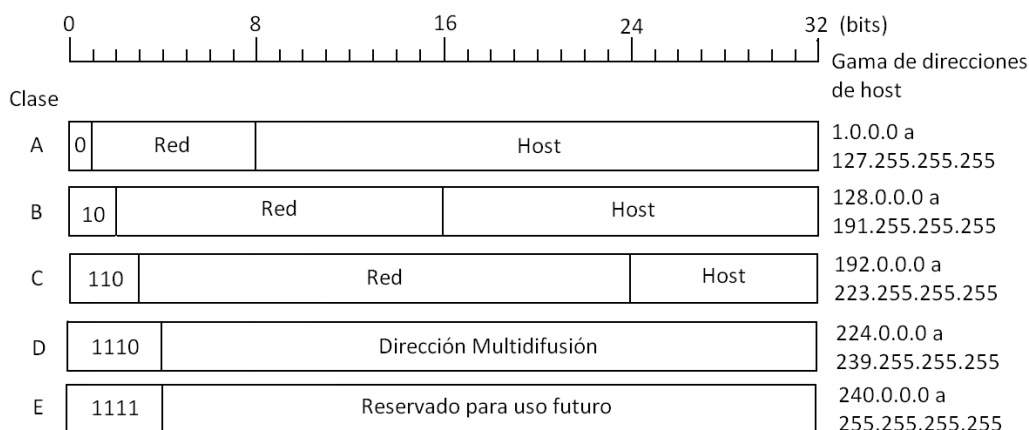
El campo de *Opciones* se diseñó para proporcionar un recurso que permitiera que las versiones subsiguientes del protocolo incluyeran información no presente en el diseño original, para permitir que los experimentadores prueben ideas nuevas y para evitar la asignación de bits de encabezado a información pocas veces necesaria.

Opción	Descripción
Seguridad	Especifica qué tan secreto es el datagrama
Enrutamiento estricto desde el origen	Indica la ruta completa a seguir
Enrutamiento libre desde el origen	Da una lista de los enrutadores que no deben evitarse
Registrar ruta	Hace que cada enrutador agregue su dirección IP
Marca de tiempo	Hace que cada enrutador agregue su dirección y marca de tiempo

Direcciones IP

Cada *host* y enrutador de Internet tiene una dirección IP, que codifica su número de red y su número de *host*. La combinación es única. Todas las direcciones IP son de 32 bits de longitud y se usan en los campos de *Dirección de origen* y de *Dirección de destino* de los paquetes IP. En importante mencionar que una dirección IP realmente no se refiere a un *host*. En realidad se refiere a una interfaz de red, por lo que si un *host* está en dos redes, debe tener dos direcciones IP.

Las direcciones IP se dividieron en cinco categorías. Esta asignación se ha llamado **direccionamiento con clase**. Ya no se utiliza.



Los números de redes son manejados por una corporación no lucrativa llamada **ICANN (Corporación de Internet para la Asignación de Nombres y Números)** para evitar conflictos.

Algunas direcciones IP especiales son:

00000000000000000000000000000000	Cuando arranca un host
000 000 Host	Un host de esta red
11111111111111111111111111111111	Difusión en la LAN
Red 111 111	Difusión en red Distante
127 (Cualquier cosa)	Loopback (dirección local de prueba)

Subredes

Permiten la división de una red en varias partes para uso interno, pero aún actuar como una sola red ante el mundo exterior.

En lugar de tener una sola dirección de clase B con 14 bits para el número de red y 16 bits para el número de *host*, algunos bits se eliminan del número de *host* para crear un número de subred. Por ejemplo, si la universidad tiene 35 departamentos, podría utilizar un número de subred de 6 bits ($2^5 < 35 < 2^6$) y un número de *host* de 10 bits, lo que permitiría hasta 64 Ethernets, cada una con un máximo de 1022 *hosts* (todos 0's o 1's no están disponibles, como se mencionó anteriormente). Esta división podría cambiarse posteriormente en caso de que no fuera correcta.

Para implementar subredes, el enrutador principal necesita una **máscara de subred**. Las máscaras de subred también se pueden escribir en notación decimal con puntos, o agregando a la dirección IP una diagonal seguida del número de bits usados para los números de red y subred. Para el ejemplo de la figura, la máscara de subred puede escribirse como 255.255.252.0. Una notación alternativa es /22 para indicar que la máscara de subred tiene una longitud de 22 bits.

	10	Red	Subred	Host
Mascara Subred	11	111111 11111111	111111	00 00000000

	Red		Subred	Host
Subred 1	10000010	00110010	000001	00 00000001
Subred 2	10000010	00110010	000010	00 00000001
Subred 3	10000010	00110010	000011	00 00000001

Para ver el funcionamiento de las subredes, es necesario explicar la manera en que se procesan los paquetes IP en un enrutador. Cada enrutador tiene una tabla en la que se lista cierto número de direcciones IP (red, 0) y cierto número de direcciones IP (esta red, *host*). El primer tipo indica cómo llegar a redes distantes. El segundo tipo indica cómo llegar a redes locales. **La interfaz de red a utilizar para alcanzar el destino, así como otra información, está asociada a cada tabla.**

Cuando llega un paquete IP, si el paquete es para una red distante, se reenvía al siguiente enrutador de la interfaz dada en la tabla; si es para un *host* local, se envía directamente al destino. Si la red no está en la tabla, el paquete se reenvía a un enrutador predeterminado con tablas más extensas.

Al introducirse subredes, se cambian las tablas de enrutamiento, agregando entradas con forma de (esta red, subred, 0) y (esta red, esta subred, *host*). Por lo tanto, un enrutador de la subred *k* sabe cómo llegar a todas las demás subredes y a todos los *hosts* de la subred *k*. No tiene que saber los detalles sobre los *hosts* de otras subredes. De hecho, todo lo que se necesita es hacer que cada enrutador haga un AND booleano con la máscara de subred de la red para deshacerse del número de *host* y buscar la dirección resultante en sus tablas.

Por lo tanto, la división de redes reduce espacio en la tabla de enrutamiento creando una jerarquía de tres niveles, que consiste en red, subred y *host*.

CIDR—Enrutamiento interdominios sin clases

Desgraciadamente, el IP se está convirtiendo con rapidez en víctima de su propia popularidad: se le están acabando las direcciones. Este desastre inminente ha propiciado una gran cantidad de controversias y debates en la comunidad de Internet sobre lo que debe hacerse al respecto.

En teoría, existen cerca de dos mil millones de direcciones, pero la práctica de organizar el espacio de direcciones por clases desperdicia millones de ellas. En particular, el verdadero villano es la red clase B. Para la mayoría de las organizaciones, una red clase A, con 16 millones de direcciones, es demasiado grande, y una red clase C, de 256 direcciones, es demasiado pequeña. Una red clase B, con 65,536, es la adecuada.

En resumen, la mayoría de las soluciones resuelven un problema pero crean uno nuevo. La solución que se implementó y que dio a Internet un respiro es el **CIDR (Enrutamiento Interdominios sin Clases)**. El concepto básico del CIDR, es asignar las direcciones IP restantes en bloques de tamaño variable (pero de potencias de 2), independientemente de las clases. Eliminar las clases hace más complicado el reenvío.

Con CIDR, este algoritmo sencillo ya no funciona. En cambio, cada entrada de tabla de enrutamiento se extiende para darle una máscara de 32 bits. De esta manera, ahora hay una sola tabla de enrutamiento para todas las redes que consten de un arreglo de tres variables (dirección IP, máscara de subred, línea saliente). Cuando llega un paquete, primero se extrae su dirección de destino IP. Luego (conceptualmente) se analiza la tabla de enrutamiento entrada por entrada, enmascarando la dirección de destino y comparándola con la entrada de la tabla buscando una correspondencia. Es posible que coincidan entradas múltiples (con diferentes longitudes de máscara de subred), en cuyo caso se usa la máscara más larga. De esta manera, si hay una coincidencia para una máscara /20 y una máscara /24, se usa la entrada /24.

NAT—Traducción de Dirección de Red

Las direcciones IP son escasas. Un ISP podría tener una dirección de /16 (clase B), dándole 65,534 números de host. Si tiene más clientes que esos, tiene un problema. Para clientes propios con las conexiones de línea conmutada, una manera de resolver el problema es asignar dinámicamente una dirección IP a una computadora cuando ésta llama e inicia la sesión y tomar de vuelta la dirección IP cuando se termina la sesión. Esta estrategia trabaja bien para un ISP con un número moderado de usuarios propios, pero falla para ISPs que sirven sobre todo a clientes comerciales.

Para empeorar las cosas, más y más usuarios caseros se suscriben a ADSL. Dos de los rasgos de estos servicios son: (1) el usuario consigue una dirección IP permanente y (2) no hay ningún cargo por la conexión, por lo que los usuarios de ADSL y de cable se quedan registrados de manera permanente. Este desarrollo se agrega a la escasez de direcciones IP. La solución a largo plazo es que todo Internet emigre a IPv6, que tiene direcciones de 128 bits. Como consecuencia, algunas personas sentían que se necesitaba un arreglo rápido a corto plazo. Este arreglo surgió en la forma de la Traducción de Dirección de Red (NAT).

La idea básica de NAT es asignar una sola dirección IP a cada compañía (o a pocas) para el tráfico de Internet. Dentro de la compañía, cada computadora tiene una dirección IP única que se usa para enrutar el tráfico interno. Sin embargo, cuando un paquete sale de la compañía y va al ISP, se presenta una traducción de dirección. Para hacer posible este esquema los tres rangos de direcciones IP se han declarado como privados. Las compañías pueden usarlos internamente cuando lo deseen. La única regla es que ningún paquete que contiene estas direcciones puede aparecer en la propia Internet. Los tres rangos reservados son:

10.0.0.0	10.255.255.255/8	16.777.216 hosts
172.16.0.0	172.31.255.255/12	1.048.576 hosts
192.168.0.0	192.168.255.255/16	65.536 hosts

Dentro de las instalaciones de la compañía, cada máquina tiene una dirección única de la forma 10.x.y.z. Sin embargo, en este ejemplo cuando un paquete sale de las instalaciones de la compañía, pasa a través de una **caja NAT** que convierte la dirección interna de origen de IP, 10.0.0.1, a la verdadera dirección IP de la compañía, 198.60.42.12. A menudo, la caja NAT se combina en un solo dispositivo con un *firewall*

Cuando la respuesta vuelve, ¿cómo sabe ahora la caja NAT con qué dirección se reemplaza?

Los diseñadores de NAT observaron que la mayoría de paquetes IP lleva cargas útiles de TCP o UDP. Los puertos son enteros de 16 bits que indican dónde empieza y dónde acaba la conexión TCP. Estos puertos proporcionan el campo requerido para hacer que NAT funcione.

Aunque esta clase de esquema resuelve el problema, muchas personas en la comunidad de IP, lo consideran a primera vista como una abominación, debido a que:

- NAT viola el modelo arquitectónico de IP que establece que cada dirección IP identifica una sola máquina globalmente.
- NAT cambia a Internet de una red sin conexión a un tipo de red orientada a la conexión.
- NAT viola la regla más fundamental de los protocolos de capas: la capa k no puede hacer ninguna suposición de en qué capa $k + 1$ ha puesto el campo de carga útil.
- En Internet no se exige que los procesos utilicen TCP o UDP.
- Algunas aplicaciones insertan direcciones IP en el cuerpo del texto. Fallará cualquier intento por usar luego estas direcciones.

Protocolos de Control en Internet

Protocolo de Mensajes de Control en Internet (ICMP)

Cuando ocurre algo inesperado, el **ICMP** informa del evento. Hay definidos alrededor de una docena de tipos de mensajes ICMP. Cada tipo de mensaje ICMP se encapsula en un paquete IP.

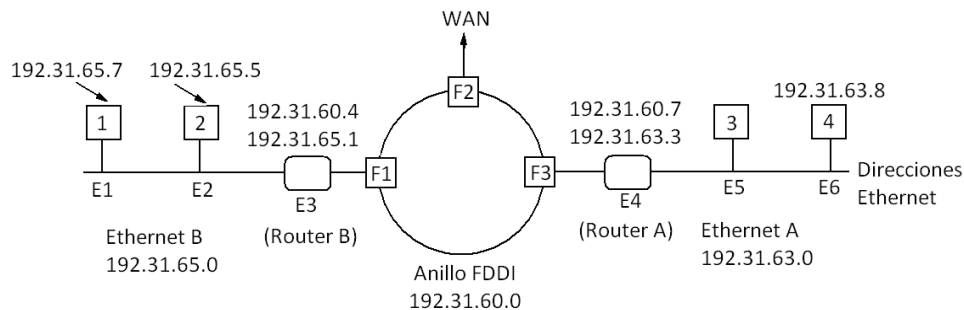
Tipo de mensaje	Descripción
Destination unreachable	El paquete no se pudo entregar
Time exceeded	Campo de tiempo de vida = 0
Parameter problem	Campo de encabezado no válido
Source quench	Paquete regulador
Redirect	Enseña a un enrutador sobre geografía
Echo	Pregunta a una máquina si está viva
Echo reply	Sí, estoy viva
Timestamp request	Misma que solicitud de eco, pero con marca de tiempo
Timestamp reply	Misma que respuesta de eco, pero con marca de tiempo

ARP—Protocolo de Resolución de Direcciones

Aunque en Internet cada máquina tiene una (o más) direcciones IP, en realidad éstas no pueden usarse para enviar paquetes porque el hardware de capa de enlace de datos no entiende las direcciones de Internet. Una tarjeta de red sólo entiende direcciones LAN. Las tarjetas envían y reciben tramas basadas en direcciones Ethernet de 48 bits. No saben nada de direcciones IP.

Empezaremos viendo cómo un usuario en el *host* 1 envía un paquete a un usuario en el *host* 2. Supongamos que el emisor sabe el nombre del receptor pretendido, posiblemente algo como *mary@eagle.cs.uni.edu*. El primer paso es encontrar la dirección IP para el *host* 2, conocido como *eagle.cs.uni.edu*. Esta consulta la realiza el DNS, que asumiremos que devuelve la dirección IP al *host* 2 (192.31.65.5).

El software de la capa superior en el *host* 1 elabora ahora un paquete con 192.31.65.5 en el campo *Dirección de destino* y lo da a transmitir al software IP. Este puede buscar la dirección y ver que el destino esté en su propia red, pero necesita alguna manera de encontrar la dirección Ethernet de destino. Una solución es tener un archivo de configuración en alguna parte del sistema que relacione direcciones IP con direcciones Ethernet. Aun cuando esta solución es ciertamente posible, para las organizaciones con miles de máquinas, conservar todos estos archivos actualizados propicia errores y consume mucho tiempo.



Una mejor solución es que el *host* 1 dé salida a un paquete de difusión hacia Ethernet preguntando: ¿quién posee la IP 192.31.65.5? De esta manera, el *host* 1 aprende que esa dirección IP 192.31.65.5 está en el *host* con la dirección Ethernet E2. El protocolo utilizado para hacer esta pregunta y obtener la respuesta se llama **Protocolo de Resolución de Direcciones (ARP)**.

La ventaja de usar ARP en lugar de archivos de configuración es la sencillez. El gerente de sistemas sólo tiene que asignar a cada máquina una dirección IP y decidir respecto de las máscaras de subred. ARP hace el resto.

A estas alturas, el software IP en el *host* 1 crea una trama Ethernet dirigida a E2, pone el paquete IP (dirigido a 192.31.65.5) en el campo de carga útil, y lo descarga hacia la Ethernet. La tarjeta Ethernet del *host* 2 detecta esta trama, la reconoce como una trama para sí mismo, lo recoge, y provoca una interrupción. El controlador de Ethernet extrae el paquete IP de la carga útil y lo pasa al software IP, que ve que esté direccionado correctamente y lo procesa.

Se pueden hacer varias optimizaciones para que ARP trabaje con más eficiencia. Para empezar, una vez que una máquina ha ejecutado ARP, guarda el resultado en caso de tener que ponerse en poco tiempo de nuevo en contacto con la misma máquina. La siguiente vez encontrará la correspondencia en su propio caché, eliminando así la necesidad de una segunda difusión. En muchos casos el *host* 2 necesitará devolver una respuesta, forzando, también, a que se ejecute el ARP para determinar la dirección Ethernet del emisor. Esta difusión de ARP puede evitarse teniendo el *host* 1 que incluir su correspondencia IP a Ethernet en el paquete ARP. Cuando la difusión de ARP llega al *host* 2, se introduce (192.31.65.7, E1) en el caché ARP del *host* 2 para uso futuro. De hecho, todas las máquinas en Ethernet pueden introducir esta correspondencia en su caché ARP.

Otra optimización más es que cada máquina difunda su correspondencia cuando arranca. Por lo general, esta difusión se hace en forma de un ARP que busca su propia dirección IP. No debe haber

una respuesta, pero un efecto lateral de la difusión es hacer una entrada en el caché ARP de todas las máquinas. Si llega (inesperadamente) una respuesta, es que la misma dirección IP se ha asignado a dos máquinas. La más reciente debe informar al gerente de sistemas y no arrancar.

Para permitir que las correspondencias cambien, las entradas en el caché ARP deben expirar en unos cuantos minutos.

Ahora veamos de nuevo la figura anterior. Esta vez el *host* 1 quiere enviar un paquete al *host* 4 (192.31.63.8). Si utiliza ARP fallará porque el *host* 4 no verá la difusión (los enrutadores no envían difusiones a nivel Ethernet). Hay dos soluciones. Primera, podría configurarse el enrutador B para que responda a las solicitudes de ARP de la red 192.31.63.0 (y posiblemente de otras redes locales). En este caso, el *host* 1 introducirá (192.31.63.8, E3) en el caché ARP y enviará felizmente todo el tráfico del *host* 4 al enrutador local. Esta solución se llama **proxy ARP**. La segunda solución es hacer que el *host* 1 vea de inmediato que el destino está en una red remota y simplemente envíe todo ese tráfico a una dirección Ethernet predefinida que se ocupe de todo el tráfico remoto, en este caso E3. Esta solución no requiere que el enrutador B sepa a qué redes remotas está sirviendo.

De cualquier modo, lo que sucede es que el *host* 1 empaqueta el paquete IP en el campo de carga útil de una trama Ethernet dirigida a E3. Cuando el enrutador B obtiene la trama Ethernet, retira el paquete IP del campo de carga útil y busca la dirección IP en sus tablas de enrutamiento. Descubre que se supone que los paquetes para la red 192.31.63.0 van al enrutador 192.31.60.7. Si aún no conoce la dirección FDDI de 192.31.60.7, transmite un paquete ARP al anillo y aprende que su dirección del anillo es F3. Inserta entonces el paquete en el campo de carga útil de una trama FDDI dirigido a F3 y la coloca en el anillo.

En el enrutador A, el controlador de FDDI retira el paquete del campo de carga útil y lo da al software IP, que ve que necesita enviar el paquete a 192.31.63.8. Si esta dirección IP no está en su caché ARP, transmite una solicitud de ARP en la Ethernet A y aprende que la dirección de destino es E6, por lo que construye una trama Ethernet dirigida a E6, pone el paquete en el campo de carga útil y lo envía a través de Ethernet. Cuando la trama Ethernet llega al *host* 4, se extrae el paquete de la trama y se pasa al software IP para su procesamiento.

Ir del *host* 1 a una red distante a través de una WAN funciona esencialmente de la misma manera, sólo que esta vez las tablas del enrutador B le dicen que utilice el enrutador de WAN cuya dirección FDDI es F2.

RARP, BOOTP y DHCP

A veces se tiene que resolver el problema inverso: dada una dirección Ethernet, ¿cuál es la dirección IP correspondiente? Esto ocurre cuando se inicializa una estación de trabajo sin disco.

La primera solución inventada fue usar el **RARP (Protocolo de Resolución de Dirección de Retorno)**. Este protocolo permite que una estación de trabajo recientemente inicializada transmita su dirección Ethernet y diga: "Mi dirección Ethernet de 48 bits es 14.04.05.18.01.25. ¿Alguien allá afuera conoce mi dirección IP?" El servidor RARP ve esta solicitud, busca la dirección Ethernet en sus archivos de configuración y devuelve la dirección IP correspondiente.

Una desventaja de RARP es que usa una dirección de destino de todos los 1s (de difusión limitada) para llegar al servidor RARP. Sin embargo, dichas difusiones no las envían los enrutadores, por lo que se necesita un servidor RARP en cada red. Para resolver este problema, se inventó un protocolo de arranque alternativo llamado **BOOTP**. El BOOTP usa mensajes UDP. También proporciona información adicional a una estación de trabajo sin disco. Un problema es que BOOTP

requiere configuración manual de tablas para relacionar una dirección IP con una dirección Ethernet.

