Comunicación de Datos

- Introducción
  - Clases de redes
  - Jerarquías de protocolos
  - Problemas en el diseño de los niveles
  - Servicios
- Modelos de referencia de redes
  - OSI
  - TCP/IP
  - OSI vs. TCP/IP
  - Un ejemplo: Novell NetWare
- El nivel físico
  - La velocidad máxima de un canal
  - Medios de transmisión
  - El sistema telefónico
    - Los local loops
    - Los troncales y la multiplexación
    - MDT en el sistema telefónico
  - Conmutación
  - Narrowband ISDN
  - Broadband ISDN y ATM
  - Conmutadores de ATM
  - Satélites
- El nivel de enlace
  - Asuntos de diseño
    - Servicios para el nivel de red
    - Marcos
    - Control de errores
    - Control de flujo
  - Detección y corrección de errores
  - Códigos de detección de errores
    - Códigos de CRC
  - Protocolos elementales de enlace
  - Protocolos de *ventana deslizante*
    - SLIP y PPP
    - El nivel de enlace de ATM
- Redes de broadcast
  - ALOHA
  - Protocolos de acceso múltiple con sentido de portador
  - Protocolos de CSMA con la detección de choques
  - Protocolos libre de choques
  - IEEE 802.3 y Ethernet
  - Bridges
  - LANs de velocidad alta

- El nivel de red
    - Estructura interna de la subred
    - Algoritmos de ruteo
        - Algoritmos estáticos
        - Ruteo de vector de distancia
        - Ruteo de estado de enlace
        - Ruteo jerárquico
        - Ruteo de broadcast
    - Algoritmos de control de congestión
        - Formación del tráfico
        - Control de congestión en subredes de circuitos virtuales
        - Paquetes de bloqueo
        - Pérdida de carga
    - Internets
    - El nivel de red en la Internet
        - Protocolos de control
        - IPv6
- El nivel de transporte
    - Primitivas del servicio de transporte
    - Protocolos de transporte
        - Establecimiento de una conexión
        - Desconexión
        - Control de flujo
        - Multiplexación
        - Recuperación de caídas
    - El protocolo de TCP
        - Implementación del protocolo
        - El encabezamiento de TCP
        - Administración de conexiones
        - Política de transmisión
        - Control de congestión
        - Administración de relojes
    - Rendimiento
        - Diseño para rendimiento mejor
        - Procesamiento rápido de TPDUs
- El nivel de aplicación
    - DNS--Domain Name System
        - Espacio de nombres de DNS
        - Registros de recurso
        - Servidores de nombres

Introducción

- ¿Qué es una red de computadores? Una colección interconectada de computadores autónomos.
- ¿Para qué se usan las redes?
    - Compartir recursos, especialmente la información (los datos)
    - Proveer la confiabilidad: más de una fuente para los recursos
    - La escalabilidad de los recursos computacionales: si se necesita más poder computacional, se puede comprar un cliente más, en vez de un nuevo mainframe
    - Comunicación

Clases de redes

- Podemos clasificar las redes en las dimensiones de la tecnología de transmisión y del tamaño.
- Tecnología de transmisión
    - **Broadcast.** Un solo canal de comunicación compartido por todas las máquinas. Un *paquete* mandado por alguna máquina es recibido por todas las otras.
    - **Point-to-point.** Muchas conexiones entre pares individuales de máquinas. Los paquetes de A a B pueden atravesar máquinas intermedias, entonces se necesita el ruteo (*routing*) para dirigirlos.
- Escala
    - Multicomputadores: 1 m
    - LAN (local area network): 10 m a 1 km
    - MAN (metropolitan area network): 10 km
    - WAN (wide area network): 100 km a 1.000 km
    - Internet: 10.000 km
- LANs
    - Normalmente usan la tecnología de broadcast: un solo cable con todas las máquinas conectadas.
    - El tamaño es restringido, así el tiempo de transmisión del peor caso es conocido.
    - Velocidades típicas son de 10 a 100 Mbps (megabits por segundo; un megabit es 1.000.000 bits, no 2^{20}).
- WANs
    - Consisten en una colección de *hosts* (máquinas) o LANs de hosts conectados por una *subred.*
    - La subred consiste en las líneas de transmisión y los *ruteadores,* que son computadores dedicados a cambiar de ruta.
    - Se mandan los paquetes de un ruteador a otro. Se dice que la red es *packet-switched* (paquetes ruteados) o *store-and-forward* (guardar y reenviar).
- Internet
    - Una *internet* es una red de redes vinculadas por *gateways,* que son computadores que pueden traducir entre formatos incompatibles.
    - La Internet es un ejemplo de una internet.
- Redes inalámbricas
    - Una red inalámbrica usa radio, microondas, satélites, infrarrojo, u otros mecanismos para comunicarse.
    - Se pueden combinar las redes inalámbricas con los computadores móviles, pero los dos conceptos son distintos:

Inalámbrico	Móvil	Aplicación
No	No	Workstations estacionarias

No	Sí	Uso de un portable en un hotel
Sí	No	LANs en un edificio antiguo sin cables
Sí	Sí	PDA (personal digital assistant) para inventario

Jerarquías de protocolos

- El software para controlar las redes se tiene que estructurar para manejar la complejidad.
- Se organiza la mayor parte de las redes en una pila de niveles.
- Cada nivel ofrece ciertos servicios a los niveles superiores y oculta la implantación de estos servicios. Usa el nivel inferior siguiente para implementar sus servicios.
- El nivel n de una máquina se comunica con el nivel n de otra máquina. Las reglas y convenciones que controlan esta conversación son el *protocolo de nivel n.*
- Las entidades en niveles correspondientes de máquinas distintas son *pares.* Son los pares que se comunican.
- En la realidad el nivel n de una máquina no puede transferir los datos directamente al nivel n de otra. Se pasa la información hacia abajo de un nivel a otro hasta que llega al nivel 1, que es el medio físico.
- Entre los niveles están las interfaces. Las interfaces limpias permiten cambios en la implementación de un nivel sin afectar el nivel superior.
- Un nivel que tiene que transmitir un paquete a otra máquina puede agregar un *encabezamiento* al paquete y quizás partir el paquete en muchos. Por ejemplo, el encabezamiento puede identificar el mensaje y el destino. El nivel 3 de la mayor parte de las redes impone un límite en el tamaño de los paquetes.

Problemas en el diseño de los niveles

- Un mecanismo para identificar los remitentes y los recibidores.
- Transferencia de datos:
  - **Simplex.** Solamente en un sentido.
  - **Half-duplex.** En ambos, pero uno a la vez.
  - **Full-duplex.** En ambos a la vez.
- Control de errores y detección de recepción.
- Orden de mensajes.
- Velocidades distintas de transmisión y recepción.
- Ruteo.

Servicios

- Cada nivel provee un servicio al nivel superior.
- Hay dos tipos de servicios:
  - **Servicio orientado a la conexión.** Como el sistema telefónico. La conexión es como un tubo, y los mensajes llegan en el orden en que fueron mandados.
  - **Servicio sin conexión.** Como el sistema de correo. Cada mensaje trae la dirección completa del destino, y el ruteo de cada uno es independiente.

- Se caracterizan los servicios por la calidad de servicio.
    - Compara la transferencia de archivos con la comunicación de voz (ambas orientadas a la conexión).
    - Para e-mail un servicio sin conexión y no confiable es suficiente, esto se llama *servicio de datagrama*. Para dar confianza los servicios de datagrama con acuses de recibo son posibles.
- Cada servicio define un conjunto de primitivas (tales como "solicitar" o "acusar recibo"). Por contraste el protocolo es el conjunto de reglas que controlan el formato y significado de los paquetes intercambiados por entidades de par. Se usan los protocolos para implementar los servicios.

Modelos de referencia de redes

Examinamos dos arquitecturas de red importantes: ISO OSI y TCP/IP.

OSI

- OSI es el *Open Systems Interconnection Reference Model*. Tiene siete niveles. En realidad no es una arquitectura particular, porque no especifica los detalles de los niveles, sino que los estándares de ISO existen para cada nivel.
- **Nivel físico.** Cuestiones: los voltajes, la duración de un bit, el establecimiento de una conexión, el número de polos en un enchufe, etc.
- **Nivel de enlace.** El propósito de este nivel es convertir el medio de transmisión crudo en uno que esté libre de errores de transmisión.
    - El remitente parte los datos de input en *marcos de datos* (algunos cientos de bytes) y procesa los *marcos de acuse*.
    - Este nivel maneja los marcos perdidos, dañados, o duplicados.
    - Regula la velocidad del tráfico.
    - En una red de broadcast, un subnivel (el subnivel de acceso medio, o *medium access sublayer*) controla el acceso al canal compartido.
- **Nivel de red.** Determina el ruteo de los paquetes desde sus fuentes a sus destinos, manejando la congestión a la vez. Se incorpora la función de contabilidad.
- **Nivel de transporte.** Es el primer nivel que se comunica directamente con su par en el destino (los de abajo son de máquina a máquina). Provee varios tipos de servicio (por ejemplo, un canal punto-a-punto sin errores). Podría abrir conexiones múltiples de red para proveer capacidad alta. Se puede usar el encabezamiento de transporte para distinguir entre los mensajes de conexiones múltiples entrando en una máquina. Provee el control de flujo entre los hosts.
- **Nivel de sesión.** Parecido al nivel de transporte, pero provee servicios adicionales. Por ejemplo, puede manejar *tokens* (objetos abstractos y únicos) para controlar las acciones de participantes o puede hacer *checkpoints* (puntos de recuerdo) en las transferencias de datos.
- **Nivel de presentación.** Provee funciones comunes a muchas aplicaciones tales como traducciones entre juegos de caracteres, códigos de números, etc.
- **Nivel de aplicación.** Define los protocolos usados por las aplicaciones individuales, como e-mail, `telnet`, etc.