Para eliminar este paso conducente a errores, el BOOTP se extendió y se le dio un nuevo nombre: **DHCP (Protocolo de Configuración de Host Dinámico)**. DHCP permite asignación de dirección IP manual y automática. Este servidor no necesita estar en la misma LAN que el *host* solicitante. Puesto que el servidor DHCP no se puede alcanzar por difusión, se necesita un **agente de retransmisión DHCP** en cada LAN.

Para encontrar su dirección IP, una máquina inicializada recientemente difunde un paquete DHCP DISCOVER. El agente de retransmisión DHCP de su LAN intercepta todas las difusiones DHCP. Cuando encuentra un paquete DHCP DISCOVER, envía el paquete mediante unidifusión al servidor DHCP, posiblemente en una red distante. La única pieza de información que el agente de retransmisión necesita es la dirección IP del servidor DHCP.

Un problema que surge con la asignación automática de direcciones IP de un rango de direcciones, es por cuánto tiempo debe asignarse una dirección IP. Para esto, se usa una técnica llamada **arrendamiento**. Simplemente, antes de que expire el arriendo, el *host* debe pedirle una renovación al DHCP. Si no hace una solicitud o ésta se le niega, el *host* ya no puede usar la dirección IP que se le dio antes.

Enrutamiento en Internet

OSPF - Protocolos de enrutamiento de puerta de enlace interior

Un algoritmo de enrutamiento dentro de un sistema autónomo se llama **protocolo de puerta de enlace interior (IGP)**; un algoritmo para enrutamiento entre sistemas autónomos se llama **protocolo de puerta de enlace exterior (EGP)**.

El protocolo de puerta de enlace interior original de Internet era un protocolo de vector de distancia (RIP). Su sucesor, se llamó **OSPF (Abrir Primero la Ruta más Corta)**.

OSPF soporta tres tipos de conexiones y redes:

1. Las líneas punto a punto exactamente entre dos enrutadores.
2. Redes de multiacceso con difusión (por ejemplo, la mayoría de las LANs).
3. Redes de multiacceso sin difusión (por ejemplo, la mayoría de las WANs de paquetes conmutados).

Una red de **multiacceso** es la que puede tener múltiples enrutadores, cada uno de los cuales se puede comunicar directamente con todos los demás. Todas las LANs y WANs tienen esta propiedad.

OSPF funciona resumiendo la colección de redes reales, enrutadores y líneas en un grafo dirigido en el que a cada arco se asigna un costo. Entonces calcula la ruta más corta con base en los pesos de los arcos. Los arcos del nodo de la red a los enrutadores tienen peso 0 y se omiten del grafo.

Muchos de los sistemas autónomos en Internet son grandes por sí mismos y nada sencillos de administrar. OSPF les permite dividirlos en **áreas** numeradas donde un área es una red o un conjunto de redes inmediatas. Las áreas no se traslapan ni la necesidad es exhaustiva, es decir, algunos enrutadores no pueden pertenecer a área alguna. Un área es una generalización de una subred. Fuera de un área, su topología y detalles no son visibles.

Cada sistema autónomo tiene un área de **red dorsal**, llamada 0. Todas las áreas se conectan a la red dorsal, posiblemente por túneles, de modo que es posible entrar desde cualquier área en el sistema autónomo a cualquier otra área en el sistema autónomo mediante la red dorsal. En el grafo un túnel se representa como un arco y tiene un costo. Cada enrutador que se conecta a dos o más áreas es parte de la red dorsal. Como con otras áreas, la topología de la red dorsal no es visible fuera de ésta.

Dentro de un área, cada enrutador tiene la misma base de datos del estado del enlace y ejecuta el mismo algoritmo de la ruta más corta. Su trabajo principal es calcular el camino más corto desde sí mismo a cualquier otro enrutador en el área, incluso el enrutador que se conecta a la red dorsal, de la que debe haber una por lo menos. Un enrutador que conecta dos áreas necesita las bases de datos para las dos áreas y debe ejecutar el algoritmo de la ruta más corta por separado.

Durante la operación normal, pueden necesitarse tres tipos de rutas: dentro del área, entre áreas y entre sistemas autónomos. Las rutas dentro del área son las más fáciles, puesto que el enrutador de origen ya conoce el camino más corto al enrutador de destino. El enrutamiento entre áreas siempre procede en tres pasos: va del origen a la red dorsal; va a través de la red dorsal al área de destino; va al destino. Este algoritmo fuerza una configuración de estrella en OSPF con la red dorsal actuando como concentrador y las otras áreas como rayos. Los paquetes se enrutan del origen al destino "como están". No se encapsulan ni se entunelan, a menos que vayan a un área cuya única conexión a la red dorsal sea un túnel.

OSPF distingue cuatro clases de enrutadores:

1. Enrutadores internos que están totalmente dentro de un área.
2. Enrutadores de límite de área que conectan dos o más áreas.
3. Enrutadores de la red dorsal que están en la red dorsal.
4. Enrutadores fronterizos de sistemas autónomos que se comunican con los enrutadores de otros sistemas autónomos.

Estas clases se pueden traslapar. Por ejemplo, todos los enrutadores de límite de área forman parte de la red dorsal automáticamente. Además, un enrutador que está en la red dorsal pero no forma parte de cualquier otra área también es un enrutador interno.

Cuando un enrutador se inicia, envía mensajes HELLO en todas sus líneas punto a punto y los multidifunde en las LANs al grupo que contiene los enrutadores restantes. En las WANs, necesita alguna información de configuración para saber a quién contactar. A partir de las respuestas, cada enrutador aprende quiénes son sus vecinos. Todos los enrutadores en la misma LAN son vecinos.

OSPF trabaja intercambiando información entre enrutadores adyacentes, que no es lo mismo que entre enrutadores vecinos. En particular, es ineficaz tener cualquier enrutador en la LAN que se comunica con cualquier otro enrutador en la LAN. Para evitar esta situación, se elige un enrutador como enrutador designado. Se dice que es adyacente a todos los demás enrutadores en su LAN, e intercambia información con ellos. Los enrutadores vecinos que no son adyacentes no intercambian información entre sí. Un enrutador designado como respaldo siempre se guarda actualizado, para facilitar la transición en caso de que el primer enrutador designado se cayera y necesitara ser reemplazado de manera inmediata.

Durante la operación normal, cada enrutador inunda periódicamente con mensajes LINK STATE UPDATE a cada uno de sus enrutadores adyacentes. Este mensaje da su estado y proporciona los costos usados en la base de datos topológica. Para hacerlos confiables, se confirma la recepción de los mensajes de inundación. Cada mensaje tiene un número de secuencia para que un enrutador pueda ver si un LINK STATE UPDATE entrante es más viejo o más nuevo que el que tiene actualmente. Los enrutadores también envían estos mensajes cuando una línea sube o baja o su costo cambia.

Los mensajes DATABASE DESCRIPTION dan los números de secuencia de todas las entradas de estado del enlace poseídas por el emisor actualmente. Comparando sus propios valores con los del emisor, el receptor puede determinar quién tiene los valores más recientes. Estos mensajes se usan cuando se activa una línea.

Cualquier socio puede pedir información del estado del enlace al otro usando los mensajes LINK STATE REQUEST. El resultado de este algoritmo es que cada par de enrutadores adyacentes hace una verificación para ver quién tiene los datos más recientes, y de esta manera se difunde la nueva información a lo largo del área. Todos estos mensajes se envían como paquetes IP.

Tipo de mensaje	Descripción
Hello	Descubre quiénes son los vecinos
Link state update	Proporciona los costos del emisor a sus vecinos
Link state ack	Confirma la recepción de la actualización del estado del enlace
Database description	Anuncia qué actualizaciones tiene el emisor
Link state request	Solicita información del socio

Finalmente podemos reunir todas las piezas. Utilizando la inundación de mensajes, cada enrutador informa a todos los demás enrutadores en su área sobre sus vecinos y costos. Esta información permite a cada enrutador construir el grafo para su(s) área(s) y calcular la ruta más corta. El área de la red dorsal también hace esto. Además, los enrutadores de la red dorsal aceptan la información de los enrutadores del límite de área para calcular la mejor ruta de cada enrutador de la red dorsal a cada enrutador restante. Esta información se difunde a los enrutadores de límite de área que la anuncian dentro de sus áreas. Usando esta información, un enrutador que está a punto de enviar un paquete dentro del área puede seleccionar el enrutador de mejor salida a la red dorsal.

BGP—Protocolo de Puerta de Enlace de Frontera

Dentro de un solo sistema autónomo, el protocolo de enrutamiento recomendado es OSPF. Entre los sistemas autónomos se utiliza un protocolo diferente, el **Protocolo de Puerta de Enlace de Frontera (BGP)**. Se necesita un protocolo diferente entre sistemas autónomos porque los objetivos de un protocolo de puerta de enlace interior y un protocolo de puerta de enlace exterior no son los mismos. Todo lo que tiene que hacer un protocolo de puerta de enlace interior es mover lo más eficazmente posible los paquetes del origen al destino. No tiene que preocuparse por las políticas.

Los enrutadores del protocolo de puerta de enlace exterior tienen que preocuparse en gran manera por la política. Algunos ejemplos de limitaciones de enrutamiento son:

1. Ningún tránsito a través de ciertos sistemas autónomos.
2. Nunca ponga Irak en una ruta que inicie en el Pentágono.
3. No pasar por Estados Unidos para llegar de la Columbia Británica a Ontario.
4. Transite por Albania sólo si no hay otra alternativa al destino.
5. El tráfico que empieza o termina en IBM no debe transitar por Microsoft.

Las políticas en cada enrutador de BGP se configuran manualmente y no son parte del protocolo.

Desde el punto de vista de un enrutador de BGP, el mundo consiste en sistemas autónomos y las líneas que los conectan. Dos sistemas autónomos se consideran conectados si hay una línea entre un enrutador fronterizo en cada uno. Dado el especial interés de BGP en el transporte de tráfico, las redes se agrupan en una de tres categorías. La primera son las **redes stub**, que tienen sólo una conexión con el grafo de BGP. Éstas no se pueden usar para transportar tráfico porque no hay nadie en el otro lado. Luego vienen las **redes multiconectadas**. Éstas podrían usarse para el transporte de tráfico excepto que lo rechacen. Finalmente, están las **redes de tránsito**, como redes dorsales, que están dispuestas a ocuparse de paquetes de terceros, posiblemente con algunas restricciones, y normalmente por pago.

Los pares de enrutadores de BGP se comunican entre sí estableciendo conexiones TCP. Operando de esta manera proporcionan comunicación confiable y ocultan todo detalle de red que pase a través de ellos.

Básicamente, BGP es muy parecido a un protocolo de vector de distancia, pero muy diferente de la mayoría de otros como RIP. En lugar de mantener el costo para cada destino, cada enrutador de BGP guarda el registro de la ruta utilizada, por lo que se conoce como un protocolo de vector de ruta. Del mismo modo, en lugar de darle a cada vecino el costo de cada posible destino estimado periódicamente, cada enrutador de BGP les dice el camino exacto que está usando.

Multidifusión de Internet

IP apoya la multidifusión, usando direcciones clase D. Cada dirección clase D identifica un grupo de *hosts*. Hay 28 bits disponibles para identificar los grupos, por lo que pueden existir al mismo tiempo más de 250 millones de grupos. Cuando un proceso envía un paquete a una dirección clase D, se hace el mejor esfuerzo para entregarlo. Se soportan dos tipos de direcciones de grupo: las permanentes y las temporales.

Los grupos temporales se deben crear antes de que se puedan usar. Un proceso puede pedir a su *host* que se una a un grupo específico. También puede pedirle que deje el grupo. Cada *host* conserva el registro de qué grupos pertenecen actualmente a sus procesos.

La multidifusión se implementa mediante enrutadores de multidifusión especiales que pueden o no colocarse con los enrutadores normales. Alrededor de una vez por minuto, cada uno de estos enrutadores envía una multidifusión de hardware a los *hosts* en su LAN pidiéndoles que devuelvan información de los grupos a los que pertenecen actualmente sus procesos.

Estos paquetes de preguntas y respuestas utilizan un protocolo llamado **IGMP (Protocolo de Administración de Grupo de Internet)** que es vagamente análogo al ICMP. Tiene sólo dos tipos de paquetes: pregunta y respuesta, cada uno con un formato simple, fijo, que contiene alguna información de control en la primera palabra del campo de carga útil y una dirección clase D en la segunda palabra. El enrutamiento de multidifusión se crea utilizando árboles de difusión.

IP móvil

El villano real en IP es el propio esquema de direccionamiento. Cada dirección IP contiene un número de red y un número de *host*. Los enrutadores en todo el mundo tienen tablas de enrutamiento que indican qué línea usar para conseguir conectarse a la red 160.80. Siempre que un paquete llegue con un destino de dirección IP de la forma 160.80.xxx.yyy, saldrá de esa línea.

Si de repente la máquina con esa dirección se lleva fuera a algún sitio distante, los paquetes se le seguirán enrutando a su LAN principal (o enrutador). Dar a la máquina una nueva dirección IP que corresponda a su nueva situación es poco atractivo.

Otro método es que los enrutadores tengan que usar direcciones IP completas para enrutamiento, en lugar de sólo la red. Sin embargo, esta estrategia requeriría que cada enrutador tuviera millones de entradas de tablas, a un costo astronómico para Internet.

Se buscaba una solución, teniendo en cuenta los siguientes objetivos:

- Cada *host* móvil debe poder usar su dirección IP principal en cualquier parte.
- No se permiten cambios de software a los *hosts* fijos.
- No se permiten cambios al software ni a las tablas del enrutador.
- La mayoría de paquetes para *host* móviles no debe hacer desvíos en la ruta.
- No se debe incurrir en sobrecarga cuando un *host* móvil está en casa.

La solución escogida fue la de los agentes base y foráneos. Cuando un *host* móvil se presenta a un sitio foráneo, contacta al *host* foráneo y se registra. El *host* foráneo entonces contacta al agente de base del usuario y le da una **dirección temporal**, normalmente la propia dirección IP del agente foráneo.

Cuando un paquete llega a la LAN principal del usuario, entra a algún enrutador adjunto a la LAN. Por lo general, el enrutador trata de localizar al *host*, difundiendo un paquete ARP preguntando, por ejemplo: ¿cuál es la dirección Ethernet de 160.80.40.20? El agente de base responde a esta pregunta dando su propia dirección de Ethernet. El enrutador envía entonces los paquetes para 160.80.40.20 al agente principal. A su vez, este último los canaliza a la dirección temporal encapsulándolos en el campo de carga útil de un paquete IP dirigido al agente foráneo. El agente foráneo entonces lo desencapsula y lo entrega a la dirección del enlace de datos del *host* móvil. Además, el agente de base da la dirección temporal al emisor, para que los paquetes futuros se puedan canalizar directamente al agente foráneo. Esta solución reúne todos los requisitos declarados anteriormente.

Tal vez valga la pena mencionar un pequeño detalle. Cuando el *host* se mueve, el enrutador probablemente tenga su dirección Ethernet en el caché (próxima a quedar invalidada). Reemplazar esa dirección Ethernet con la del agente de base se hace por un truco llamado **ARP gratuito**. Éste es un mensaje especial, no solicitado al enrutador que hace reemplazar una entrada específica del caché, en este caso la del *host* móvil que está a punto de salir. Cuando el *host* móvil vuelve después, se usa el mismo truco para actualizar de nuevo el caché del enrutador.

Nada en el diseño le impide a un *host* móvil ser su propio agente foráneo, pero ese método sólo funciona si el *host* móvil (en su capacidad de agente foráneo) está conectado lógicamente a Internet en su sitio actual. Incluso, el *host* móvil debe tener la capacidad de adquirir una dirección IP (temporal). Esa dirección IP debe pertenecer a la LAN a que está adjunto actualmente.

¿Cómo se localizan agentes? Cada agente debe transmitir periódicamente su dirección y el tipo de servicios que está dispuesto a proporcionar. Cuando un *host* móvil llega a alguna parte, simplemente puede escuchar estas difusiones, llamadas **anuncios**. Como alternativa, puede difundir un paquete que anuncie su llegada y esperar que el agente foráneo local responda a él.

Otro problema que se tuvo que resolver es qué hacer con los *host* móviles mal educados que se van sin decir adiós. La solución es hacer válido el registro sólo para un intervalo de tiempo fijo. Si no se actualiza periódicamente, queda fuera para que el *host* foráneo pueda limpiar sus tablas.

Otro problema más es la seguridad.

IPv6

Los días del IP en su forma actual (IPv4) están contados, por sus problemas técnicos y capacidad de direcciones. La IETF comenzó a trabajar en 1990 en una versión nueva del IP, con los objetivos:

- ✓ Manejar miles de millones de *hosts*, aunque se desperdicien bits en las direcciones
- ✓ Reducir el tamaño de las tablas de enrutamiento.
- ✓ Simplificar el protocolo, para permitir a los enrutadores el procesamiento más rápido de los paquetes.
- ✓ Proporcionar mayor seguridad (autenticidad y confidencialidad) que el IP actual.
- ✓ Prestar mayor atención al tipo de servicio, especialmente con datos en tiempo real.
- ✓ Ayudar a la multidifusión permitiendo la especificación de alcances.
- ✓ Posibilitar que un *host* sea móvil sin cambiar su dirección.
- ✓ Permitir que el protocolo evolucione.
- ✓ Permitir que el protocolo viejo y el nuevo coexistan por años.

Tras muchos análisis, revisiones e intrigas, se seleccionó una versión modificada de la combinación de las propuestas, llamada ahora **SIPP (Protocolo Simple de Internet Mejorado)**, y se le dio la designación **IPv6**.

El IPv6 cumple los objetivos bastante bien: mantiene las buenas características del IP, descarta y reduce las malas, y agrega nuevas donde se necesitan. En general, IPv6 no es compatible con IPv4, pero es compatible con todos los demás protocolos Internet, a veces con algunas pequeñas modificaciones:

- IPv6 tiene direcciones más grandes que el IPv4; son de 16 bytes de longitud, lo que resuelve el problema que se buscaba resolver
- IPv6 simplificó el encabezado, que contiene sólo 7 campos (contra 13 en el IPv4). Este cambio aumentó la velocidad de enrutamiento
- IPv6 tiene un mejor apoyo de las opciones.
- IPv6 posee un avance importante es la seguridad.
- Se ha puesto mayor atención en la calidad del servicio.

El encabezado principal del IPv6

Versión	Clase de tráfico	Etiqueta de flujo	
Tamaño de carga útil		Siguiente encabezado	Límite de salto
Dirección de origen			
Dirección de destino			

- Versión (6 bits): Permite a los enrutadores ver que tipo de IP se está usando
- Clase de Tráfico (8 bits): Se usa para ver los requisitos de cada paquete, como tiempo real
- Etiqueta de Flujo (20 bits): aún es experimental, pero se usará para permitir a un origen y a un destino establecer una pseudoconexión con propiedades y requisitos particulares. El flujo puede establecerse por adelantado, dándole un identificador. Cuando aparece un paquete con una *Etiqueta de flujo* diferente de cero, todos los enrutadores pueden buscarla en sus tablas internas para ver el tipo de tratamiento especial que requiere. Cada flujo está designado por la dirección de origen, la dirección de destino y el número de flujo.
- Longitud de carga útil (16 bits): Indica cuántos bytes le siguen al encabezado
- Encabezado siguiente (8 bits): La razón por la que pudo simplificarse el encabezado es que puede haber encabezados adicionales (opcionales) de extensión. Este campo indica cuál de los seis encabezados de extensión (actualmente), de haberlos, sigue a éste. Si este encabezado es el último encabezado de IP, el campo de *Encabezado siguiente* indica el manejador de protocolo de transporte (por ejemplo, TCP, UDP) al que se entregará el paquete.
- Límite de saltos (8 bits): Es igual al tiempo de vida en IPv4.
- Dirección Origen (16 bytes) y Dirección Destino (16 bytes): Se ha desarrollado una nueva notación para escribir direcciones de 16 bytes: se escriben como ocho grupos de cuatro dígitos hexadecimales, separados los grupos por dos puntos, como sigue:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Ya que muchas direcciones tendrán muchos ceros en ellas, se han autorizado tres optimizaciones. Primero, los ceros a la izquierda de un grupo pueden omitirse, por lo que 0123 puede escribirse como 123. Segundo, pueden reemplazarse uno o más grupos de 16 ceros por un par de signos de dos puntos. Por tanto, la dirección anterior se vuelve ahora

8000::0123:4567:89AB:CDEF

Por último, las direcciones IPv4 pueden escribirse como un par de signos de dos puntos y un número decimal anterior separado por puntos, como por ejemplo

::192.31.20.46

Compararemos el encabezado de IPv4 con el de IPv6 para ver lo que se ha dejado fuera en IPv6.

- El campo *IHL* se fue porque el encabezado de IPv6 tiene una longitud fija.
- El campo de *Protocolo* se retiró porque el campo *Encabezado siguiente* indica lo que sigue al último encabezado de IP
- Se retiraron todos los campos relacionados con la fragmentación, puesto que el IPv6 tiene un enfoque distinto hacia la fragmentación. Cuando un *host* envía un paquete de IPv6 demasiado grande, en lugar de fragmentarlo, el enrutador que es incapaz de reenviarlo devuelve un mensaje de error. Este mensaje indica al *host* que divida todos los paquetes futuros a ese destino. Es mucho más eficiente hacer que el *host* envíe paquetes del tamaño correcto desde el principio que hacer que los enrutadores los fragmenten sobre la marcha.
- El campo de *Suma de verificación* desaparece, porque su cálculo reduce el desempeño. Con las redes confiables de hoy, además del hecho de que la capa de enlace de datos y las capas de transporte normalmente tienen sus propias sumas de verificación, el provecho de otra suma de verificación no valía el costo de desempeño que generaba.

Encabezados de extensión

Estos encabezados pueden usarse para proporcionar información extra, pero codificada de una manera eficiente. Hay seis tipos de encabezados de extensión definidos actualmente, todos opcionales, pero si hay más de uno, deben aparecer justo después del encabezado fijo, y de preferencia en el orden listado.

Cada elemento está codificado como una tupla (tipo, longitud, valor). El *Tipo* es un campo de 1 byte que indica la opción de la que se trata. Los valores de *Tipo* se han escogido de modo que los dos primeros bits indican a los enrutadores que no saben cómo procesar la opción lo que tienen que hacer. Las posibilidades son: saltar la opción, descartar el paquete, descartar el paquete y enviar de regreso un paquete ICMP, y lo mismo que lo anterior, pero no enviar paquetes ICMP a direcciones de multidifusión (para evitar que un paquete de multidifusión malo genere millones de informes ICMP).

Encabezado de extensión	Descripción
Opciones salto por salto	Información diversa para los enrutadores
Opciones de destino	Información adicional para el destino
Enrutamiento	Ruta total o parcial a seguir
Fragmentación	Manejo de fragmentos de datagramas
Autenticación	Verificación de la identidad del emisor
Carga útil de seguridad encriptada	Información sobre el contenido encriptado

La *Longitud* también es un campo de 1 byte, e indica la longitud del valor (0 a 255 bytes). El *Valor* es cualquier información requerida, de hasta 255 bytes.

Controversias a la hora de definir IPv6

- El argumento sobre la longitud de las direcciones.
- Longitud del campo de *Límite de saltos*.
- Tamaño máximo de paquete. Se llegó a un punto medio: los paquetes normales se limitan a 64 KB, pero puede usarse el encabezado de extensión de salto por salto para permitir los jumbogramas.
- La desaparición de la suma de verificación del IPv4.
- Los *hosts* móviles también fueron tema de contienda. Si una computadora portátil vuela al otro lado del mundo, ¿puede continuar operando en el destino con la misma dirección IPv6, o tiene que usar un esquema con agentes foráneos y agentes de base?
- Seguridad. Todos estaban de acuerdo en que se necesitaba. La guerra fue sobre dónde y cuándo. Se definió en la capa de red, pero las aplicaciones simplemente pueden abstenerse de usar las características de seguridad del IP y encargarse ellas mismas del asunto.
- Muchos países (pero no todos) tienen leyes de exportación estrictas en lo referente a criptografía.

UNIDAD 6: La Capa de Transporte

CAPÍTULO 6: LA CAPA DE TRANSPORTE

La tarea de esta capa es proporcionar un transporte de datos confiable, económico y eficiente de la máquina de origen a la máquina de destino, independientemente de la red o redes físicas en uso. Por lo general sus usuarios son los procesos de la capa de aplicación.

El hardware o software que se encarga del trabajo se llama **entidad de transporte**, la cuál puede estar en el *kernel* del SO, en un proceso de usuario independiente, en un paquete de biblioteca o en la tarjeta de red.

Usaremos las siglas **TPDU (Unidad de Datos del Protocolo de Transporte)** para referirnos a los mensajes enviados de una entidad de transporte a otra. Por lo tanto, las TPDU están contenidas en la carga útil de un paquete, y a su vez éstos en la carga útil de tramas.

Así como hay 2 tipos de servicio de red, el orientado y el no orientado a la conexión, hay 2 tipos de servicio de transporte. En ambos casos, las conexiones tienen 3 fases: establecimiento, transferencia de datos y liberación. El direccionamiento y control de flujo son semejantes en ambas capas.

¿El servicio de la capa de transporte es muy parecido a la capa de red, por qué hay 2 capas? El código de transporte se ejecuta por completo en las máquinas de los usuarios, pero la capa de red, por lo general, se ejecuta en los enrutadores, los cuáles son operados por la empresa portadora. ¿Qué ocurre si la capa de red ofrece un servicio poco confiable? ¿Qué tal si esa capa pierde paquetes con frecuencia? ¿Qué ocurre si los enrutadores se caen de vez en cuando? Problemas...

Los usuarios no tienen control sobre la capa de red, por lo que no pueden resolver los problemas de un mal servicio usando mejores enrutadores o incrementando el manejo de errores en la capa de enlace. La única posibilidad es poner encima de la capa de red otra capa que mejore la calidad de servicio. En esencia, la existencia de la capa de transporte hace posible que el servicio de transporte sea más confiable que el servicio de red subyacente.

Gracias a la capa de transporte es posible escribir programas de aplicación usando un conjunto estándar de primitivas, y que éstos programas funcionen en una amplia variedad de redes sin necesidad de preocuparse por lidiar con diferentes interfaces de subred y transmisiones no confiables.

Las 4 capas inferiores pueden verse como el **proveedor de servicio de transporte**, y las capas superiores son el **usuario del servicio de transporte**.

Primitivas del servicio de transporte Genéricas

Cada servicio de transporte tiene su propia interfaz. El servicio de transporte Orientado a la Conexión (OC) sí es confiable. La esencia de este servicio es la de ocultar las imperfecciones del servicio de red para que los procesos usuarios puedan dar por hecho simplemente la existencia de un flujo de bits libre de errores.

El servicio de transporte, a diferencia del servicio de red, debe ser adecuado y fácil de usar, ya que muchos programas ven las primitivas de transporte. Algunas primitivas pueden ser:

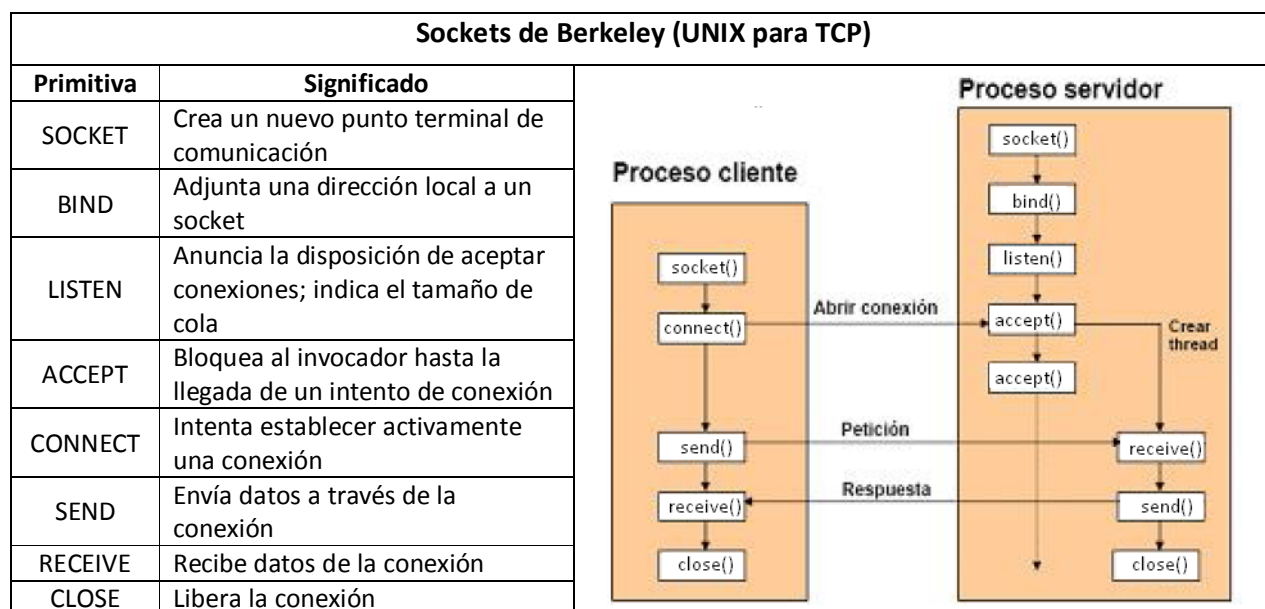
Primitiva	Paquete Enviado	Significado
LISTEN	-	Se bloquea hasta que algún proceso intenta la conexión
CONNECT	CONNECTION REQ.	Intenta activamente establecer una conexión
SEND	DATA	Envía información
RECIEVE	-	Se bloquea hasta que llega un paquete DATA
DISCONNECT	DISCONNECTION REQ.	Este lado quiere liberar la conexión

Para comenzar, el servicio ejecuta una primitiva LISTEN para bloquear al servidor hasta la aparición de un cliente. Cuando un cliente desea comunicarse con el servidor ejecuta la primitiva CONNECT, bloqueando al invocador y enviando una TPDU *CONNECTION REQ* al servidor. Al llegar ésta, la entidad de transporte verifica que el servidor esté bloqueado en LISTEN. A continuación, se desbloquea el servidor y envía una TPDU *CONNECTION ACCEPTED* de regreso al cliente, con lo cual se establece la conexión. Ambas partes pueden ahora intercambiar información usando las primitivas SEND y RECEIVE. Mientras ambos lados puedan llevar el control de quién tiene el turno para transmitir, este esquema funciona bien.

Las confirmaciones de recepción son manejadas por las entidades de transporte usando el protocolo de capa de red, y son transparentes para los usuarios de transporte. De la misma forma, las entidades de transporte necesitarán preocuparse por los temporizadores y retransmisiones.

Cuando ya no se necesita una conexión, debe liberarse. Hay 2 formas de hacer la desconexión:

- Asimétrica: Cualquiera de los 2 usuarios puede emitir una primitiva DISCONNECT, que resulta en el envío de una TPDU *DISCONNECT* a la entidad de transporte remota. A su llegada, se libera la conexión
- Simétrica: Cada parte se cierra por separado, independientemente de la otra. Cuando una de las partes emite DISCONNECT, indica que ya no tiene más datos que enviar, pero aún permite recibir. Una conexión se cierra cuando ambas partes emitieron DISCONNECT.



ELEMENTOS DE LOS PROTOCOLOS DE TRANSPORTE

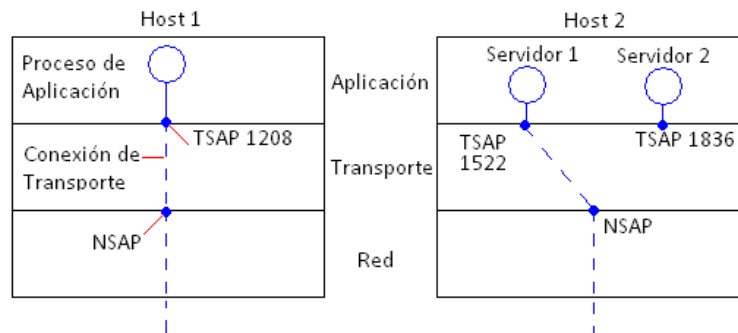
Los protocolos de transporte se parecen a los de capa de enlace de datos. Ambos se encargan del control de errores, secuenciación y control de flujo. Pero tienen diferencias:

- En la capa de enlace de datos, dos enrutadores se comunican directamente mediante un canal físico, mientras que en la capa de transporte hay una subred de por medio.
- En la capa de enlace, el establecimiento de una conexión es sencillo: el otro extremo siempre está ahí. En la capa de transporte, es más complicado
- En la capa de transporte, se debe prever la existencia potencial de capacidad de almacenamiento en la subred. Las consecuencias de esto pueden ser desastrosas

Direccionamiento

Cuando un proceso de aplicación desea establecer una conexión con un proceso de aplicación remoto, debe especificar a cuál se conectará. El método que normalmente se emplea es definir direcciones de transporte en las que los procesos pueden estar a la escucha de solicitudes de conexión. En Internet, a estos puntos se los llama **puertos**. Usaremos el término genérico **TSAP (Punto de Acceso al Servicio de Transporte)**. Los puntos análogos de la capa de red (direcciones) se llaman **NSAP**.

Los TSAP sirven para distinguir los múltiples puntos terminales de transporte que comparten un NSAP.



Protocolo Inicial de Conexión

En lugar de que cada servidor concebible escuche un TSAP bien conocido, cada máquina que desea ofrecer servicio a usuarios remotos tiene un **servidor de procesos** especial que actúa como Proxy de los servidores de menor uso. Los usuarios potenciales de un servicio comienzan por emitir una solicitud CONNECT, especificando la dirección TSAP del servicio que desean. Si no hay ningún servidor esperándolos, consiguen una conexión al servidor de procesos. Tras obtener una solicitud entrante, el servidor de procesos genera el archivo solicitado, permitiéndole heredar la conexión con el usuario existente.

Aunque el protocolo de conexión inicial funciona bien para aquellos servidores que pueden crearse conforme son necesarios, hay muchas situaciones en las que los servicios existen independientemente del servidor de procesos. [Por ejemplo, un servidor de archivos necesita operar un hardware especial y no puede simplemente crearse sobre la marcha cuando alguien quiere comunicarse con él.](#)

Para manejar esta situación, se usa con frecuencia un esquema alternativo, llamado **servidor de nombres o servidor de directorio**. Para encontrar la dirección TSAP correspondiente a un nombre de servicio dado, como "hora del día", el usuario establece una conexión con el servidor de nombres (que escucha un TSAP bien conocido). Entonces el usuario envía un mensaje especificando el nombre del servicio, y el servidor de nombres devuelve la dirección TSAP. Luego el usuario libera la conexión con el servidor, y establece una con el servicio deseado.

En este modelo, al crearse un servicio nuevo, debe registrarse en el servidor de nombres, dando tanto su nombre de servicio como la dirección de su TSAP.

Establecimiento de una Conexión

A primera vista, pareciera suficiente con que una entidad de transporte enviara una TPDU *CONNECTION REQUEST* al destino y esperara una respuesta de *CONNECTION ACCEPTED*. El problema ocurre cuando la red puede esperar, almacenar o duplicar paquetes.

La peor pesadilla posible es la que sigue. Un usuario establece una conexión con un banco, envía mensajes indicando al banco que transfiera una gran cantidad de dinero a la cuenta de una persona no del todo confiable, y entonces libera la conexión. Por mala fortuna, cada paquete de la transacción se duplica y almacena en la subred. Tras liberar la conexión, todos los paquetes salen de la subred y llegan al destino en orden, solicitando al banco que establezca una conexión nueva, transfiera el dinero (nuevamente) y libere la conexión. El banco no tiene manera de saber que son duplicados; debe suponer que ésta es una segunda transacción independiente, y transfiere nuevamente el dinero.

Esto puede atacarse de varias maneras, ninguna de las cuales es muy satisfactoria. Una es usar direcciones de transporte desechables. Otra posibilidad es dar a cada conexión un identificador de conexión y ponerlo en cada TPDU. Por desgracia, este esquema tiene una falla básica: requiere que cada entidad de transporte mantenga una cierta cantidad de información histórica durante un tiempo indefinido. Si se cae una máquina y pierde su memoria, ya no sabrá qué identificadores de conexión usó. Debemos diseñar un mecanismo para matar a los paquetes viejos que aún andan vagando por ahí.

El tiempo de vida de un paquete puede restringirse a un máximo conocido usando una de las siguientes técnicas:

1. Diseño de subred restringido: Incluye cualquier método que evite que los paquetes hagan ciclos, combinado con una manera de limitar el retardo por congestiones a través de la trayectoria más larga posible (ahora conocida).
2. Contador de saltos en cada paquete: Consiste en incrementar el conteo de saltos cada vez que se reenvía el paquete. El protocolo de enlace de datos simplemente descarta cualquier paquete cuyo contador de saltos ha excedido cierto valor.
3. Marca de tiempo en cada paquete: Requiere que cada paquete lleve la hora en la que fue creado, acordando los enrutadores descartar cualquier paquete que haya rebasado cierto tiempo predeterminado. Este último método requiere que los relojes de los enrutadores estén sincronizados, lo que no es una tarea fácil a menos que se logre la sincronización externamente a la red.

En la práctica, necesitaremos garantizar no sólo que el paquete está muerto, sino también que todos los acuses de recibo están muertos, por lo que ahora introduciremos T , que es un múltiplo pequeño del tiempo de vida de paquete máximo verdadero. Si esperamos un tiempo T tras el envío de un paquete, podemos estar seguros de que todos los rastros suyos ya han desaparecido, y que ni él ni sus acuses de recibo aparecerán repentinamente de la nada para complicar el asunto.

Teniendo limitados los tiempos de vida de los paquetes, es posible proponer una manera a prueba de errores de establecer conexiones seguras.

Para resolver el problema de una máquina que pierde toda la memoria acerca de su estado tras una caída, se propuso equipar cada host con un reloj de hora del día. Los relojes de los diferentes hosts no necesitan estar sincronizados. Se supone que el reloj continúa operando aun ante la caída de un host.

La idea básica es asegurar que nunca estén pendientes dos TPDU de número idéntico al mismo tiempo. Cuando se establece una conexión, los k bits de orden menor del reloj se usan como número inicial de secuencia (también k bits). Por tanto cada conexión comienza a numerar sus TPDU con un número de secuencia diferente. El espacio de secuencia también debe ser lo bastante grande para que, al dar la vuelta los números de secuencia, las TPDU viejas con el mismo número de secuencia hayan desaparecido hace mucho tiempo.

Esta relación lineal entre tiempo y números secuenciales iniciales se muestra en la figura

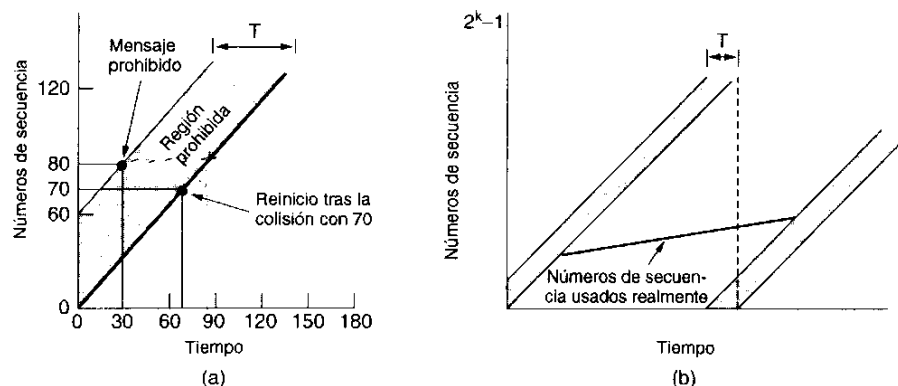


Figura 6-10. (a) Las TPDU no pueden entrar en la zona prohibida. (b) El problema de la resincronización.

Una vez que ambas entidades de transporte han acordado el número de secuencia inicial, puede usarse cualquier protocolo de ventana corrediza para el control de flujo de datos.

Ocurre un problema cuando se cae un *host*. Al regresar nuevamente, sus entidades de transporte no saben dónde estaban en el espacio de secuencias. Una solución es requerir que las entidades de transporte estén inactivas durante T segundos tras una recuperación para permitir que todas las TPDU viejas mueran. Sin embargo, en una interred compleja, T puede ser bastante grande.

Por esto, es necesario introducir una nueva restricción en el uso de números de secuencia. Debemos evitar la asignación de números de secuencia nuevos (es decir, asignados a TPDU nuevas) durante un tiempo T antes de su uso potencial como números iniciales de secuencia. Las combinaciones ilegales de tiempo y número de secuencia se muestran como la **región prohibida** en la figura. Antes de enviar cualquier TPDU por alguna conexión, la entidad de transporte debe leer el reloj y comprobar que no está en la región prohibida.