TCP/IP

- Tiene como objetivos la conexión de redes múltiples y la capacidad de mantener conexiones aun cuando una parte de la subred esté perdida.
- La red es packet-switched y está basada en un nivel de internet sin conexiones. Los niveles físico y de enlace (que juntos se llaman el "nivel de host a red" aquí) no son definidos en esta arquitectura.
- **Nivel de internet.** Los hosts pueden introducir paquetes en la red, los cuales viajan independientemente al destino. No hay garantias de entrega ni de orden.
  Este nivel define el *Internet Protocol* (IP), que provee el ruteo y control de congestión.

- **Nivel de transporte.** Permite que pares en los hosts de fuente y destino puedan conversar. Hay dos protocolos:
    - **Transmission Control Protocol (TCP).** Provee una conexión confiable que permite la entrega sin errores de un flujo de bytes desde una máquina a alguna otra en la internet. Parte el flujo en mensajes discretos y lo monta de nuevo en el destino. Maneja el control de flujo.
    - **User Datagram Protocol (UDP).** Es un protocolo no confiable y sin conexión para la entrega de mensajes discretos. Se pueden construir otros protocolos de aplicación sobre UDP. También se usa UDP cuando la entrega rápida es más importante que la entrega garantizada.
- **Nivel de aplicación.** Como en OSI. No se usan niveles de sesión o presentación.

OSI vs. TCP/IP

- OSI define claramente las diferencias entre los servicios, las interfaces, y los protocolos.
    - Servicio: lo que un nivel hace
    - Interfaz: cómo se pueden accesar los servicios
    - Protocolo: la implementación de los servicios

  TCP/IP no tiene esta clara separación.
- Porque OSI fue definido antes de implementar los protocolos, los diseñadores no tenían mucha experiencia con donde se debieran ubicar las funcionalidades, y algunas otras faltan. Por ejemplo, OSI originalmente no tiene ningún apoyo para broadcast.
- El modelo de TCP/IP fue definido después de los protocolos y se adecúan perfectamente. Pero no otras pilas de protocolos.
- OSI no tuvo exíto debido a
    - Mal momento de introducción: insuficiente tiempo entre las investigaciones y el desarrollo del mercado a gran escala para lograr la estandarización
    - Mala tecnología: OSI es complejo, es dominado por una mentalidad de telecomunicaciones sin pensar en computadores, carece de servicios sin conexión, etc.
    - Malas implementaciones
    - Malas políticas: investigadores y programadores contra los ministerios de telecomunicación
- Sin embargo, OSI es un buen modelo (no los protocolos). TCP/IP es un buen conjunto de protocolos, pero el modelo no es general. Usarémos una combinación de los dos:

Nivel de aplicación
Nivel de transporte
Nivel de red
Nivel de enlace
Nivel físico

Un ejemplo: Novell NetWare

- Es el sistema de red más popular en el mundo de PC.
- Modelo de cliente-servidor para los LANs.
- Arquitectura:

Aplicación	SAP, servidor de archivos, ...
Transporte	NCP, SPX

Red	IPX
Enlace	Ethernet, token ring, ARCnet
Físico	Ethernet, token ring, ARCnet

- IPX es como IP, pero con direcciones de 10 bytes.
- NCP está orientado a la conexión.
- SAP (Service Advertising Protocol): Cada minuto cada servidor manda un broadcast de sus servicios y dirección.