El protocolo puede meterse en problemas de dos maneras. Si un *host* envía demasiados datos, con demasiada rapidez a través de una conexión recién abierta, la curva de número de secuencia real contra tiempo puede subir con mayor rapidez que la curva de número de secuencia inicial contra tiempo. Esto significa que la tasa de datos máxima en cualquier conexión es de una TPDU por pulso de reloj, y también significa que la entidad de transporte debe esperar hasta que el reloj pulse antes de abrir una nueva conexión tras un reinicio por caída, no sea que se use dos veces el mismo número de secuencia. Ambos puntos son argumentos a favor de un pulso de reloj corto (unos cuantos milisegundos).

Desafortunadamente, el ingreso en la región prohibida desde abajo al enviar con demasiada rapidez no es la única manera de meterse en problemas. Por la figura debe quedar claro que con cualquier tasa de datos menor que la tasa del reloj, la curva de números de secuencia reales usados contra tiempo tarde o temprano entrará en la región prohibida por la izquierda. Cuanto mayor sea la pendiente de la curva de números de secuencia reales, mayor será el retardo de este evento. Como indicamos antes, justo antes de enviar cada TPDU, la entidad de transporte debe comprobar que no está a punto de entrar en la región prohibida; de ser así, debe retardar la TPDU durante T seg o resincronizar los números de secuencia.

El método basado en reloj resuelve el problema del duplicado retrasado de las TPDU de datos, pero para que este método resulte de utilidad, debe establecerse primero una conexión. Dado que las TPDU de control también pueden retrasarse, hay el problema potencial de lograr que ambos lados acuerden en número de secuencia inicial. Supóngase, por ejemplo, que se establecen

conexiones haciendo que el host 1 envíe una TPDU *CONNECTION REQUEST* con el número de secuencia inicial y el número de puerto de destino a un igual remoto, el host 2. El receptor, el host 2, reconoce entonces esta solicitud enviando de regreso una TPDU *CONNECTION ACCEPTED*. Si la TPDU *CONNECTION REQUEST* se pierde, pero aparece con retardo una *CONNECTION REQUEST* duplicada en el host 2, se establecerá incorrectamente la conexión.

Para resolver este problema, Tomlinson desarrolló el **acuerdo de tres vías**. Este protocolo de establecimiento no requiere que ambos lados comiencen a transmitir con el mismo número de secuencia. El host 1 escoge un número de secuencia, x , y envía al host 2 una TPDU *CONNECTION REQUEST* que lo contiene. El host 2 responde con una TPDU *CONNECTION ACCEPTED* reconociendo x y anunciando su propio número de secuencia inicial, y . Por último, el host 1 confirma el número de secuencia inicial del host 2 en la primera TPDU de datos que envía.

Lo importante a notar aquí es que no haya combinación de viejas TPDU que puedan causar la falla del protocolo, permitiendo el establecimiento de una conexión accidentalmente cuando nadie la quiere.

Libерación de una conexión

Recordemos que hay 2 formas de liberar una conexión. La forma **asimétrica** es como la del sistema telefónico: cuando una parte cuelga, se interrumpe la conexión. Trata la conexión como dos conexiones unidireccionales distintas. Esta forma es abrupta y puede resultar en la pérdida de datos. **Por ejemplo, si un host emite DISCONNECT antes de que le llegue una TPDU, se liberará la conexión, pero se perderán datos.**

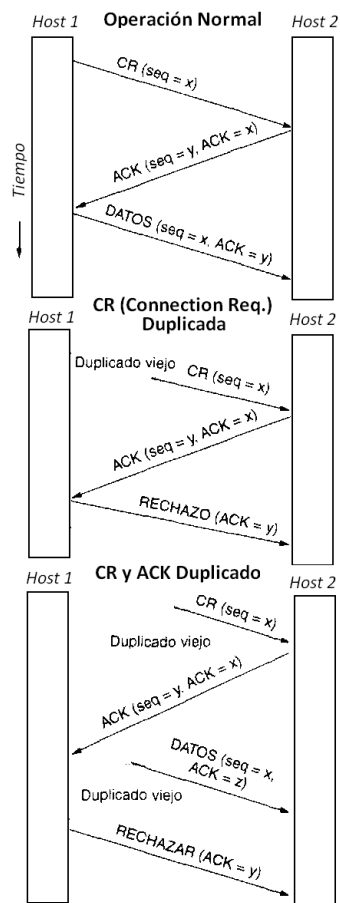
La liberación asimétrica es ideal cuando cada proceso tiene una cantidad fija de datos por enviar y sabe con certidumbre cuándo los ha enviado. En otras situaciones, la determinación de si se ha efectuado o no todo el trabajo y debe terminarse o no la conexión, no es tan obvia. Podríamos pensar en un protocolo en el que el host 1 diga: "Ya terminé. ¿Terminaste también?". Si el host 2 responde "Ya terminé. Adiós", la conexión puede liberarse con seguridad.

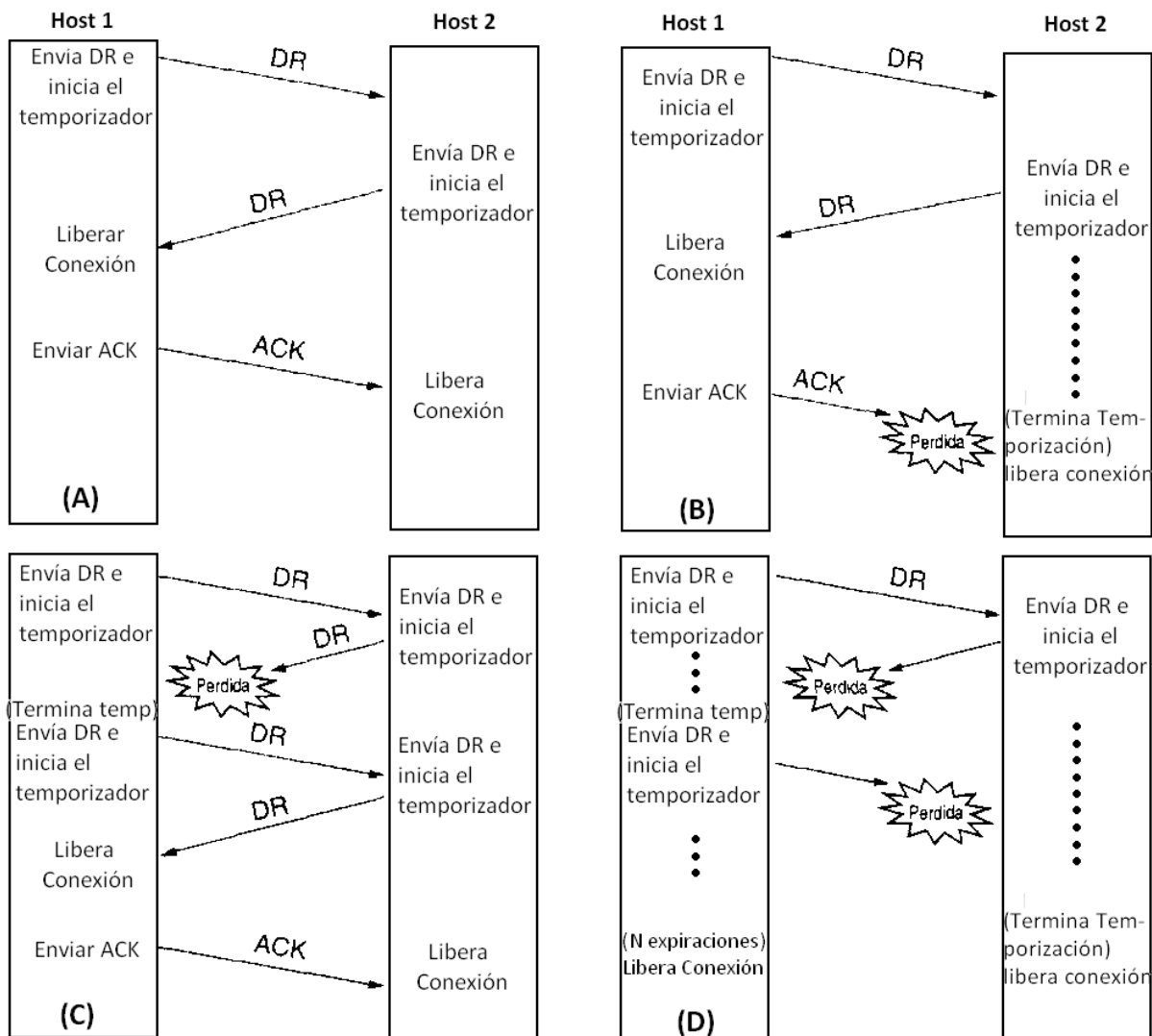
Por desgracia, este protocolo no siempre funciona, debido al **problema de los 2 ejércitos**. Resumiendo este problema: nunca podemos estar seguros de la respuesta. Ninguna de las partes está preparada para desconectarse hasta estar convencida que la otra también lo está.

En la figura podemos ver los 4 escenarios de liberación usando un acuerdo de 3 vías. Aunque este protocolo no es infalible, generalmente es adecuado.

La liberación de una conexión significa que la entidad de transporte remueve la información sobre la conexión de su tabla de conexiones abiertas y avisa de alguna manera al dueño de la conexión (el usuario de transporte). Esta acción es diferente de la emisión de una primitiva *DISCONNECT* por parte de un usuario de transporte.

Nuestra última situación en la figura, tras N reintentos, el emisor se dará por vencido y liberará la conexión. Mientras tanto, expira el temporizador del receptor y también la finaliza.





Aunque este protocolo generalmente es suficiente, en teoría puede fallar si se pierden el DR (*DISCONNECTION REQUEST*) inicial y N retransmisiones. El transmisor se dará por vencido y liberará la conexión, pero el otro lado no sabrá nada sobre los intentos de desconexión y seguirá plenamente activo. Esta situación resulta en una conexión medio abierta.

Pudimos haber evitado este problema no permitiendo que el transmisor se diera por vencido tras N reintentos, sino obligándolo a seguir insistiendo hasta recibir una respuesta. Sin embargo, si se permite que el otro lado termine de temporizar, entonces el transmisor continuará eternamente, pues nunca hará su aparición una respuesta. Si no permitimos que el lado receptor termine la temporización, entonces el protocolo se atora en la figura (b).

Otra manera de cortar la conexión medio abierta es tener una regla que diga que, si no ha llegado ninguna TPDU durante cierta cantidad de segundos, se libera automáticamente la conexión. De esta manera, si un lado llega a desconectarse, el otro lado detectará la falta de actividad y también se desconectará. Por supuesto que si se pone en práctica esta regla, es necesario que cada entidad de transporte tenga un temporizador que se detenga y se reinicie con cada envío de una TPDU. Si expira este temporizador, se transmite una TPDU ficticia, simplemente para evitar que el otro lado se desconecte. Por otra parte, si se usa la regla de desconexión automática y se pierden demasiadas TPDU ficticias una tras otra en una conexión en reposo, primero un lado y luego el otro se desconectarán automáticamente.

Control de Flujo y Almacenamiento en búfer

Veamos ahora la manera en que se manejan las conexiones mientras están en uso. En muchos aspectos es igual que en la capa de enlace de datos, pero en otros es diferente. La similitud básica es que en ambas capas se requiere una ventana corrediza u otro esquema en cada conexión para evitar que un transmisor rápido abrume a un receptor lento. La diferencia principal es que un enrutador por lo regular tiene relativamente pocas líneas, y que un host puede tener numerosas conexiones. Esta diferencia hace que sea impráctico implementar la estrategia de búfers de enlace de datos, en la capa de transporte.

En la capa de enlace de datos, el lado transmisor debe poner en búfer las tramas de salida porque cabe la posibilidad de que tengan que retransmitirse. Si la subred provee un servicio de datagramas, la entidad de transporte emisora también debe manejar búfers, por la misma razón. Si el receptor sabe que el transmisor pone en búfer todas las TPDU hasta que se confirman, el receptor podría o no dedicar búfers específicos a conexiones específicas, según considere necesario. Por ejemplo, el receptor podría mantener un solo grupo de búfers compartidos por todas las conexiones. Cuando entra una TPDU, se hace un intento por adquirir dinámicamente un búfer nuevo. Si hay uno disponible, se acepta la TPDU; de otro modo, se descarta. Dado que el transmisor está preparado para retransmitir las TPDU perdidas por la subred, no hay nada de malo en hacer que el receptor se deshaga de las TPDU, aunque se desperdicien algunos recursos. El emisor simplemente sigue intentando hasta que recibe una confirmación de recepción.

En resumen, si el servicio de red no es confiable, el transmisor debe poner en búfer todas las TPDU enviadas, igual que la capa de enlace de datos. Sin embargo, con un servicio de red confiable son posibles otros arreglos. En particular, si el emisor sabe que el receptor siempre tiene espacio de búfer, no necesita retener copias de las TPDU que envía. Sin embargo, si el receptor no puede garantizar que se aceptará cada TPDU de entrada, el emisor tendrá que usar búfers de todas maneras. En el último caso, el emisor no puede confiar en ACK de la capa de red porque éste sólo significa que ha llegado la TPDU, no que ha sido aceptada.

Todavía queda la cuestión del tamaño de los búfers. Se pueden elegir de tamaño fijo, variable, o dedicar un solo búfer circular grande por conexión.

El equilibrio óptimo entre el almacenamiento en búfer de origen y en el destino depende del tipo de tráfico soportado por la conexión. Para un tráfico de bajo ancho de banda con ráfagas, ([Ej: terminal interactiva](#)), es mejor adquirir los búfers dinámicamente en ambos extremos.

A medida que se abren y cierran conexiones, y a medida que cambia el patrón de tráfico, el emisor y receptor necesitan ajustar dinámicamente sus asignaciones de búfers. Por esto, el protocolo de transporte debe permitir que un host emisor solicite espacio en búfer en el otro extremo. Una manera general de manejar la asignación dinámica de búfers es desacoplarlos de las ACK.

Hasta ahora hemos supuesto tácitamente que el único límite impuesto a la tasa de datos del emisor es la cantidad de espacio de búfer disponible en el receptor. A medida que siga cayendo significativamente el precio de la memoria, podría llegar a ser factible equipar a los hosts con tanta memoria que la falta de búfers dejaría de ser un problema.

Si el espacio de búfer ya no limita el flujo máximo (ya que la memoria está bajando de precio), aparecerá otro cuello de botella: la capacidad de carga de la subred. Se necesita un mecanismo basado en la capacidad de carga de la subred en lugar de la capacidad de buffer del receptor. Este control de flujo debe aplicarse al emisor para evitar que estén pendientes demasiadas TPDU sin confirmación a la vez. Belsnes propuso el uso de un esquema de flujo de ventana corrediza en el que el transmisor ajusta dinámicamente el tamaño de la ventana para igualarla a la capacidad de carga de la red.

Multiplexión

En la capa de transporte puede surgir la necesidad de multiplexión por varias razones.

- Multiplexión hacia arriba: Cuando llega una TPDU, se necesita algún mecanismo para saber a qué proceso asignarla.
- Multiplexión hacia abajo: Supongamos que una subred utiliza circuitos virtuales de manera interna y que impone una tasa de datos máxima a cada uno. Si un usuario necesita más ancho de banda del que puede proporcionar el CV, una alternativa es abrir múltiples conexiones de red y distribuir el tráfico entre ellas.

Recuperación de caídas

Si la entidad de transporte está por entero dentro de los hosts, la recuperación de caídas de red y de enrutador es directa. Si la capa de red proporciona servicio de datagramas, las entidades de transporte esperan la pérdida de algunas TPDU todo el tiempo, y saben cómo manejarla. Si la capa de red proporciona servicio orientado a conexiones, entonces la pérdida de un circuito virtual se maneja estableciendo uno nuevo y sondeando la entidad de transporte remota para saber cuáles TPDU ha recibido y cuáles no. Estas últimas pueden retransmitirse.

Un problema más complicado es la manera de recuperarse de caídas del host. Si se cae el servidor, al regresar, sus tablas se reinician, por lo que ya no sabe con precisión dónde estaba. El servidor podría enviar una TPDU de difusión a todos los demás hosts, anunciando que acaba de caerse y solicitando a sus clientes que informen sobre el estado de todas las conexiones abiertas. Cada cliente puede estar en uno de dos estados: una TPDU pendiente, S1, o ninguna TPDU pendiente, S0. Con base en esta información de estado, el cliente debe decidir si retransmitirá o no la TPDU más reciente. Veamos el funcionamiento del protocolo en la tabla:

Estrategia Usada por el host emisor	Estrategia usada por el host receptor						W: Write C: Caída A: ACK
	Primero ACK, luego escritura			Primero escritura, luego ACK			
	AC(W)	AWC	C(AW)	C(WA)	WAC	WC(A)	
Siempre Retransmitir	Bien	Duplicado	Bien	Bien	Duplicado	Duplicado	
Nunca retransmitir	Perdido	Bien	Perdido	Perdido	Bien	Bien	
Retransmitir en S0	Bien	Duplicado	Perdido	Perdido	Duplicado	Bien	
Retransmitir en S1	Perdido	Bien	Bien	Bien	Bien	Duplicado	

Sin importar cómo se programen el transmisor y el receptor, siempre habrá situaciones en las que el protocolo no podrá recuperarse correctamente. El servidor puede programarse de una de dos maneras: mandar acuse de recibo primero o escribir primero. El cliente puede programarse de cuatro maneras: siempre retransmitir la última TPDU, nunca retransmitir la última TPDU, retransmitir sólo en el estado S0 o retransmitir sólo en el estado S1. Esto da ocho combinaciones pero, como veremos, para cada combinación existe algún grupo de eventos que hacen que falle el protocolo.

Según nuestra regla básica de que no debe haber eventos simultáneos, la caída de un host y su recuperación no pueden hacerse transparentes a las capas superiores.

La recuperación de una caída de capa N sólo puede hacerla la capa N + 1, y sólo si la capa superior retiene suficiente información de estado. Como se mencionó antes, la capa de transporte puede recuperarse de fallas de la capa de red, siempre y cuando cada extremo de una conexión lleve el registro de dónde está.

PROTOCOLOS DE TRANSPORTE DE INTERNET: UDP (PROTOCOLO DE DATAGRAMA DE USUARIO)

Es el protocolo no orientado a la conexión. UDP transmite segmentos que consisten en un encabezado de 8 bytes seguido por la carga útil. El encabezado contiene 4 campos de 16 bits: puerto origen, puerto destino, longitud y suma de verificación.

Los 2 puertos del encabezado sirven para identificar los puntos terminales dentro de las máquinas de origen y destino. Sin los campos de puerto, la capa de transporte no sabría que hacer con el paquete.

UDP no realiza control de flujo, control de errores o retransmisión cuando se recibe un segmento erróneo. Todo lo anterior le corresponde a los procesos de usuario. Lo que sí realiza es proporcionar una interfaz al protocolo IP con la característica agregada de desmultiplexar varios procesos utilizando los puertos. Una aplicación que usa UDP es el DNS.

Llamada a Procedimiento Remoto (RPC)

Cuando un proceso en la máquina 1 llama a uno en la máquina 2, el proceso invocador de la primera se suspende y la ejecución del procedimiento se lleva a cabo en la 2. La información se puede transportar desde el invocador al proceso invocado en los parámetros, y se puede regresar el resultado del procedimiento. Esto es transparente al programador. Esta técnica se conoce como RPC (Llamada a Procedimiento Remoto) y se ha vuelto la base de muchas aplicaciones de redes. Tradicionalmente, el procedimiento invocador se conoce como cliente y el proceso invocado, como servidor, por lo que aquí utilizaremos esos nombres.

El propósito esencial de RPC es hacer que una llamada a procedimiento remoto sea lo más parecida posible a una local. En la forma más simple, para llamar a un procedimiento remoto, el programa cliente debe enlazarse con un pequeño procedimiento de biblioteca, llamado stub del cliente, que representa al procedimiento servidor en el espacio de direcciones del cliente. De forma similar, el servidor se enlaza con una llamada a procedimiento denominada stub del servidor. Estos procedimientos ocultan el hecho de que la llamada a procedimiento desde el cliente al servidor no es local.

El empaquetamiento de los parámetros se conoce como marshaling.