El nivel físico

La velocidad máxima de un canal

- Se puede representar cualquiera señal de datos con una *serie Fourier*. La serie consiste en términos de frecuencias distintas, y se suman los términos para reconstruir la señal.
- Ningún medio de transmisión puede transmitir señales sin perder algún poder. Normalmente un medio puede transmitir las frecuencias desde 0 hasta algún límite f; las frecuencias mayores se atenúan fuertemente.
- Cuanto más cambios por segundo de una señal (la razón de *baud*), tanto más términos de frecuencias altas que se necesitan.
- Entonces, el ancho de banda de un canal determina la velocidad de la transmisión de datos, aun cuando el canal es perfecto.
- Si tenemos un canal de ancho de banda H (en Hertz) y V niveles discretos de señal, la velocidad máxima en un canal perfecto (en bits por segundo) es

  $$v_{max} = 2H \log_2 V$$

  Esto es el teorema de Nyquist.
- Una línea telefónica tiene un ancho de banda de aproximadamente 3000 Hz. No puede transmitir las señales binarias más rápidamente que 6000 bps. ¿Cómo pueden transmitir los módems modernos a velocidades mayores?
- En realidad los canales no son perfectos y sufren del ruido aleatorio. Si el poder de la señal es S y el poder de ruido es R, la *razón de señal a ruido* es S/R. Normalmente se expresa esta razón en los *decibeles (dB)*, que son $10\log_{10}S/R$.
- La velocidad máxima en bps de un canal con ancho de banda H Hz y razón de señal a ruido de S/R es

  $$v_{max} = H \log_2(1+S/R)$$

  Es debido a Shannon.
- Si una línea telefónica tiene un S/R de 30 dB (o 1000), un valor típico, no puede transmitir más de 30.000 bps, independientemente del número de niveles de señal.

Medios de transmisión

- **Medios magnéticos.** Si el costo por bit o ancho de banda es muy importante, las cintas magnéticas ofrecen la mejor opción.
  - Una cinta de video (Exabyte) puede almacenar 7 GB.
  - Una caja de 50 cm puede almacenar 1000 cintas, o 7000 GB.
  - En los Estados Unidos se puede mandar una caja de este tipo de cualquier punto a cualquier otro en 24 horas.
  - El ancho de banda entonces es 648 Mbps. Si el destino es solamente a una hora de distancia, el ancho de banda es más de 15 Gbps.
- **Par trenzado (*twisted pair*).** Consiste en dos alambres de cobre enroscados (para reducir interferencia eléctrica). Puede correr unos kilómetros sin la amplificación. Es usado en el sistema telefónico.

- **Cable coaxial.** Un alambre dentro de un conductor cilíndrico. Tiene un mejor blindaje y puede cruzar distancias mayores con velocidades mayores (por ejemplo, 1-2 Gbps).
- **Fibra óptica.** Hoy tiene un ancho de banda de 50.000 Gbps, pero es limitada por la conversión entre las señales ópticas y eléctricas (1 Gbps). Los pulsos de luz rebotan dentro de la fibra. En una fibra de modo único los pulsos no pueden rebotar (el diámetro es demasiado pequeño) y se necesita menor amplificación (por ejemplo, pueden cruzar 30 km a unos Gbps).

Además de estos hay también medios inalámbricos de transmisión. Cada uno usa una banda de frecuencias en alguna parte del espectro electromagnético. Las ondas de longitudes más cortas tienen frecuencias más altas, y así apoyan velocidades más altas de transmisión de datos. De lambda f = c se deriva la relación entre la banda de longitud de onda y la banda de frecuencia: delta f = (c delta lambda)/lambda^2

- **Radio.** 10 KHz-100 MHz. Las ondas de radio son fáciles de generar, pueden cruzar distancias largas, y entrar fácilmente en los edificios. Son omnidireccionales, lo cual implica que los transmisores y recibidores no tienen que ser alineados.
    - Las ondas de frecuencias bajas pasan por los obstáculos, pero el poder disminuye con el cubo de la distancia.
    - Las ondas de frecuencias más altas van en líneas rectas. Rebotan en los obstáculos y la lluvia las absorbe.
- **Microondas.** 100 MHz-10 GHz. Van en líneas rectas. Antes de la fibra formaban el centro del sistema telefónico de larga distancia. La lluvia las absorbe.
- **Infrarrojo.** Se usan en la comunicación de corta distancia (por ejemplo, controlo remoto de televisores). No pasan por las paredes, lo que implica que sistemas en distintas habitaciones no se interfieren. No se pueden usar afuera.
- **Ondas de luz.** Se usan lasers. Ofrecen un ancho de banda alto con costo bajo, pero el rayo es muy angosto, y el alineamiento es difícil.

El sistema telefónico

- En general hay que usarlo para redes más grandes que un LAN.
- Consiste en las oficinas de conmutación, los alambres entres los clientes y las oficinas (los *local loops*), y los alambres de las conexiones de larga distancia entre las oficinas (los *troncales*). Hay una jerarquía de las oficinas.
- La tendencia es hacia la señalización digital. Ventajas:
    - La regeneración de la señal es fácil sobre distancias largas.
    - Se pueden entremezclar la voz y los datos.
    - Los amplificadores son más baratos porque solamente tienen que distinguir entre dos niveles.
    - La mantención es más fácil; es fácil detectar errores.

Los local loops

- Son analógicos. Los computadores tienen que usar un módem para convertir una señal digital en uno analógica, y en la oficina de compañía de teléfonos un *codec* convierte a digital de nuevo.
- Tres problemas de transmisión:
    - **Atenuación.** Los componentes Fourier diferentes de una señal se atenúan por montos distintos.
    - **Distorsión de retraso.** Los componentes diferentes tienen velocidades diferentes. Dos bits en un cable se pueden entremezclar.

- **Ruido.** Tipos: termal, *cross talk* (inducción entre alambres), y impulsos (de puntos de poder).
- Debido a estos problemas no es deseable tener un gran rango de frecuencias en la señal. Por desgracia las ondas cuadradas de la señalización digital tienen un espectro grande. Por lo tanto los módems transmiten un *portador de onda sinuosidal* y modulan la amplitud, la frecuencia, o la fase.
- Otro problema es los ecos. Frecuentemente se refleja una parte de la señal. Una solución para la voz es un *supresor de eco*, que cambia la línea de full-duplex a half-duplex y cambia el sentido de transmisión rápidamente. Un tono de 2100 Hz puede desactivar los supresores (un ejemplo de la *señalización en banda*). Una alternativa es un *cancelador de eco*, que preserva la transmisión full-duplex y resta una estimación del eco a la señal.
- Al largo plazo hay que convertir los local loops a la fibra, pero es muy caro. Una solución intermedia es instalar la fibra primero solamente en las calles y continuar usar el par trenzado para la conexión al domicilio.

Los troncales y la multiplexación

- El costo de instalar y mantener una línea troncal es casi lo mismo para una línea de ancho de banda bajo como para una línea de ancho de banda alta. Por lo tanto las compañías de teléfonos multiplexan llamadas múltiples en una sola línea de ancho de banda alto.
- **Multiplexación de división de frecuencias (MDF).** Se usan filtros para restringir cada canal telefónico a solamente 3000 Hz. Para asegurar una buena separación se alocan 4000 Hz para cada canal. Se eleva la frecuencia de cada canal de voz y entonces se combinan; cada canal es independiente de los otros.
- **Multiplexación de división de longitud de onda.** Es la misma idea como MDF, pero con luz y fibras. Ya que cada canal en una fibra no puede tener un ancho de más de unos gigahertz (debido a la velocidad máxima de convertir entre señales ópticas y eléctricas), es una buena manera de usar el ancho de banda de cerca 25.000 GHz de una fibra. En este caso los canales entrantes deben tener frecuencias distintas y se combinan con un prisma.
- **Multiplexación de división de tiempo (MDT).** El problema con MDF es que hay que usar circuitería analógica. Por contraste se puede manejar la MDT completamente con la electrónica digital. En MDT cada usuario tiene sucesivamente todo el ancho de banda del canal por un momento. Se puede usar MDT solamente con los datos digitales.

MDT en el sistema telefónico

- El primer paso en el uso de MDT es la conversión de las señales analógicas. Debido al teorema de Nyquist, se puede capturar toda la información de una señal de H Hertz con una frecuencia de muestras de 2H. Un *codec* (coder-decoder) muestrea el flujo 8000 veces por segundo (125 microsegundos por muestra). Este proceso se llama (en el mundo telefónico) *Pulse Code Modulation (PCM)*.
- Un ejemplo de un portador de MDT es una línea T1, que multiplexa 24 canales de voz.
    - Un solo codec muestrea cada canal sucesivamente; cada uno produce 7 bits de dato y 1 bit de control por muestra. Por tanto hay 7×8000 = 56.000 bps de datos por canal, y 8000 bps de control.
    - Cada marco del T1 tiene 24×8 = 192 bits, más un bit para control de marcos. Tenemos 193 bits cada 125 microsegundos, que es 1,544 Mbps.
    - El bit 193 alterna entre 0 y 1. El recibidor lo usa para la sincronización.
- Un T2 (6,312 Mbps) consiste en 4 canales T1, un T3 (44,736 Mbps) de 6 T2, y un T4 (274,176 Mbps) de 7 T3. Cada uno agrega bits de control y de marco.
- *SONET (Synchronous Optical Network)* es un sistema de MDT para la fibra. El marco cada 125 microsegundos tiene 810 bytes, que implica 51,84 Mbps.