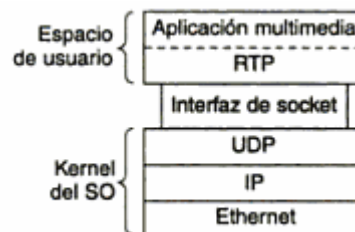
A pesar de la elegancia conceptual de RPC, hay algunas desventajas ocultas. La más grande es el uso de parámetros de apuntador, que son imposibles, ya que el cliente y el servidor están en distintos espacios de direcciones. En algunos casos se pueden usar trucos para hacer que el paso de apuntadores sea posible.

Un segundo problema es el paso de arreglos sin especificar la longitud. Bajo esta circunstancia, es imposible que el stub del cliente aplique marshalling a los parámetros ya que no puede determinar su longitud.

Un tercer problema es que no siempre es posible deducir los tipos de parámetros (Ej: `printf()`)

Un cuarto problema se relaciona con el uso de variables globales compartidas.

RPC se utiliza ampliamente, pero se necesitan algunas restricciones para hacerlo funcionar bien en la práctica. RPC no necesita utilizar paquetes UDP, pero RPC y UDP son una buena combinación y UDP se utiliza comúnmente con RPC.



El protocolo de transporte en tiempo real (RTP)

Se utiliza ampliamente UDP en las aplicaciones multimedia en tiempo real. Gradualmente se volvió más claro que tener un protocolo de transporte genérico para múltiples aplicaciones sería una excelente idea. Y fue por esto que nació el RTP. Se decidió colocarlo en espacio de usuario y ejecutarlo sobre UDP. Debido a este diseño, es un poco difícil decir en cuál capa está RTP. Probablemente la mejor descripción es que es un protocolo de transporte que está implementado en la capa de aplicación.

La función básica de RTP es multiplexar varios flujos de datos en tiempo real en un solo flujo de paquetes UDP. A cada paquete enviado en un flujo RTP se le da un número más grande que a su predecesor. Esta numeración permite al destino determinar si falta algún paquete. Si falta alguno, la mejor acción que el destino puede realizar es aproximar el valor faltante mediante la interpolación. En consecuencia, RTP no tiene control de flujo, control de errores, confirmaciones de recepción ni ningún mecanismo para solicitar retransmisiones.

Para permitir la interconectividad, RTP define diversos perfiles, y para cada perfil se podrían permitir múltiples formatos de codificación.

Otra característica que muchas de las aplicaciones en tiempo real necesitan es la marcación del tiempo (timestamping). La idea aquí es permitir que el origen asocie una marca de tiempo con la primera muestra de cada paquete. Este mecanismo permite que el destino realice una pequeña cantidad de almacenamiento en búfer y reproduzca cada muestra el número exacto de milisegundos después del inicio del flujo, independientemente de cuándo llegó el paquete que contiene la muestra. La marcación del tiempo no sólo reduce los efectos de la fluctuación, sino que también permite que múltiples flujos estén sincronizados entre sí.

RTP tiene un hermano menor llamado **RTCP (Protocolo de Control de Transporte en Tiempo Real)**. Mejora la retroalimentación, sincronización entre flujos, y la interfaz de usuario, pero no transporta ningún tipo de datos. Además, proporciona una forma para nombrar los diversos orígenes. Esta información puede desplegarse en la pantalla del receptor para indicar quién está hablando en ese momento.

PROTOCOLOS DE TRANSPORTE DE INTERNET: TCP (PROTOCOLO DE CONTROL DE TRANSMISIÓN)

Se diseñó específicamente para proporcionar un flujo de bytes confiable de extremo a extremo a través de una interred no confiable (algo que IP no proporciona). Cada máquina que soporta TCP tiene una entidad de transporte TCP, que maneja flujos TCP e interactúa con la capa IP. Una entidad TCP acepta flujos de datos de usuario de procesos locales, los divide en fragmentos que no excedan los 64 KB, y envía cada fragmento como un datagrama IP independiente. Cuando los datagramas que contienen datos TCP llegan a una máquina, se pasan a la entidad TCP, la cual reconstruye los flujos de bytes originales.

El modelo del servicio TCP

El servicio TCP se obtiene al hacer que tanto el servidor como el cliente creen puntos terminales, llamados sockets. Cada socket tiene un número (dirección), que consiste en la dirección IP del host y un puerto. Un puerto es el nombre TCP para un TSAP. Para obtener el servicio TCP, se debe establecer de manera explícita una conexión entre un socket en la máquina emisora y uno en la máquina receptora.

Un socket puede utilizarse para múltiples conexiones al mismo tiempo. Las conexiones se identifican mediante los identificadores de socket de los dos extremos, es decir (socket 1, socket2).

Los números de puerto menores que 1024 se llaman puertos bien conocidos y se reservan para servicios estándar. Ciertamente podría ser posible que el demonio FTP se conecte a sí mismo al puerto 21 en tiempo de arranque, que el demonio telnet se conecte a sí mismo al puerto 23 en tiempo de arranque, y así sucesivamente. Sin embargo, hacer lo anterior podría llenar la memoria con demonios que están inactivos la mayor parte del tiempo. En su lugar, lo que se hace generalmente es que un solo demonio, llamado *inetd* (demonio de Internet) en UNIX, se conecte a sí mismo a múltiples puertos y espere la primera conexión entrante. Cuando eso ocurre, *inetd* bifurca un nuevo proceso y ejecuta el demonio apropiado en él, dejando que ese demonio maneje la solicitud. De esta forma, los demonios distintos a *inetd* sólo están activos cuando hay trabajo para ellos. *Inetd* consulta un archivo de configuración para saber cuál puerto utilizar. En consecuencia, el administrador del sistema puede configurar el sistema para tener demonios permanentes en los puertos más ocupados (por ejemplo, el puerto 80) e *inetd* en los demás.

NOTA: VER ANEXO DE PUERTOS Y PROTOCOLOS

Todas las conexiones TCP son el duplex total y de punto a punto. TCP no soporta la multidifusión ni la difusión.

Una conexión TCP es un flujo de bytes, no de mensajes. Los límites de los mensajes no se preservan de extremo a extremo. El software TCP no tiene idea lo que significan los bytes y no le interesa averiguarlo. Un byte es sólo un byte.

Cuando una aplicación pasa datos a TCP, éste decide si los envía inmediatamente o los almacena en un búfer. Las aplicaciones pueden usar el indicador PUSH, que es una señal para TCP de que no debe retrasar la transmisión.

Una última característica del servicio TCP que vale la pena mencionar son los datos urgentes. Cuando un usuario interactivo oprime las teclas Supr o Ctrl+C para interrumpir una operación remota que ha iniciado, la aplicación emisora coloca información de control en el flujo de datos y se la da a TCP junto con el indicador URGENT. Este evento ocasiona que TCP interrumpa el encolamiento de datos y transmita inmediatamente todo lo que tenga para esa conexión. Cuando el destino recibe los datos urgentes, se interrumpe la aplicación receptora.


El Protocolo TCP

Cada byte de una conexión TCP tiene su propio número de secuencia de 32 bits. La entidad TCP emisora y la receptora intercambian datos en forma de segmentos. Un segmento consiste en un encabezado TCP fijo de 20 bytes seguido de cero o más bytes de datos. El software de TCP decide el tamaño de los segmentos. Hay dos límites que restringen el tamaño de segmento:

- Cada segmento, con encabezado TCP, debe caber en la carga útil de 65,515 bytes del IP
- Cada red tiene una unidad máxima de transferencia (MTU) y cada segmento debe caber en la MTU. En la práctica la MTU es generalmente, de 1500 bytes (el tamaño de la carga útil en Ethernet) y, por tanto, define el límite superior del tamaño de segmento.

El protocolo básico usado por las entidades TCP es el de ventana corrediza de tamaño variable. El campo *tamaño de ventana* en el encabezado indica la cantidad de bytes que pueden enviarse comenzando por el byte cuya recepción se ha confirmado.

Analicemos ahora el encabezado TCP:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Puerto Origen																Puerto Destino															
Número de Secuencia																															
Numero de confirmación de recepción (siguiente byte esperado)																															
Cantidad palabras de 32 bits												URG	ACK	PSH	RST	SYN	FIN	Tamaño de Ventana													
Suma de Verificación																Apuntador Urgente (Desplazamiento a datos urgentes)															
Opciones (0 o más palabras de 32 bits)																															
Datos (opcional)																															

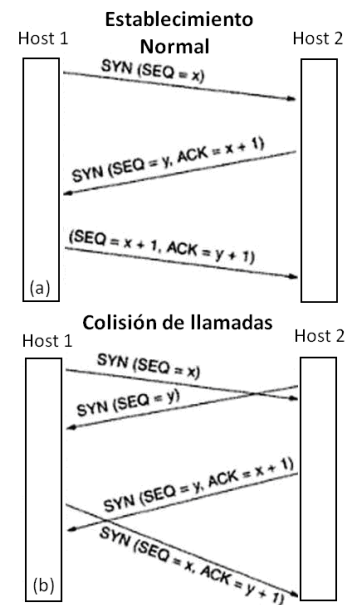
El bit SYN se usa para establecer conexiones. La solicitud de conexión tiene SYN = 1 y ACK = 0 para indicar que el campo de confirmación de recepción no está en unos. La respuesta de conexión sí lleva una confirmación de recepción, por lo que tiene SYN=ACK=1. En esencia, el bit SYN se usa para denotar *CONNECTION REQUEST* y *CONNECTION ACCEPTED*, y el bit ACK sirve para distinguir entre ambas posibilidades.

El campo *opciones* ofrece una forma de agregar características extras no cubiertas por el encabezado normal. La opción más importante es la que permite que cada host especifique la carga útil TCP máxima que está dispuesta a aceptar.

Establecimiento de una conexión TCP

El establecimiento se realiza usando el acuerdo de 3 vías visto anteriormente. La primitiva CONNECT envía un segmento TCP con el bit SYN encendido y el bit ACK apagado, y espera una respuesta. Al llegar el segmento al destino, la entidad TCP ahí revisa si hay un proceso que haya ejecutado un listen en el puerto indicado en el campo de Puerto de destino. Si no lo hay, envía una respuesta con el bit RST encendido para rechazar la conexión. Si algún proceso está escuchando en el puerto se devuelve un segmento de confirmación de recepción.

En el caso en que dos hosts intentan simultáneamente establecer una conexión entre los mismos dos sockets, la secuencia de eventos es la que se ilustra en la figura (b). El resultado de estos eventos es que sólo se establece una conexión, no dos, pues las conexiones se identifican por sus puntos terminales.



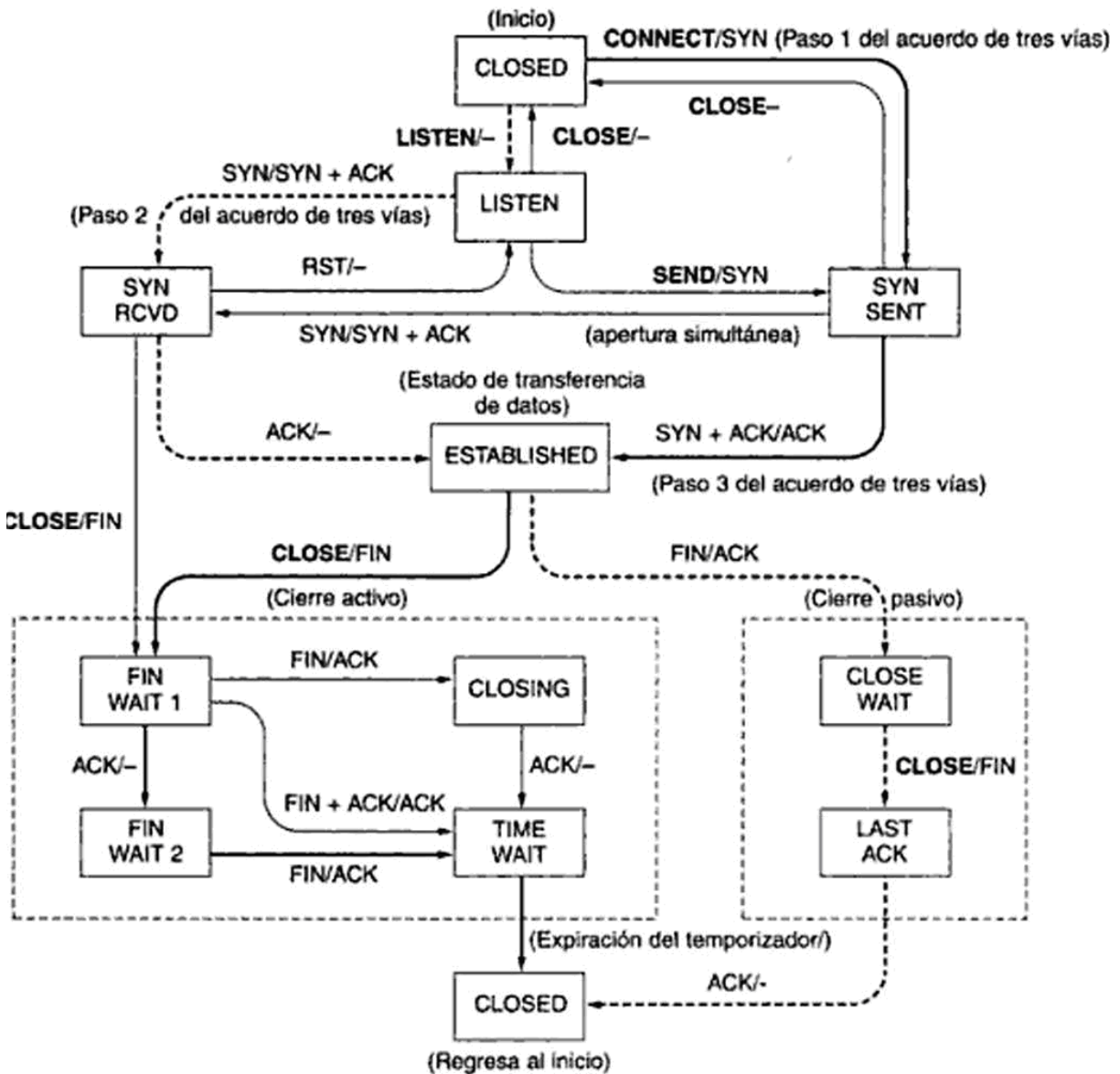
Liberación de una conexión TCP

Aunque las conexiones TCP son duplex total, para entender la manera en que se liberan las conexiones es mejor visualizarlas como un par de conexiones simples. Para liberarlas, cualquier parte puede enviar a la otra un segmento TCP con el bit FIN en 1. Al confirmarse la recepción de FIN, ese sentido se apaga. Sin embargo, puede continuar un flujo de datos indefinido en el otro sentido. Cuando ambos sentidos se han apagado, se libera la conexión.

Normalmente, se requiere 4 segmentos TCP para liberar una conexión, un FIN y un ACK en cada sentido. Pero es posible incluir el primer ACK en el segundo FIN, reduciendo la cuenta a 3.

En esencia, no hay diferencia entre la liberación secuencial o simultánea por parte de los hosts. Para evitar el problema de los 2 ejércitos, se usan temporizadores

Máquina de estados finitos de Administración de Conexiones TCP



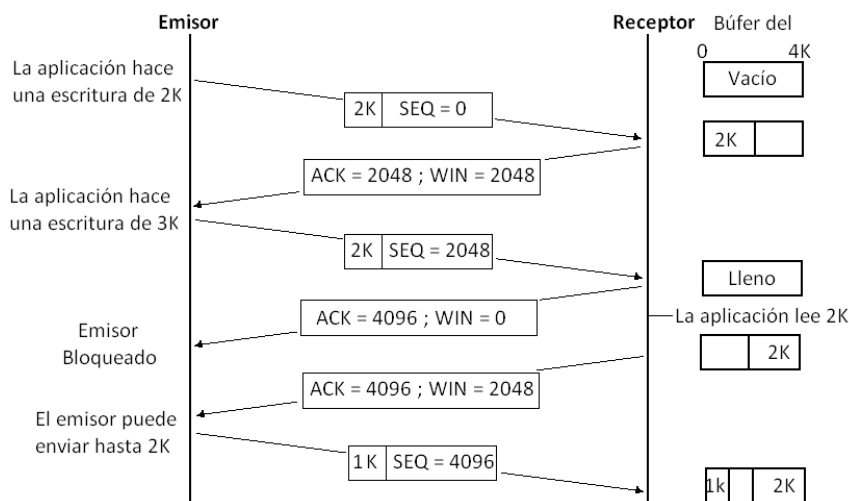
La línea continua gruesa en la trayectoria normal de un cliente

La línea punteada gruesa es la trayectoria normal de un servidor

Las líneas delgadas son eventos poco comunes

Cada transacción está indicada como evento/acción

Política de Transmisión de TCP: manejo de búfers



Cuando la ventana es de 0, el emisor normalmente no puede enviar segmentos, salvo en 2 situaciones. Una es cuando hay datos urgentes. La otra situación sucede porque el emisor puede enviar un segmento de 1 byte para hacer que el receptor anuncie el siguiente byte esperado y el tamaño de la ventana. El estándar TCP proporciona explícitamente esta opción para evitar un bloqueo irreversible si llega a perderse un anuncio de ventana.

No se requiere que los emisores envíen datos tan pronto como llegan de la aplicación. Tampoco se requiere que los receptores envíen confirmaciones de recepción tan pronto como sea posible.

Neagle sugirió que al llegar datos al emisor un byte a la vez, simplemente se envíe el primer byte y se almacenen los demás en búfer hasta la confirmación de recepción del byte pendiente, y recién ahí se envíen los demás.

Otro problema que puede arruinar el desempeño de TCP es el **síndrome de ventana tonta**, que ocurre cuando se pasan datos a la entidad emisora en bloques grandes, pero una aplicación interactiva del lado receptor lee datos a razón de 1 byte a la vez. Inicialmente, el búfer TCP del lado receptor está lleno y el emisor lo sabe (es decir, tiene un tamaño de ventana de 0). Entonces la aplicación interactiva lee un carácter del flujo TCP. Esta acción hace feliz al TCP receptor, por lo que envía una actualización de ventana al emisor indicando que está bien que envíe 1 byte. El emisor accede y envía 1 byte. El búfer ahora está lleno, por lo que el receptor confirma la recepción del segmento de 1 byte pero establece la ventana en 0. Este comportamiento puede continuar indefinidamente.

La solución de Clark es evitar que el receptor envíe una actualización de ventana para 1 byte. En cambio, se le obliga a esperar hasta tener disponible una cantidad de espacio, y luego lo anuncia. Específicamente, el receptor no debe enviar una actualización de ventana hasta que pueda manejar el tamaño máximo de segmento que anunció al establecerse la conexión, o que su búfer quede a la mitad de capacidad, lo que sea más pequeño. Además, el emisor también puede ayudar al no enviar segmentos muy pequeños.

El algoritmo de Nagle y la solución de Clark al síndrome de ventana tonta son complementarios. La meta es que el emisor no envíe segmentos pequeños y que el receptor no los pida.

El TCP receptor también puede hacer más para mejorar el desempeño que simplemente actualizar ventanas en unidades grandes.

Otro problema del receptor es qué debe hacer con los segmentos fuera de orden. Pueden conservarse o descartarse, al albedrío del receptor.

Control de congestión en TCP

Cuando la carga ofrecida a cualquier red es mayor que la que puede manejar, se genera una congestión. Aunque la capa de red también intenta manejarlos, gran parte del trabajo pesado recae sobre el TCP porque la solución real a la congestión es la disminución de la tasa de datos.

En teoría, puede manejarse la congestión aprovechando un principio de física: la ley de conservación de los paquetes. La idea es no inyectar un paquete nuevo en la red hasta que salga uno viejo (es decir, se entregue). El TCP intenta lograr esta meta manipulando dinámicamente el tamaño de la ventana.

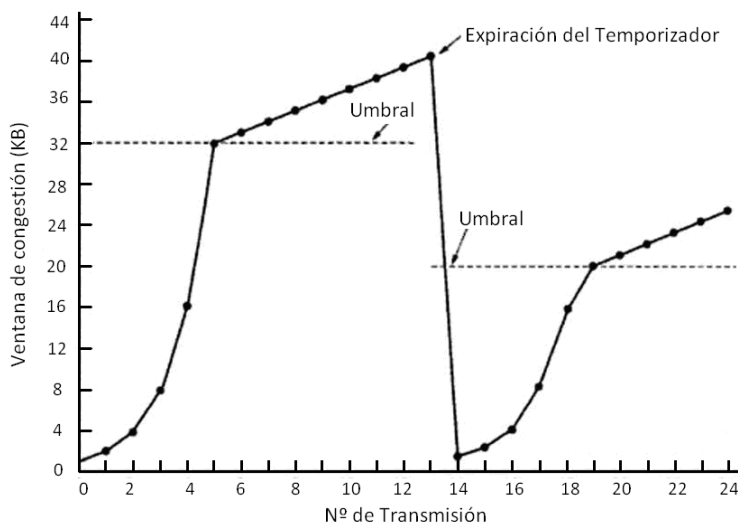
El primer paso del manejo de la congestión es su detección. Hoy día, la pérdida de paquetes por errores de transmisión es relativamente rara debido a que las troncales de larga distancia son de fibra (aunque las redes inalámbricas son otra historia). En consecuencia, la mayoría de las expiraciones de tiempo en Internet se deben a congestión.