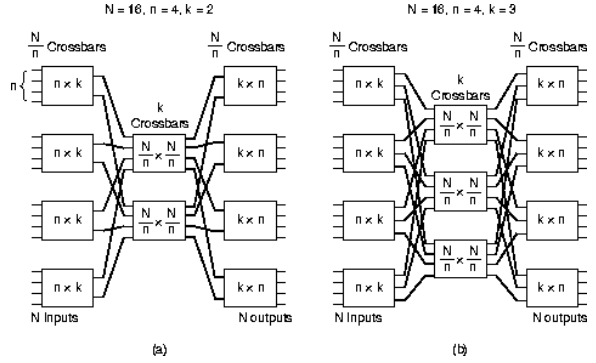
Conmutación

- Los dos tipos principales son la conmutación de circuito y la conmutación de paquetes.
-

	de circuito	**de paquete**
Ruta dedicado de "cobre"	Sí	No
Ancho de banda disponible	Fijo	Dinámico
Posibilidad de malgastar ancho de banda	Sí	No
Transmisión de store-and-forward	No	Sí
Cada paquete toma la misma ruta	Sí	No
Inicialización de la ruta	Necesario	No necesario
Puntos donde la congestión puede ocurrir	En inicialización	Con cada paquete
Cobrar	Por minuto	Por paquete

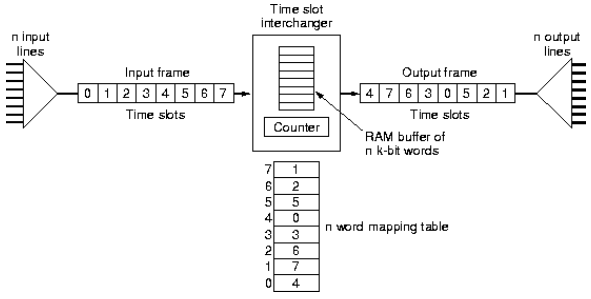
- **Conmutadores de crossbar (travesaño).** Tiene N inputs, N outputs, y N^2 intersecciones. Problema: la escalabilidad. Si N=1000, tenemos 1.000.000 intersecciones.

- **Conmutadores de división de espacio.** Consisten en tres (o más) etapas de conmutadores de crossbar. En la primera etapa hay N/n crossbars con n inputs y k outputs cada uno. En la segunda hay k crossbars de N/n × N/n. La tercera etapa es el revés de la primera.
  - El número de intersecciones es $2kN + k(N/n)^2$. Si N=2000, n=50, y k=10, hay solamente 24.000. Empero permite solamente 200 conexiones simultáneas.
  - Con valores de k mayores hay menor probabilidad de bloqueo, pero el costo del conmutador aumenta.



- **Conmutadores de división de tiempo.** Es digital. La operación tiene unas etapas:
  - Se examinan los n canales de input sucesivamente para construir un marco de input con n entradas de k bits. (En una línea T1 k=8 y se procesan 8000 marcos por segundo.)
  - El intercambiador de entradas de tiempo acepta los marcos de input. Ubica las entradas en orden en una tabla de RAM y entonces lee las entradas a un marco de output usando la tabla de mapping.
  - Se mandan los contenidos del marco de output a los canales de output.

  La limitación de conmutadores de división de tiempo es el tiempo de ciclo de la memoria. Si cada acceso requiere T microsegundos, el tiempo para procesar un marco es 2nT, y debe ser menos de 125 microsegundos. Si T es 100 nanosegundos, n=625. Se puede construir conmutadores con etapas múltiples para solucionar este problema.



Narrowband ISDN

- ISDN es *Integrated Services Digital Network*. Es un servicio inventado en 1984 por las compañías de teléfonos para proveer una conexión digital directamente al cliente. Usa conmutación de circuito. Ahora está disponible en muchos mercados.
- Para la casa ofrece dos canales de 64 kbps para voz/dato y uno de 16 kbps para el control fuera de banda. Para la empresa, 23 o 30 canales de voz/dato en vez de dos.
- Problema: ¡Es demasiado lento! El proceso de estandarización duró años. Durante el mismo período la tecnología de red avanzó rápidamente. Ahora LANs de 10 y 100 Mbps son comunes.
- Sin embargo, un uso interesante es conexiones de Internet de la casa.