Explicaremos lo que TCP hace para evitar que ocurra. Al establecerse una conexión, se tiene que seleccionar un tamaño de ventana adecuado. Aún así, pueden ocurrir desbordamientos debido a congestión interna en la red.

La solución de Internet es aceptar que existen dos problemas potenciales (capacidad de la red y capacidad del receptor) y manejarlos por separado. Para ello, cada emisor mantiene dos ventanas: la ventana que ha otorgado el receptor y una segunda ventana, la ventana de congestión. Cada una refleja la cantidad de bytes que puede enviar el emisor. La cantidad de bytes que pueden enviarse es la cifra menor de las dos ventanas.

Al establecer una conexión, asigna una ventana de tamaño 2^X segmentos, y envía esa cantidad de segmentos. Si recibe todos los ACK, aumenta X en 1, y repite el proceso. La ventana de congestión sigue creciendo exponencialmente hasta ocurrir una expiración del temporizador o alcanzar el tamaño de la ventana receptora. (Ej: [las ráfagas de 1024 y 2048 funcionaron bien, pero la de 4096 no; usamos 2048 como tamaño de ventana](#)). Este algoritmo, **arranque lento**, lo maneja TCP.

El algoritmo de control de congestión de Internet usa un 3º parámetro, llamado **umbral**, además de las ventanas de congestión y recepción. Al ocurrir una expiración del temporizador, se



establece el umbral en la mitad de la ventana de congestión actual, y la ventana de congestión se reestablece a un segmento máximo. Luego se usa el arranque lento para determinar lo que puede manejar la red, excepto que el crecimiento exponencial termina al alcanzar el umbral. A partir de ese punto, las transmisiones exitosas aumentan linealmente la ventana de congestión. Este algoritmo supone que probablemente es aceptable recortar la ventana de congestión a la mitad y luego

aumentar gradualmente desde allí.

Si no ocurren más expiraciones de temporizador, la ventana de congestión continuará creciendo hasta el tamaño de la ventana del receptor, donde permanecerá constante mientras no ocurran expiraciones del temporizador y la ventana del receptor no cambie de tamaño.

Administración de Temporizadores del TCP

El TCP usa varios temporizadores (al menos conceptualmente) para hacer su trabajo. El más importante de estos es el temporizador de retransmisión. Al enviarse un segmento, surge la pregunta: ¿qué tan grande debe ser el intervalo de expiración del temporizador? La solución es usar un algoritmo muy dinámico que ajuste constantemente el intervalo de expiración del temporizador, con base en mediciones continuas del desempeño de la red.

El temporizador de retransmisiones no es el único usado por el TCP. El segundo temporizador es el temporizador de persistencia, diseñado para evitar el siguiente bloqueo irreversible. El receptor envía una confirmación de recepción con un tamaño de ventana de 0, indicando al emisor que espere. Después, el receptor actualiza la ventana, pero se pierde el paquete con la actualización. Ahora, tanto el emisor como el receptor están esperando que el otro haga algo. Cuando termina el temporizador de persistencia, el emisor envía un sondeo al receptor. La respuesta al sondeo da el tamaño de la ventana. Si aún es de cero, se inicia el temporizador de persistencia nuevamente y se repite el ciclo. Si es diferente de cero, pueden enviarse datos.

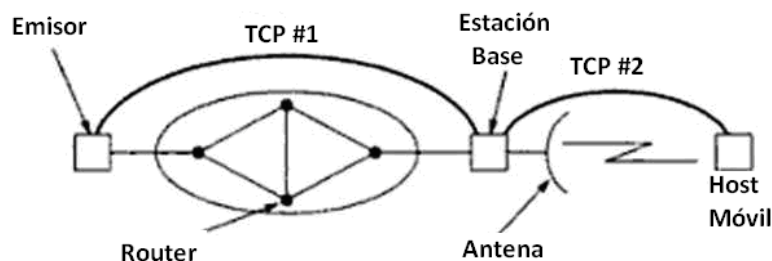
Un tercer temporizador usado en algunas implementaciones es el **temporizador de seguir con vida** (*KEEPALIVE TIMER*). Cuando una conexión ha estado inactiva durante demasiado tiempo, el temporizador de seguir con vida puede expirar, haciendo que un lado compruebe que el otro aún está ahí. Si no se recibe respuesta, se termina la conexión. El último temporizador usado en cada conexión TCP es el que se usa en el estado TIMED WAIT durante el cierre; opera durante el doble del tiempo máximo de vida de paquete para asegurar que, al cerrarse una conexión, todos los paquetes creados por ella hayan desaparecido.

TCP y UDP Inalámbricos

En teoría, los protocolos de transporte deben ser independientes de la tecnología de la capa de red subyacente. En la práctica sí importa, ya que la mayoría de las implementaciones de TCP han sido optimizadas cuidadosamente con base en supuestos que se cumplen en las redes alámbricas, pero no en las inalámbricas. El problema principal es el algoritmo de control de congestión. Hoy casi todas las implementaciones de TCP suponen que las expiraciones del temporizador ocurren por congestionamientos, no por paquetes perdidos. En consecuencia, al expirar un temporizador, el TCP disminuye su velocidad y envía con menor ímpetu.

Desgraciadamente, los enlaces de transmisión inalámbrica son muy poco confiables. Pierden paquetes todo el tiempo. El enfoque adecuado para el manejo de paquetes perdidos es reenviarlos tan pronto como sea posible. Reducir la velocidad simplemente empeora las cosas.

Con frecuencia, la trayectoria del emisor al receptor no es homogénea. Ahora es más difícil la decisión correcta en el caso de una expiración del temporizador, ya que es importante saber dónde ocurrió el problema. Una solución propuesta por Bakne y Badrinath, el **TCP Indirecto**, es la división de la conexión TCP en dos conexiones distintas. La primera va del emisor a la estación base. La segunda va de la estación base al receptor. La estación base simplemente copia paquetes entre las conexiones en ambas direcciones.



La ventaja de este esquema es que ahora ambas conexiones son homogéneas. La desventaja es que se viola por completo la semántica del TCP.

Una solución diferente, debido a Balakrishnan, no quebranta la semántica del TCP. Funciona haciendo varias modificaciones pequeñas al código de la capa de red de la estación base. Uno de los cambios es la adición de un agente espía que observa y almacena en caché los segmentos TCP que van al host móvil y las confirmaciones de recepción que regresan de él.

Sin embargo, una desventaja de esta transparencia es que, si el enlace inalámbrico tiene muchas pérdidas, el temporizador del transmisor podría expirar esperando una confirmación de recepción e invocar el algoritmo de control de congestión. Con en TCP indirecto, el algoritmo de control de congestión nunca iniciará hasta que realmente haya congestión en la parte alámbrica de la red.

El documento de Balakrishnan, también sugiere una solución al problema de segmentos perdidos que se originan en el host móvil. Al notar una estación base un hueco en los números de secuencia de entrada, genera una solicitud de repetición selectiva de los bytes faltantes con una opción TCP.

Gracias a estos dos mecanismos, el enlace inalámbrico se hace más confiable en ambas direcciones, sin que el origen lo sepa y sin cambiar la semántica del TCP.

Si bien el UDP no tiene los mismos problemas que el TCP, la comunicación inalámbrica también le produce dificultades. El problema principal es que los programas usan el UDP pensando que es altamente confiable.

TCP para Transacciones (Similar a UDP para RCP)

Al inicio de este capítulo vimos las RPC como una forma de implementar sistemas cliente-servidor. Si tanto la solicitud como la respuesta son lo suficientemente pequeñas como para caber en paquetes sencillos y la operación tiene la misma potencia, conviene UDP. Si estas condiciones no se cumplen, conviene TCP. Para realizar RCP con TCP se requieren por lo menos 9 mensajes en el mejor de los casos.

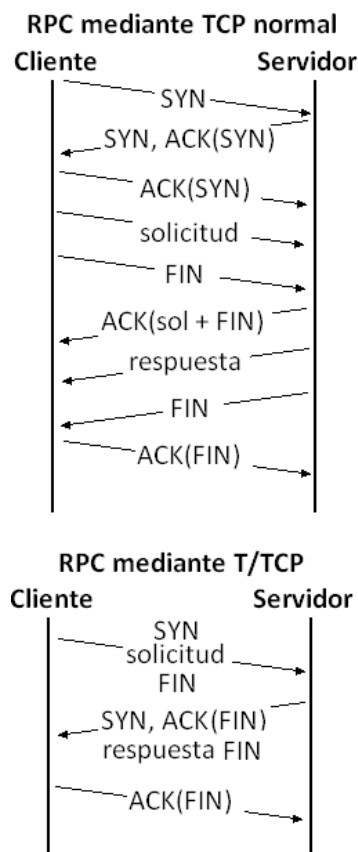
Con esto surge rápidamente la pregunta de si hay alguna forma para combinar la eficiencia de RPC utilizando UDP (sólo dos mensajes) con la contabilidad de TCP. Puede hacerse con una variante TCP experimental llamada **T/TCP (TCP para Transacciones)**.

El primer paquete del cliente contiene el bit SYN, la solicitud misma y el paquete FIN. Lo que dice es: Deseo establecer una conexión, aquí están los datos, y con esto termino.

Cuando el servidor obtiene la solicitud, busca o calcula la respuesta y elige cómo responder. Si la respuesta se ajusta en un paquete, da la respuesta que dice: "Confirmando la recepción de tu paquete FIN, aquí está la respuesta, y con esto termino". A continuación el cliente confirma la recepción del paquete FIN del servidor, y el protocolo termina en tres mensajes.

Sin embargo, si el resultado es de más de un paquete, el servidor también tiene la opción de no encender el bit FIN, en cuyo caso puede enviar múltiples paquetes antes de cerrar su dirección.

Otra propuesta es **SCTP (Protocolo de Transmisión de Control de Flujo)**.



ASPECTOS DEL DESEMPEÑO

En el entendimiento del desempeño de las redes, muy poca teoría tiene en realidad alguna utilidad en la práctica.

Llamaremos en esta sección paquete al paquete de capa de red junto al TPDU encapsulado en él. Analizaremos 5 aspectos del desempeño de las redes:

1. Problemas de Desempeño

- a. Congestión: Causada por sobrecargas temporales de los recursos.
- b. Desequilibrio estructural de recursos: Ej: [línea de Gigabits conectada a un Pentium 1](#)
- c. Sobrecarga síncrona: Si se difundiera una TPDU errónea a 10000 PC, cada una podría devolver un mensaje de error. Se generaría una **tormenta de difusión** que podría paralizar la red. Ej: [Se corta la luz, y miles de PC llaman al servidor DHCP: este se cuelga](#).
- d. Falta de Afinación del Sistema: Ej: [si no se ha asignado suficiente memoria como espacio de búfer, ocurrirán desbordamientos y se perderán varias TPDU](#). Otro asunto relativo al afinamiento es el establecimiento correcto de los temporizadores.
- e. Protocolos antiguos en líneas de Gigabits
- f. Capacidad del canal: Para lograr un buen desempeño, la ventana del receptor debe tener cuando menos el producto ancho de banda-retardo, y de preferencia ser un poco más grande, puesto que el receptor podría no responder instantáneamente.
- g. Fluctuación: Hay problemas en aplicaciones de tiempo real

2. Medición del Desempeño de una red: Los pasos del mismo son:

- 1) Medir los parámetros pertinentes y el desempeño de la red
- 2) Tratar de entender lo que ocurre, teniendo en cuenta los siguientes postulados:
 - ✓ Asegurarse que el tamaño de la muestra es lo bastante grande y representativo
 - ✓ Tener cuidado de usar relojes de intervalos grandes
 - ✓ Asegurarse que no ocurre nada inesperado durante las pruebas
 - ✓ El caché y búfers pueden arruinar las mediciones
 - ✓ Entender lo que se está midiendo
 - ✓ Tener cuidado con la extrapolación de resultados
- 3) Cambiar un parámetro

3. Diseño de sistemas con mejor desempeño

Una red mal diseñada puede mejorarse sólo hasta un límite. Más allá, tiene que rehacerse. Algunas reglas a tener en cuenta son:

- La velocidad de CPU es más importante que la velocidad de la red. La sobrecarga de los SO y protocolos domina el tiempo de utilización en el alambre
- Reducir el nº de paquetes para reducir la sobrecarga de software. A más paquetes, más tamaño se ocupa en encabezados
- Reducir al mínimo las conmutaciones de contexto: Éstas provocan que se guarden los datos temporalmente en búfers internos.
- Reducir al mínimo las copias de TPDU en el sistema
- Es posible comprar más ancho de banda pero no un retardo menor
- Evitar la congestión y expiraciones de temporizador

4. Procesamiento rápido de las TPDU's

La idea es hacer predicción de encabezado. Consiste en hacer una comprobación rápida para ver si el encabezado es el esperado. Mejora notablemente el desempeño

5. Protocolos para redes futuras de alto desempeño

La primera reacción al aparecer una nueva red, es intentar usar sobre ella los viejos protocolos. Esto provoca varios problemas:

- 1) Tamaño del nº de secuencia. Muchas redes usan nº de secuencia de 32 bits. En una Ethernet de 1 Gbps, el tiempo para que den vuelta los nº de secuencia es de 34 segundos, bastante bajo contra los 120 segundos de vida de un paquete en Internet
- 2) Hay menos tiempo disponible para el procesamiento de los protocolos del que había antes, por lo que los protocolos deben simplificarse
- 3) El protocolo de retroceso N se desempeña mal en las líneas con un producto ancho de banda-retardo grande
- 4) Las líneas de gigabits son diferentes de las de megabits, ya que están limitadas por el retardo en lugar del ancho de banda
- 5) En muchas aplicaciones de gigabits, como la multimedia, la variación en los tiempos de llegada de los paquetes es tan importante como el retardo medio mismo. Una tasa de entrega lenta pero uniforme con frecuencia es preferible a una rápida pero variable.

"Se debe diseñar pensando en la velocidad, no en la optimización del ancho de banda"

Una manera de acelerar el procedimiento es construir interfaces de red rápidas en hardware. Lo malo de esto es que, a menos que el protocolo sea excesivamente sencillo, "hardware" simplemente significa una tarjeta con una 2ª CPU y su propio programa.

Veamos ahora el asunto de la realimentación en los protocolos de alta velocidad. Debido al ciclo de retardo (relativamente) largo, debe evitarse la retroalimentación: la señalización del receptor al emisor tarda demasiado.

En pocas palabras, la elevación de las velocidades inevitablemente empuja el diseño hacia la operación orientada a la conexión, o algo similar. Por supuesto, si el ancho de banda se incrementa en el futuro de forma tal que no importe desperdiciarlo, las reglas cambiarán.

La disposición de los paquetes es una consideración importante en redes de gigabits. El encabezado debería contener la menor cantidad posible de campos, para no desperdiciar tiempo en procesamiento. El encabezado y datos deben tener sumas de verificación aparte, para poder comprobar que el encabezado esté correcto antes de analizar datos.

El tamaño máximo de los datos debe ser lo bastante grande para permitir una operación eficiente inclusive ante retardos largos.

Otra característica valiosa es la capacidad de enviar una cantidad normal de datos junto con la solicitud de conexión. Así, puede ahorrarse el tiempo de un viaje de ida y vuelta.

Sobre el software de los protocolos, una idea clave es concentrarse en el caso exitoso. Para lograr que los protocolos operen rápidamente, el diseñador debería enfocarse en reducir al mínimo el tiempo de procesamiento cuando todo funciona bien. La reducción al mínimo del tiempo de procesamiento cuando ocurren errores es secundaria. Un segundo asunto relacionado con el software es la minimización del tiempo de copiado.

UNIDAD 7: La Capa de Aplicación

DNS (SISTEMA DE NOMBRES DE DOMINIO)

Para los nombres de recursos se introdujeron los nombres ASCII con el fin de separar los nombres de máquina de las direcciones de máquina. La esencia de DNS es la invención de un esquema de nombres jerárquico basado en dominios y un sistema de BD Distribuido para implementar este esquema de nombres.

Conceptualmente, Internet se divide en 200 **dominios** de nivel superior ([gov](#), [edu](#), [org](#),...), cada uno de los cuales abarca muchos hosts. Cada dominio se divide en subdominios, los cuales a su vez se dividen, y así sucesivamente (Ej: [edu.yale](#)). Todos estos dominios pueden representarse mediante un árbol.

Las hojas del árbol representan los dominios sin subdominios. Un dominio de hoja puede tener un solo host, o miles. Cada dominio se nombra por la ruta hacia arriba desde él hasta la raíz (sin nombre). Los componentes se separan por puntos y no distinguen mayúsculas o minúsculas.

Los dominios de nivel superior se dividen en dos categorías: genéricos ([edu](#), [gov](#), [org](#), [etc.](#)) y de país ([ar](#), [br](#), [au](#), [etc.](#)). En general, obtener un dominio de segundo nivel (Ej: [mi-empresa.com](#)) es fácil. Simplemente se necesita ir con el registrador de dominio de nivel superior correspondiente (com, en este caso) para ver si el nombre está disponible y si no es una marca registrada de alguien más. Si no hay problema, el solicitante paga una cuota anual y obtiene el nombre.

No hay ninguna regla que impida registrarse bajo 2 dominios de nivel superior, pero pocas organizaciones lo hacen, excepto las multinacionales (Ej: [Sony.com](#) y [Sony.nl](#)). Cada dominio controla como se asignan los dominios que hay bajo él. Ej: [Japón tiene los dominios ac.jp y co.jp, que son los espejos de edu y com.](#)

Cada dominio puede tener un grupo de **registros de recursos** asociados a él. Por lo tanto, la función real del DNS es relacionar los dominios de nombres con los registros de recursos. Un registro de recursos tiene 5 tuplas: *Nombre_dominio*, *tiempo_vida*, *clase*, *tipo* y *valor*.

En teoría, un solo servidor de nombres podría contener toda la base de datos DNS y responder a todas las consultas dirigidas a ella. Para evitar los problemas asociados a tener una sola fuente de información, el espacio de nombres DNS se divide en **zonas** no traslapantes, donde cada zona contiene una parte del árbol.

Un **registro autorizado** es uno que proviene de la autoridad que administra el registro y, por lo tanto, siempre es correcto. Los registros autorizados contrastan con los registros en caché, que podrían no estar actualizados.

Si el dominio es remoto y no hay información disponible localmente sobre el dominio solicitado, el DNS envía un mensaje de consulta al DNS de nivel superior en el que le solicita dicho dominio.

Una vez que estos registros regresan al servidor de nombres, se almacenan en caché, por si se necesitan posteriormente. Sin embargo, esta información no es autorizada, puesto que los cambios hechos en [cs.yale.edu](#) no se propagarán a todos los caché del mundo que puedan saber sobre ella. Por esta razón, las entradas de cache no deben vivir demasiado tiempo. Ésta es la razón por la cual el campo *Tiempo_vida* se incluye en cada registro de recursos.

El método descrito anteriormente se conoce como **consulta recursiva**.

Vale la pena indicar que cuando un cliente DNS no recibe una respuesta antes de que termine su temporizador, probará la siguiente vez con otro servidor. DNS no ayuda a localizar personas, recursos, servicios u otros objetos en general. Para esto, se usa **LDAP (Protocolo ligero de acceso al directorio)**

CORREO ELECTRÓNICO

Los primeros sistemas de e-mail consistían simplemente en protocolos de transferencia de archivos, con la convención de que la primera línea de cada mensaje contenía la dirección del destinatario. Los problemas de este sistema eran:

- × El envío de un mensaje a un grupo de personas era laborioso
- × Los mensajes no tenían estructura interna
- × El remitente no sabía si había llegado o no
- × Si alguien quería encomendar a otro el manejo de su correo, era muy complicado
- × La GUI estaba mal integrada al sistema de transmisión
- × No era posible crear y enviar mensajes que contuvieran una mezcla de texto, fotos, voz.