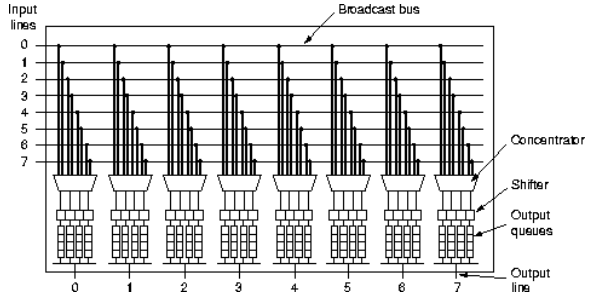
Broadband ISDN y ATM

- Broadband ISDN provee un circuito virtual digital para transferir paquetes de tamaños fijos (celdas) con una velocidad de 155 Mbps. Está basado en ATM (*Asynchronous Transfer Mode*), que es una tecnología de conmutación de paquetes.
- No se pueden usar los conmutadores de división de espacio ni de tiempo con ATM. Tampoco se pueden usar los local loops existentes. La conversión a ATM representa un cambio enorme.
- Broadband ISDN es una combinación de la conmutación de circuito y de paquetes. El servicio es orientado a la conexión pero es implementado con conmutación de paquetes. Hay dos clases de conexiones:
    - **Circuitos virtuales permanentes.** Persisten meses o años.
    - **Circuitos virtuales conmutados.** Temporales, como llamadas de teléfono.
- La creación de un circuito en ATM es el proceso de encontrar un camino por la red. Los conmutadores en la ruta guardan entradas de tabla y tal vez reserven recursos. Cuando un paquete llega en un conmutador, busca qué circuito virtual pertenece en el encabezamiento del paquete y determina en qué línea debiera reenviar el paquete.
- ATM es asíncrono. Por contraste con T1, no hay ningún requerimiento que las celdas de fuentes distintas se alternan rígidamente. Los ordenes aleatorios y incluso brechas en el flujo son permisibles.
- ATM no especifica el medio; ambos los cables y las fibras son posibles. Las conexiones son punto-a-punto y half-duplex. La velocidad principal es 155,52 Mbps; la alternativa es 622,08 Mbps (estas son compatibles con SONET).

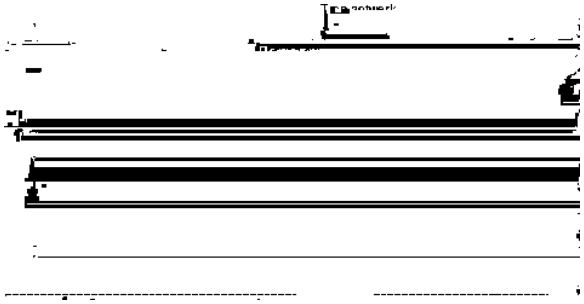
Conmutadores de ATM

- Las celdas de ATM llegan con una velocidad de alrededor de 150 Mbps, o 360.000 celdas por segundo (una celda cada 2,7 microsegundos; con el ATM más rápido, cada 700 nanosegundos). Un conmutador tiene desde 16 a 1024 líneas de input. Para poder construir los conmutadores es necesario que las celdas sean cortas (53 bytes).
- Requerimientos:
    - La taza de perder paquetes debe ser muy baja (1 celda en 10^{12}, por ejemplo).
    - Nunca se puede cambiar el orden de las celdas en un circuito virtual.
- Un problema básico: ¿Qué pasa cuando dos celdas quieren ir por la misma línea de output en el mismo ciclo?

    - No podemos descartar una de las celdas.
    - Podemos usar una cola para cada línea de input. Introduce el efecto de bloqueo de la cabeza de cola: Puede ser celdas que se pueden rutear tras de la cuál está bloqueada.
    - Otra posibilidad es una cola para cada línea de output.
- **Conmutador de knockout.** Tiene un bus de broadcast para cada línea de input. La activación de las intersecciones determinan las líneas de output. Cada línea de output tiene una sola cola virtual que se representa con n reales y un *shifter*. Porque n es normalmente menos que el número de líneas de input, un *concentrador* escoge las celdas a descartar si demasiados llegan.



- **Conmutador de Batcher-banyan.** El problema con el conmutador de knockout es que semejante a un conmutador de crossbar. El Batcher-banyan es un conmutador de etapas múltiples para los paquetes.



Satélites

- Funcionan como repetidores de microondas. Un satélite contiene algunos transponedores que reciben las señales de alguna porción del espectro, las amplifican, y las retransmiten en otra frecuencia.
- Hay tres bandas principales: C (que tiene problemas de interferencia terrenal), Ku, y Ka (que tienen problemas con la lluvia).
- Un satélite tiene 12-20 transponedores, cada uno con un ancho de banda de 36-50 MHz. Un velocidad de transmisión de 50 Mbps es típica. Se usa la multiplexación de división de tiempo.
- La altitud de 36.000 km sobre el ecuador permite la órbita geosíncrona, pero no se pueden ubicar los satélites con espacios de menos de 1 o 2 grados.
- Los tiempos de tránsito de 250-300 milisegundos son típicos.