Arquitectura y Servicios

Los sistemas de e-mail consisten en 2 subsistemas:

- Agentes de Usuario: Normalmente es un programa, que acepta una gran variedad de comandos para redactar, recibir y contestar los mensajes y manipular buzones de correo.
 - *Envío de un e-mail*: el usuario debe proporcionar el mensaje, la dirección de destino, y posiblemente algunos otros parámetros. Muchos agentes de usuario esperan direcciones de la forma *usuario@direcciónDNS*, aunque existen muchas otras formas de direccionamiento.
 - *Lectura del e-mail*: cuando se inicia un agente de usuario, este buscará en el buzón del usuario el e-mail recibido
- Agentes de Transferencia: Mueven los mensajes de origen al destino. Son por lo común demonios del sistema que operan en segundo plano.

Por lo general, los sistemas de correo electrónico desempeñan 5 funciones básicas:

- 1) Redacción: Se refiere al proceso de crear mensajes y respuestas
- 2) Transferencia: Se refiere a mover mensajes del remitente al destinatario
- 3) Generación del Informe: ¿Se entregó, se rechazó, se perdió?
- 4) Visualización: Cómo se muestran los mensajes
- 5) Disposición: Tiene que ver con lo que el destinatario hace con el mensaje recibido

La mayoría de los sistemas de e-mail actuales proporcionan una gran variedad de características avanzadas, como **buzón de correo** para almacenar el correo entrante, **listas de correo**, copias ocultas, prioridad, destinatarios alternos, etc.

Un detalle: si enviamos un correo a una lista, y ésta se mantiene localmente, los correos se multiplicarán localmente; si se mantiene en el servidor, se multiplicarán de manera remota.

Una idea clave en todos los sistemas modernos de e-mail es la distinción entre el **sobre** (encabezado+cuerpo) y el contenido (mensaje). El sobre es solo para el agente de transferencia.

Formatos del Mensaje

Un formato es el **RFC-822**, en el cual cada campo del encabezado consiste en una sola línea de texto ASCII que contiene el nombre del campo, dos puntos y un valor (en la mayoría de los casos). Es un estándar viejo y no distingue claramente los campos del sobre y encabezado.

Encabezado	Significado
TO:	Direcciones de e-mail de los destinatarios primarios
CC:	Direcciones de e-mail de los destinatarios secundarios
CCO:	Direcciones de e-mail para las copias ocultas
FROM:	Persona o Personas que crearon el mensaje
SENDER:	Dirección de e-mail del remitente
RECEIVED:	Línea agregada por cada agente de transferencia en la ruta
RETURN-PATH:	Puede usarse para identificar una ruta de regreso al remitente

Además de los campos anteriores, también puede contener una gran variedad de campos de encabezado usados por los agentes de usuario o los destinatarios. Ej: [Date](#), [Reply-to](#), [Message-ID](#), [In-Reply-to](#), [References](#), [Keywords](#), [Subjet](#), etc. Tras los encabezados de mensaje, viene el cuerpo del mensaje. Los usuarios pueden poner aquí lo que quieran.

En los primeros días de ARPANET, el correo electrónico consistía solamente en mensajes en inglés y ASCII. Habían problemas de envío y recepción de mensajes con acento, alfabetos no latinos, alfabetos asiáticos y mensajes que no contenían texto (Ej: [archivo](#)). Se propuso una solución, llamada **MIME (Extensiones Multipropósito de Correo Internet)**, que propone continuar usando el formato RFC-822, pero agregar una estructura al cuerpo del mensaje y definir reglas de codificación para los mensajes no ASCII. MIME define 5 nuevos encabezados de mensaje:

Encabezado	Significado
MIME-Version	Indica la versión de MIME
Content-Description	Cadena de texto que describe el contenido
Content-ID	Identificador único
Content-Transfer-Encoding	Cómo se envuelve el mensaje para su transmisión
Content-Type	Naturaleza del mensaje (<i>Ver cuadro siguiente</i>)

Tipo	Subtipo	Descripción
Texto	Plano	Texto sin formato
	Enriquecido	Texto con comandos en formato sencillo
	HTML	Agregado cuando se popularizó la Web
Imagen	GIF	Imagen fija en formato GIF
	JPEG	Imagen fija en formato
Audio	Básico	Sonido
Video	MPEG	Película en formato MPEG
Aplicación	Octet-Stream	Secuencia de Bytes no interpretados
	PostScript	Documento imprimible en PostScript
Mensaje	RFC822	Mensaje MIME RFC 822
	Parcial	Mensaje dividido para su transmisión
	Externo	El mensaje mismo debe contenerse en la red
Multipartes	Mezclado	Partes independientes en el orden especificado
	Alternativa	Mismo mensaje en diferentes formatos
	Paralelo	Las partes deben verse en forma simultánea
	Compendio	Cada parte es un mensaje RFC 822 completo

Es responsabilidad del sistema receptor seleccionar la presentación adecuada

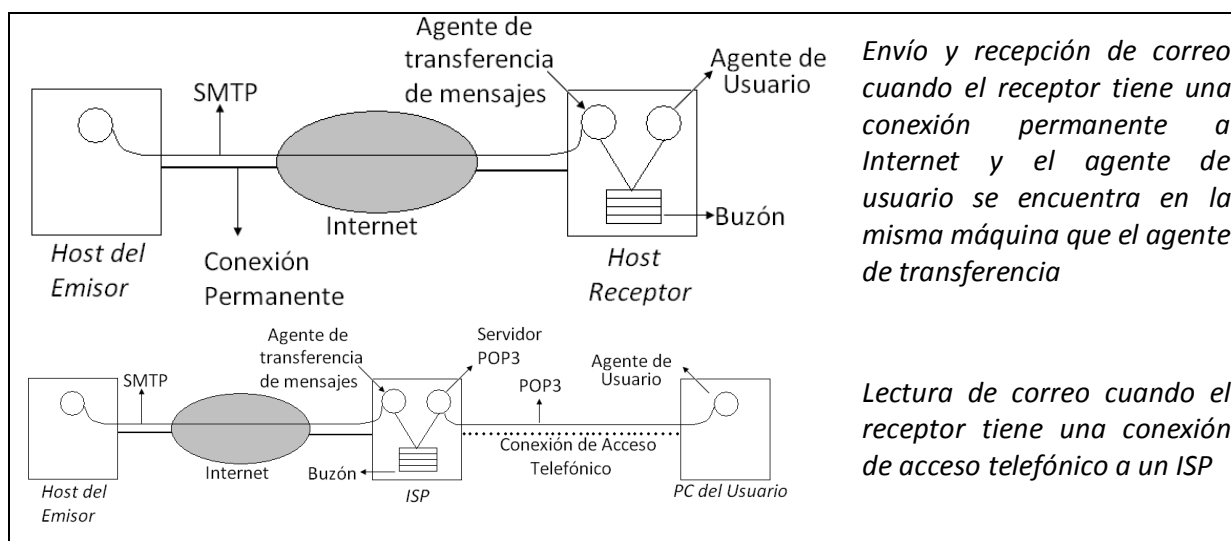
Transferencia de Mensajes: SMTP (Protocolo Simple de Transporte de Correo)

En Internet, el e-mail se entrega al hacer que la máquina de origen establezca una conexión TCP con el puerto 25 de la máquina de destino. Escuchando este puerto está un demonio de e-mail que habla con SMTP. Este demonio acepta conexiones de entrada y copia mensajes de ellas a los buzones adecuados. Si no puede entregarse el mensaje, se devuelve al remitente un informe de error que contiene la primera parte del mensaje que no pudo entregarse.

SMTP es un protocolo ASCII sencillo. Después de establecer la conexión TCP con el puerto 25, la máquina emisora, operando como cliente, espera que la máquina receptora, operando como servidor, hable primero. El servidor comienza enviando una línea de texto que proporciona su identidad e indica si está preparado o no para recibir correo. Si lo está, el cliente libera la conexión y lo intenta después.

Si el servidor está dispuesto a aceptar e-mail, el cliente anuncia de quién proviene el mensaje y a quién va dirigido. Si existe el destinatario en el destino, el servidor da al cliente el permiso de enviar el mensaje. A continuación, el cliente envía el mensaje y el servidor confirma su recepción. Por lo general, no se requieren sumas de verificación porque TCP proporciona un flujo de bytes confiable. Si hay más e-mail, se envía ahora.

Aunque el protocolo SMTP está bien definido, pueden surgir algunos problemas. Uno se relaciona con la longitud del mensaje. Algunas implementaciones más viejas no pueden manejar mensajes mayores que 64 KB. Otro problema se relaciona con las terminaciones de temporizador. Si el cliente y el servidor tienen temporizadores diferentes, uno de ellos puede terminar mientras que el otro continúa trabajando, terminando en forma inesperada la conexión. Para superar estos problemas, se creó el **ESMTP (SMTP Extendido)**.



Entrega Final: POP3 (Protocolo de Oficina de Correos 3)

El problema es el siguiente: ¿Qué sucede cuando Elena desea enviar un Correo electrónico a Carolina y esta última no está en línea es ese momento? Elena no puede establecer una conexión TCP con Carolina, y por lo tanto no puede ejecutar SMTP.

Una solución es que un agente de transferencia de mensajes en una máquina ISP acepte correo electrónico para sus clientes y lo almacene en sus buzones en una máquina ISP. Puesto que este agente puede estar en línea todo el tiempo, el e-mail puede enviarse las 24 horas del día.

Pero esta solución genera otro problema. ¿Cómo obtiene el usuario el correo electrónico del agente de transferencia de mensajes del ISP? La solución a este problema es crear otro protocolo que permita que los agentes de transferencia de usuarios (en PCs cliente) contacten al agente de transferencia de mensajes (ISP) y que el e-mail se copie del ISP al usuario. Tal protocolo es **POP3**.

POP3 inicia cuando el usuario arranca el lector de correo. Este llama al ISP y establece una conexión TCP con el agente de transferencia de mensajes en el puerto 110. Una vez establecida la conexión, el protocolo pasa por 3 estados de secuencia:

1. Autorización: Tiene que ver con el inicio de sesión por parte del usuario (*user* y *pass*)
2. Transacciones: Son las acciones que realiza el usuario sobre el buzón.
3. Actualización: Es la ejecución de las transacciones

IMAP (Protocolo de Acceso a Mensajes de Internet)

Para un usuario que tiene una cuenta de correo electrónico con un ISP que siempre se accede desde una PC, POP3 es adecuado y se utiliza ampliamente debido a su sencillez y robustez.

IMAP supone que todo el correo electrónico permanecerá en el servidor de manera indefinida en múltiples buzones de correo. IMAP proporciona mecanismos de gran alcance para leer mensajes o incluso partes de un mensaje.

A diferencia de POP3, IMAP también puede aceptar correo saliente para enviarlo al destino, así como entregar correo electrónico entrante.

Característica	POP3	IMAP
En dónde se define el protocolo	RFC 1939	RFC 2060
Puerto TCP utilizado	110	143
En dónde se almacena el correo electrónico	PC del usuario	Servidor
En dónde se lee el correo electrónico	Sin conexión	En línea
Tiempo de conexión requerido	Poco	Mucho
Uso de recursos del servidor	Mínimo	Amplio
Múltiples buzones	No	Sí
Quién respalda los buzones	Usuario	ISP
Bueno para los usuarios móviles	No	Sí
Control del usuario sobre la descarga	Poco	Mucho
Descargas parciales de mensajes	No	Si
¿Es un problema el espado en disco?	No	Con el tiempo podría serlo
Sencillo de implementar	Sí	No
Soporte amplio	Sí	En crecimiento

Correo Web

Los correos web (Ej: [Gmail](#), [Hotmail](#)) contienen agentes de transferencia normales que escuchan el puerto 25 para conexiones SMTP entrantes. La parte interesante es la forma en que se entrega el e-mail. Básicamente, cuando el usuario va a la página de correo web, se despliega un formulario para poner nombre de usuario y clave. Una vez ingresados, se envían al servidor, quien los valida. Si el login es exitoso, el servidor encuentra el buzón de usuario y construye una lista de correo formateada en HTML. Esta página web se envía al navegador para que la despliegue.

WORLD WIDE WEB (WWW)

WWW es un almacén arquitectónico para acceder a documentos vinculados distribuidos en miles de máquinas de toda Internet. Comenzó en 1989 en el CERN (Centro Europeo de Investigación Nuclear).

En 1994, el CERN y el MIT firmaron un acuerdo para establecer el W3C (WWW Consortium), una organización dedicada al desarrollo web, la estandarización de protocolos y el fomento de interoperabilidad entre los sitios.

Desde el punto de vista del usuario, Web consiste en un enorme conjunto de documentos a nivel mundial, generalmente llamados **paginas web**. Cada página puede tener vínculos a otras páginas. La idea de hacer que una página apunte a otra se conoce como **hipertexto**. En muchas páginas web, las cadenas de texto que son vínculos a otras páginas, llamadas **hipervínculos**, se resaltan. Las páginas se pueden ver mediante programas llamados **navegadores**.

¿Cómo funciona web? Cuando el usuario hace clic en un hipervínculo vinculado al servidor *abcd.com*, el navegador lo sigue, enviándole al servidor un mensaje en el que le solicita la página. Cuando ésta llega, el navegador despliega una página web en la máquina cliente.

Las páginas se nombran utilizando **URLs (Localizadores Uniformes de Recursos)**. Permite responder a: ¿Cómo se llama la página? ¿Dónde está? ¿Cómo se puede acceder?. Sirve como nombre mundial de una página. Su debilidad es que apunta a un host específico.

Un URL tiene 3 partes: el nombre del protocolo (<http>), el nombre DNS de la máquina donde se localiza la página (www.abcd.com) y por lo general, el nombre del archivo que contiene la página ([index.html](http://www.abcd.com/index.html)).

Los pasos que se dan cuando se sigue un vínculo son:

1. El navegador determina el URL (<http://www.itu.org.ar/home/index.html>)
2. El navegador pide al DNS la dirección IP de www.itu.org.ar
3. DNS responde con 156.106.192.32
4. El navegador realiza una conexión TCP con el puerto 80 (HTTP) en 156.106.192.32
5. Después envía un mensaje en el que solicita el archivo [/home/index.html](http://www.itu.org.ar/home/index.html)
6. El servidor www.itu.org.ar envía el archivo [/home/index.html](http://www.itu.org.ar/home/index.html)
7. Se libera la conexión TCP
8. El navegador despliega todo el texto de [/home/index.html](http://www.itu.org.ar/home/index.html)
9. El navegador obtiene y despliega todas las imágenes del archivo

URL no proporciona ninguna manera de referirse a una página sin decir de manera simultánea dónde está. No hay manera de decir “quiero la página xyz y no me importa de dónde la traigas”. Para resolver este problema y hacer posible la duplicación de las páginas, la IETF está trabajando en un sistema de **URNs (Nombres Universales de Recursos)**

Para permitir que todos los navegadores entiendan todas las páginas web, éstas se escriben en un lenguaje estandarizado llamado HTML.

No todas las páginas contienen HTML. Una página puede consistir en un documento con formato PDF, un icono con formato GIF, una fotografía con formato JPEG, o cualquiera de los cientos de los otros tipos de archivos. Puesto que las páginas HTML estándar pueden vincular cualquiera de éstos, el navegador tiene un problema cuando encuentra una página que no puede interpretar.

En lugar de agrandar cada vez más los navegadores incorporándoles intérpretes para una colección creciente de tipos de archivos, la mayoría de los navegadores ha elegido una solución más general. Cuando un servidor regresa una página, también regresa alguna información

adicional acerca de ella. Dicha información incluye el tipo MIME de la página. Si el tipo MIME no es de los integrados, el navegador consulta su tabla de tipos MIME que le indique cómo desplegar la página. Esta tabla asocia un tipo MIME con un visor. Hay dos posibilidades:

- **Plug-in (conector):** es un módulo de código que usa el navegador, y tiene acceso a la página actual y puede modificar su apariencia.
- **Aplicación auxiliar:** es un programa completo que se ejecuta como un proceso independiente. Por lo general simplemente acepta el nombre de un archivo de trabajo en el que se ha almacenado el archivo de contenido, abre dicho archivo y despliega su contenido. Ej: [Acrobat reader](#), [Word](#), etc.

La capacidad de ampliar el navegador con un nº significativo de tipos nuevos es conveniente, pero también puede generar problemas. Cuando Internet Explorer obtiene un archivo EXE, sabe que este archivo es un programa ejecutable, y por lo tanto no tiene una aplicación auxiliar. La acción obvia es ejecutar el programa. Sin embargo, esto podría ser un enorme hoyo de seguridad. En Unix puede ocurrir un problema análogo con las secuencias de comandos Shell.

El servidor

Cuando el usuario teclea un URL o hace clic en una línea de hipertexto, el navegador lo analiza e interpreta la parte entre `http://` y la siguiente diagonal como un nombre DNS a buscar. Una vez que el navegador tiene la dirección IP del servidor, establece una conexión TCP con el puerto 80 de ese servidor. A continuación, envía un comando que contiene el resto del URL, que es el nombre del archivo que se encuentra en ese servidor. Éste regresa el archivo para que el navegador lo despliegue. Los pasos que da un servidor en su ciclo principal son:

1. Acepta una conexión TCP de un cliente (un navegador)
2. Obtiene el nombre del archivo solicitado
3. Obtiene el archivo (del disco)
4. Regresa el archivo al cliente

El problema de este diseño es que cada solicitud requiere un acceso al disco para obtener el archivo. El resultado es que el servidor no puede atender más solicitudes por segundo que accesos al disco. Una mejora obvia es mantener un caché. Otra sería usar múltiples subprocesos. La ventaja de este esquema, es que mientras uno o más módulos de procesamiento están bloqueados esperando a que termine una operación de disco, otros módulos pueden estar trabajando activamente en otras solicitudes. Por supuesto, para obtener cualquier mejora real sobre el modelo de un solo subproceso, es necesario tener múltiples discos, a fin de que más de un disco pueda estar ocupado al mismo tiempo.

En los servidores web modernos, el *front end* pasa las solicitudes entrantes al primer módulo de procesamiento libre, que después la transporta mediante alguno de los siguientes pasos:

1. Resuelve el nombre de la página web solicitada
2. Autentica al cliente
3. Realiza control de acceso al cliente
4. Realiza control de acceso en la página web
5. Verifica el caché
6. Obtiene de disco la página solicitada
7. Determina el tipo MIME que se incluirá en la respuesta
8. Se encarga de diversos detalles
9. Regresa la respuesta al cliente
10. Realiza una entrada en el registro del servidor.

Si llegan demasiadas solicitudes por segundo, la CPU no será capaz de manejar la carga de procesamiento, sin importar cuántos discos se utilicen en paralelo. La solución es agregar más nodos (PC), posiblemente con discos replicados para evitar que los discos se vuelvan el siguiente cuello de botella. Esto se llama **granja de servidores**. Un problema con las granjas de servidores es que el caché no es compartido. Otro problema es que la conexión TCP del cliente termine en el *front end* (consulta sencilla). La solución a esto es mediante una **transferencia TCP**, mediante la cual se destina un nodo de procesamiento para que finalice la consulta.

Sin estado y cookies

Web básicamente no tiene estado. No existe el concepto de inicio de sesión. El navegador envía una solicitud a un servidor y obtiene un archivo. A continuación, el servidor olvida que ha visto alguna vez a ese cliente en particular.

Algunos sitios web requieren que los clientes se registren para poder utilizarlos. Esto da lugar a la pregunta de cómo los servidores pueden distinguir entre las solicitudes de usuarios registrados y las demás. A primera vista, uno podría pensar que los servidores los distinguen por la dirección IP, pero esta idea no funciona.

Para resolver esto, Netscape inventó una técnica muy criticada llamada **cookies**. Cuando un cliente solicita una página web, el servidor puede proporcionar información adicional junto a la página solicitada. Esta información puede incluir una cookie, que es un pequeño archivo (de menos de 4 KB). Los navegadores almacenan cookies ofrecidas en un directorio de cookies en el disco duro de la máquina del cliente.

Una cookie puede contener hasta 5 campos.

1. El *dominio* indica de dónde viene la cookie. Cada dominio puede almacenar hasta 20 cookies por cliente.
2. La *ruta* es la ruta en la estructura del directorio del servidor que identifica qué partes del árbol de archivos del servidor podrían utilizar la cookie.
3. El campo *contenido* toma la forma *nombre = valor*. Puede ser lo que el servidor desee.
4. El campo *Expira* indica cuando caduca la cookie. Si este campo está ausente, el navegador descarta la cookie cuando sale. Tal cookie se conoce como **cookie no persistente**. Si se proporciona una fecha y hora, se dice que es **persistente**.
5. El campo *seguro* puede establecerse para indicar que el navegador podría simplemente regresar la cookie a un servidor seguro.

Dominio	Ruta	Contenido	Expira	Seguro
Toms-casino.com	/	ClienteID=425863	15-10-02 17:00	Si
EBay.com	/	Cart=1-00501;1-07031	11-10-02 14:22	No
Aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jets	31-12-10 23:59	No

¿Cómo se utilizan? Justo antes de que un navegador solicite una página a un sitio web, verifica su directorio de cookies para ver si el dominio al que está solicitando la página ya colocó alguna cookie. De ser así, todas las cookies colocadas por ese dominio se incluyen en el mensaje de solicitud. Cuando el servidor las obtiene, puede interpretarlas de la forma que desee.

Las cookies también han sido objeto de malos usos. Un uso controversial de las cookies es coleccionar de manera secreta información sobre los hábitos de navegación web de los usuarios.

Para mantener algo de su privacidad, algunos usuarios configuran su navegador para que rechacen cookies. Pero esto puede darle problemas a sitios web legítimos que utilizan cookies.

Documentos web Estáticos: HTML (Lenguaje de Marcado de Hipertexto)

En la actualidad, las páginas web se escriben en un lenguaje llamado HTML, que es un lenguaje de marcado para describir cómo se van a formatear los documentos. Al integrar todos los comandos de marcado dentro de cada archivo HTML y al estandarizarlos, se hace posible que cualquier navegador web lea y reformatee cualquier página web.

Una página web consiste en un encabezado y un cuerpo encerrado entre **etiquetas**. Los comandos de las etiquetas se llaman **directivas**. Algunas etiquetas tienen parámetros, llamados **atributos**.

Ej: ` ; <body>...</body>; <h1>Texto grande </h1>`

Cuando un sitio web es complejo y consiste en muchas páginas creadas por diversos autores que trabajan para la misma compañía, con frecuencia es deseable tener una forma de evitar que páginas diferentes tengan apariencias distintas. Este problema puede resolverse usando **hojas de estilo**.

A partir de HTML 2.0 se incorporaron **formularios**, que contienen cuadros o botones que permiten a los usuarios proporcionar información o tomar decisiones, y después enviar dicha información al dueño de la página.

Documentos web Estáticos: XML y XSL

HTML, con o sin formularios, no proporciona estructura alguna para las páginas web. Además, mezcla el contenido con el formato. El W3C ha mejorado el HTML para permitir que las páginas web tengan estructura para su procesamiento automatizado. Para este propósito, se han desarrollado 2 nuevos lenguajes. El primero, **XML (Lenguaje de Marcado Extensible)**, describe el contenido Web de una forma estructurada, y el segundo, **XSL (Lenguaje de Hojas de Estilo Extensibles)**, describe el formato independientemente del contenido. Veamos un ejemplo:

```
<autor>
  <nombre> Adrián </nombre>
  <apellido> Botta </apellido>
</autor>
```

Todo lo que muestra el ejemplo es la definición de un campo *autor* que contiene un registro con un *nombre* y un *apellido*. No dice nada como desplegar la página web en la pantalla. Para proporcionar la información de formato, necesitamos un segundo archivo *autores.XSL*, que contenga la definición XSL. Este archivo es una hoja de estilo que indica cómo desplegar los datos.

Además de escribir páginas web, XML también sirve para otros propósitos. Por ejemplo, se puede utilizar como lenguaje para la comunicación entre programas de aplicación. En particular, **SOAP (Protocolo Simple de Acceso a Objetos)** es una forma de realizar RPC entre aplicaciones en forma independiente del lenguaje y de la aplicación. El cliente construye la solicitud como un mensaje XML y lo envía al servidor utilizando el protocolo http. El servidor envía una respuesta como un mensaje XML formateado. De esta manera, las aplicaciones de plataformas heterogéneas pueden comunicarse.

Documentos web Estáticos: XHTML (Lenguaje de Marcado de Hipertexto Extendido)

XHTML es básicamente HTML 4.0 reformulado en XML. Hay 6 diferencias entre XHTML y HTML:

- 1) Las páginas XHTML y los navegadores deben apegarse estrictamente al estándar
- 2) Todas las etiquetas y atributos deben estar en minúsculas
- 3) Es obligatorio el uso de etiquetas de cierre. Ej: `<img...> `

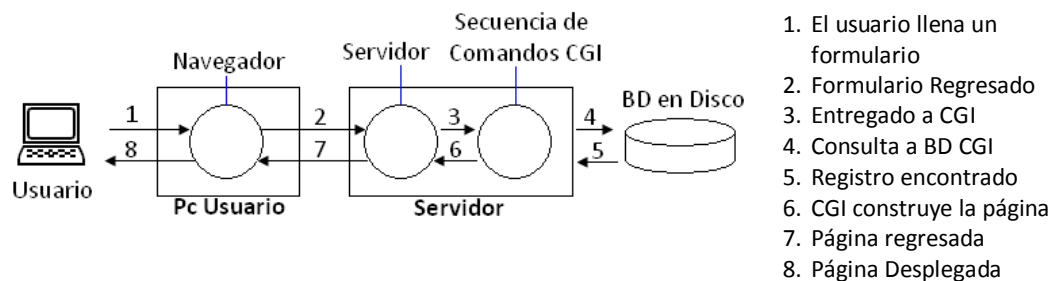
- 4) Los atributos deben estar encerrados entre comillas
- 5) Las etiquetas deben estar anidadas apropiadamente
- 6) Cada documento debe especificar su tipo de documento.

Documentos Web Dinámicos

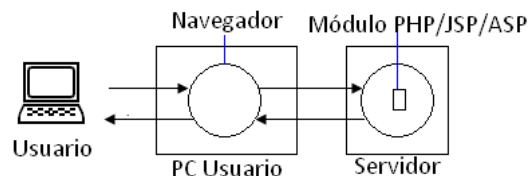
En los años recientes, cada vez más contenido es dinámico, es decir, se genera a solicitud, en lugar de almacenarlo en disco. La generación de contenido puede suceder en:

1. Servidor

- a. CGI (Interfaz de Puerta de Enlace Común): Es la forma tradicional de manejar formularios y otras páginas web interactivas. Es una interfaz estandarizada para permitir que los servidores web hablen con los programas *back-end* y las secuencias de comandos puedan aceptar datos de entrada y generar en respuesta páginas HTML.

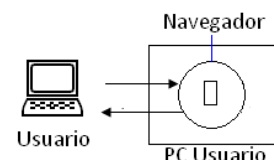


- b. PHP (Preprocesador de Hipertexto): Es un lenguaje para escribir pequeñas secuencias de comandos que se incrustan en las páginas web. Estas secuencias se ejecutan en el servidor y genera la página. Por lo general, los servidores esperan páginas web que contienen PHP con una extensión *.php* en vez de *.html*. La página (antes de ejecutarse) contiene HTML normal, excepto por la secuencia de comandos PHP dentro de la etiqueta `<?php ...?>`. PHP es un lenguaje potente, de código open source.
- c. JSP (Java Server Pages): Es similar a PHP, sólo que la parte dinámica se escribe en JAVA
- d. ASP (Active Server Pages): Es la versión de Microsoft de PHP y JSP. Usa VB Script.



2. **Cliente:** Las secuencias de comandos de CGI, PHP, JSP y ASP resuelven el problema de manejar formularios e interacciones con BD en el servidor. Lo que ninguno de ellos puede hacer es responder a los movimientos del Mouse o interactuar directamente con el usuario. Para esto, es necesario tener secuencias de comandos incrustadas en páginas HTML que se ejecuten en la máquina cliente y no en el servidor. Hay 2 métodos comunes para esto:

- a. JavaScript: Es el lenguaje de secuencias de comando más popular no estandarizado.
- b. Applets (Subprogramas): Son pequeños programas Java que se han compilado en instrucciones de máquina para una computadora virtual llamada **JVM (Máquina Virtual de Java)**.
- c. Controles ActiveX: Son programas compilados para el lenguaje de máquina Pentium y ejecutados en Hardware.



HTTP (Protocolo de Transferencia de Hipertexto)

Especifica cuáles mensajes pueden enviar los clientes a los servidores y qué respuestas obtienen. Cada interacción consiste en una solicitud ASCII, seguida de una respuesta tipo MIME.

La forma común en la que un navegador contacta un servidor es estableciendo una conexión TCP con el puerto 80 de la máquina del servidor. En HTTP 1.0, cada vez que se establecía una conexión, se enviaba una solicitud y se obtenía una respuesta. Después se liberaba dicha conexión. Desde HTTP 1.1, se soportan **conexiones persistentes**. Con ellas, es posible establecer una conexión TCP, enviar una solicitud y obtener una respuesta, y después enviar solicitudes adicionales y obtener respuestas adicionales.

HTTP se diseñó con miras a las aplicaciones orientadas a objetos futuras, por lo que se soportan otras operaciones, llamadas **métodos**, diferentes a las de solicitar una página web.

Método	Descripción
GET	Solicita la lectura de una página web
HEAD	Solicita la lectura del encabezado de una página web
PUT	Solicita el almacenamiento de una página web
POST	Inserta algo a un recurso con nombre
DELETE	Elimina la página web
TRACE	Repite la solicitud entrante
CONNECT	Reservado para uso futuro
OPTIONS	Consulta ciertas opciones

Cada solicitud obtiene una respuesta, que consiste en una línea de estado, y posiblemente una información adicional.

Código	Significado	Ejemplos
1xx	Información	100 = el servidor está de acuerdo en manejar la solicitud del cliente
2xx	Éxito	200 = solicitud exitosa; 204 = no hay contenido
3xx	Redirección	301 = página movida; 304 = la página en caché aún es válida
4xx	Error del cliente	403 = página prohibida; 404 = página no encontrada
5xx	Error del servidor	500 = error interno; 503 = intentar más tarde

A la línea de solicitud le pueden seguir líneas adicionales que contienen más información. Estas se llaman **encabezados de solicitud**. Esta información puede compararse con los parámetros de una llamada a procedimiento. Las respuestas también pueden tener **encabezados de respuesta**. Algunos encabezados pueden utilizarse en cualquier dirección. [Ejemplos](#):

- Encabezado DATE (solicitud/respuesta): Fecha y hora en que se envió el mensaje
- Encabezado COOKIE (solicitud): Regresa al servidor una cookie establecida previamente

Mejoras de Desempeño

Como consecuencia de los retardos interminables de web, los investigadores han desarrollado varias técnicas para mejorar el desempeño. A continuación examinaremos 3 de ellas:

1. Almacenamiento en Caché: Es la técnica de guardar páginas para uso posterior. El procedimiento común es que algún proceso **Proxy** mantenga el caché. Para utilizar el almacenamiento en caché, un navegador puede configurarse para que todas las solicitudes de las páginas se hagan a un Proxy en lugar del servidor real de la página. Si el Proxy tiene la página, la regresa de inmediato. De lo contrario, la obtiene del servidor, la agrega a caché para uso posterior, y la envía al cliente que la solicitó.

¿Quién debe realizar el almacenamiento en caché? Con frecuencia, todos los caché funcionan al mismo tiempo, por lo que las solicitudes primero van al Proxy local. Si este falla, consulta al Proxy de la LAN. Si este falla, prueba con el Proxy del ISP (Almacenamiento de **caché jerárquico**)

¿Por cuánto tiempo debe almacenarse en caché las páginas? Algunas páginas no deberían almacenarse en caché (Ej: [Páginas web con contenido dinámico](#)). El elemento clave para determinar cuándo expulsar una página del caché es qué tanta obsolescencia están dispuestos a aceptar los usuarios. Hay 2 métodos para tratar este problema:

- Usar una heurística para adivinar cuánto tiempo mantener la página. Ej: [ver el campo *last-modified* del encabezado](#)
- Una de las características más útiles es el encabezado *if-modified-since*, que puede ser enviado por un Proxy a un servidor. Especifica la página que el Proxy desea y la fecha en que ésta fue modificada por última vez (a partir de *last-modified*). Si la página no se ha modificado desde entonces, el servidor regresa un mensaje corto *Not-Modified*, para que el Proxy use la página de caché. Si ha sido modificada, devuelve la nueva

Otro enfoque es el almacenamiento de **caché proactivo**, que consiste en que luego de una solicitud, el Proxy pida la precarga de páginas relacionadas a la solicitada mediante hipervínculos, antes de que el usuario las solicite (o no).

2. Replicación del Servidor: Consiste en replicar la información que contiene un servidor en múltiples ubicaciones separadas considerablemente. También se conoce como **espejo**. Desafortunadamente, Web tiene un fenómeno conocido como **flash crowds**, en el que un sitio web que era un lugar alejado y desconocido y no visitado, de repente se vuelve el centro del universo. Lo que se necesita es una forma para que un sitio web que de repente note un incremento masivo en el tráfico, se clone automáticamente a sí mismo en tantas ubicaciones cuando sea necesario y mantenga funcionando a esos sitios hasta que pase la tormenta. Para esto, se necesitan acuerdos con compañías de posting.
3. Redes de Entrega de Contenido (CDN): La brillantez del capitalismo es que alguien ha descubierto como hacer dinero con WWW. Funciona como sigue. Las compañías llamadas CDN hablan con proveedores de contenido, que desean que su contenido esté disponible rápidamente, y ofrecen entregar su contenido a los usuarios finales de manera eficiente a cambio de una cuota. Después de que se firma el contrato, el dueño da a la CDN el contenido de su sitio web para que lo pre-procese y después lo distribuya

A continuación, la CDN habla con una gran cantidad de ISPs y ofrece pagarles bien a cambio de que le permitan colocar en sus LANs un servidor manejado de manera remota lleno de contenido valioso. Ésta no es solo una fuente de ingresos, sino que también proporciona a los clientes de los ISPs excelente tiempo de respuesta para obtener el contenido de la CDN, lo que da al ISP una ventaja competitiva sobre otros ISPs que no han tomado dinero gratis de la CDN. Bajo estas condiciones, firmar con una CDN es una decisión que el ISP no necesita pensar.

LA WEB INALÁMBRICA

WAP – Protocolo de Aplicaciones Inalámbricas

La idea básica es utilizar la infraestructura existente digital inalámbrica. Los usuarios pueden literalmente llamar a una puerta de enlace WAP a través del enlace inalámbrico y enviarle solicitudes de páginas web. A continuación dicha puerta de enlace verifica su caché para ver si tiene la página solicitada. Si la tiene, la envía; si no la tiene, la obtiene a través de la red alámbrica.

WAP es en esencia una pila de protocolos para acceder a Web, pero está optimizada para conexiones de ancho de banda bajo que utilizan dispositivos inalámbricos que tienen una CPU lenta, poca memoria, y una pantalla pequeña. Los protocolos Usados por WAP son:

Entorno de Aplicaciones Inalámbricas (WAE) (similar a XML)
Protocolo de Sesión Inalámbrica (WSP)
Protocolo de Transacciones Inalámbricas (WTP)
Capa Inalámbrica de Seguridad de Transporte (WTLS)
Protocolo de Datagrama Inalámbrico (WDP) (similar a UDP)
Capa del Portador (GSM, CDMA, D-AMPS, GPRS, etc)

WAP no utiliza HTML, por esto es que no fue muy aceptado

I-Mode – Information Mode

Surge en Japón. El sistema i-mode tiene 3 componentes principales: un nuevo sistema de transmisión, un nuevo microteléfono, y un nuevo lenguaje para el diseño de páginas Web. El sistema de transmisión consiste en 2 redes separadas: la red de teléfonos móviles de conmutación de circuitos existentes, y una nueva red de conmutación de paquetes construida específicamente para el servicio i-mode.

I-mode utiliza la red de conmutación de paquetes y siempre está activo (como ADSL), por lo que no se cobra una tarifa por tiempo de conexión, sino por paquete enviado. El servicio se accede desde microteléfonos, que son similares a unas Palms. Los sitios i-mode son regulados por NTT DoCoMo, y deben cumplir ciertas condiciones. El servicio está dirigido a adolescentes.

Característica	WAP	I-mode
Qué es	Pila de protocolos	Servicio
Dispositivo	Microteléfono, PDA, notebook	Microteléfono
Acceso	Marcado telefónico	Siempre activo
Red subyacente	Conmutación de circuitos	Hay 2: circuitos + paquetes
Tasa de datos	9600 bps	9600 bps
Pantalla	Monocroma	A color
Lenguaje de marcado	WML (Aplicación XML)	cHTML
Lenguaje de secuencia de comandos	WML script	Ninguno
Cargos por uso	Por minuto	Por paquete
Pago por compras	Tarjeta de crédito	Recibo telefónico
Pictogramas (emoticons)	No	Si
Estandarización	Estándar abierto del foro WAP	Propietario NTT DoCoMo
Usuario Típico	Hombre de negocios	Jóvenes
En dónde se utiliza	Europa, Japón	Japón

WAP 2.0

Se suponía que WAP 1.0, basado en estándares internacionales reconocidos, sería una herramienta para la gente de negocios en movimiento. Fracasó. I-mode era un juguete electrónico para los adolescentes japoneses. Fue un éxito en ese sentido.

Cada lado aprendió algo de la primera generación de web inalámbrica. El consorcio WAP aprendió que el contenido importa. No tener un gran número de sitios web que hablan su lenguaje de mercado es fatal (no soportaba HTML). NTT DoCoMo aprendió que un sistema propietario cerrado, estrechamente enlazado con los microteléfonos y con la cultura japonesa no es un buen producto de exportación.

La conclusión es que ambos lados aprendieron que para convencer a una gran cantidad de sitios web para que coloquen su contenido en el formato de uno, es necesario tener un lenguaje de mercado estable y abierto que sea aceptado de manera universal. Las guerras de formatos no son buenas para los negocios.

WAP 2.0 también tiene algunas nuevas características:

1. Modelo push (de actualización automática) y modelo pull (de recepción automática)
2. Soporte para integrar la telefonía en las aplicaciones
3. Mensajería multimedia
4. Inclusión de 264 pictogramas (emoticons)
5. Integración con un dispositivo de almacenamiento
6. Soporte en el navegador para plug-ins

En WAP 2.0 también hay diversas diferencias técnicas. Las dos más significativas tienen que ver con la pila de protocolos y con el lenguaje de marcado. WAP 2.0 continúa soportando la antigua pila de protocolos, pero también soporta la pila estándar de Internet con TCP y HTTP/1.1. Sin embargo, se realizaron cuatro pequeños (pero compatibles) cambios (para simplificar el código) a TCP: uso de una ventana fija de 64 KB, inicio rápido, una MTU máxima de 1500 bytes y un algoritmo de retransmisión ligeramente diferente.

XHTML	
WSP	HTTP
WTP	TLS
WTLS	TCP
WDP	IP
Capa del portador	Capa del portador

Pila de protocolos de WAP 1.0 Pila de protocolos de WAP 2.0

ANEXOS

PUERTOS Y PROTOCOLOS MÁS USADOS

Puerto	Protocolo	Protocolo de aplicación	Nombre Protocolo
20/21	TCP	FTP (Datos/Control)	Protocolo de Transferencia de Archivos
23	TCP	Telnet	
25	TCP	SMTP	
53	TCP/UDP	DNS	Protocolo Simple de Transferencia de Correo Servidor de Nombres de Dominio
67	UDP	DHCP	Protocolo de Configuración de Host Dinámico
69	UDP	TFTP	FTP Trivial
80	TCP	HTTP	Protocolo de Transporte de Hipertexto
88	TCP/UDP	Kerberos	
102	TCP	X.400	
110	TCP	POP3	Protocolo de Oficina de Correos 3
119	TCP	NNTP	Protocolo de Transporte de noticias de red
123	UDP	NTP	Protocolo de tiempo de Red
123	UDP	SNTP	Protocolo Simple de tiempo de Red
135	TCP/UDP	RPC	Llamada a procedimiento remoto
143	TCP	IMAP	Protocolo de Acceso a Mensajes de Internet
161/162	UDP	SNMP	Protocolo Simple de Manejo de Red
389	TCP/UDP	LDAP	Protocolo Ligero de Acceso a Directorios
443	TCP	HTTPS	HTTP Seguro
445	TCP	SMB	Bloque de Mensaje de Servidor
554	TCP	RTSP	Real Time Streaming Protocol
563	TCP	NNTP sobre SSL	
593	TCP	RPC sobre HTTP	
593	TCP	RPC sobre HTTP	
636	TCP/UDP	LDAP SSL	
993	TCP	IMAP sobre SSL	
995	TCP	POP3 sobre SSL	