- Los fuertes del medio son la comunicación broadcast, la comunicación móvil, y la comunicación en los áreas con el terreno difícil o la infraestructura débil. Otra posibilidad es el ancho de banda grande pero temporal.

El nivel de enlace

- El tema principal es los algoritmos para la comunicación confiable y eficiente entre dos máquinas adyacentes.
- Problemas: los errores en los circuitos de comunicación, sus velocidades finitas de transmisión, y el tiempo de propagación.

Asuntos de diseño

Servicios para el nivel de red

- **Servicio sin conexión y sin acuses de recibo.** La máquina de fuente manda marcos al destino. Es apropiado si la frecuencia de errores es muy baja o el tráfico es de tiempo real (por ejemplo, voz).
- **Servicio sin conexión y con acuses de recibo.** El recibidor manda un acuse de recibo al remitente para cada marco recibido. Los acuses son una optimización; el nivel de transporte también los usa, pero con el uso en este nivel se pueden confirmar y posiblemente reenviar los marcos individuales.
- **Servicio orientado a la conexión con acuses de recibo.** Provee un flujo confiable de bits. Las máquinas de fuente y recibo establecen una conexión antes de mandar los datos (inicializar variables, reservar buffers, etc.). Los marcos son numerados, y todos son recibidos exactamente una vez y en el orden correcto.

Marcos

- El nivel físico toma un flujo de bits y intenta entregar al destino. Los bits entregados pueden ser más, menos, o distintos a estos mandados.
- El nivel de enlace trata de detectar y corregir los errores. Normalmente se parte el flujo de bits en *marcos* y se calcula un checksum para cada uno.
- Para partir el flujo no se pueden usar brechas de tiempo en la transmisión, porque no hay ninguna garantía por el nivel físico que estas brechas serán preservadas.
- **Número de caracteres.** Un campo del encabezamiento guarda el número. Pero si el número es cambiado en una transmisión, es difícil recuperar.
- **Caracteres de inicio y fin, con relleno de caracteres.** Cada marco empieza con la secuencia ASCII de DLE STX y termina con DLE ETX. Si la secuencia está en los datos, se duplica el DLE. Pero este sistema es muy vinculado a ASCII y caracteres de 8 bits.
- **Flags de inicio y fin, con relleno de bits.** Cada marco empieza y termina con 01111110. En los datos se inserta un 0 después de cada cinco 1s consiguientes. El recibidor elimina cada 0 después de cinco 1s.
- **Infracciones en el estándar de codificación del nivel físico.** Se usa en sistemas con redundancia. Ejemplo: LANs donde se usa dos bits físicos para cada bit lógico. Entonces quedan dos combinaciones para la señalización.
- Una combinación del número de caracteres con uno de otros métodos es también posible.

Control de errores

- Se usan los acuses de recibo positivos y negativos.
- Para manejar el caso donde se pierde el marco o el acuse, el remitente mantiene temporizadores.
- Para evitar marcos duplicados se usan números de secuencia.

Control de flujo

- Se usan protocolos que prohiben que el remitente pueda mandar marcos sin la permisión implícita o explícita del recibidor.
- Por ejemplo, el remitente puede mandar n marcos y entonces tiene que esperar.

Detección y corrección de errores

- Los errores en los troncales digitales son raros. Pero son comunes en los local loops y en la transmisión inalámbrica.
- En algunos medios (por ejemplo, el radio) los errores ocurren *en grupos* (en vez de individualmente). Un grupo inicia y termina con bits invertidos, con algún subconjunto (posiblemente nulo) de los bits intermedios también invertidos.
    - Ventaja: Si tuviésemos una taza de 0,001 errores por bit y bloques de 1000 bits, la mayoría de los bloques tendrían errores. Pero con los errores en grupos, no.
    - Desventaja: Los errores en grupo son más difíciles de detectar.
- Enfoques:
    - **La corrección de errores.** Transmitir información redundante que permite deducir que debía ser un carácter transmitido.
    - **La detección de errores.** Transmitir solamente suficiente información para detectar un error.
- Términos:
    - Un *codeword* de n bits consiste en m bits de dato y r bits de redundancia o chequeo.
    - La *distancia de Hamming* de dos codewords es el número de bits distintos. Es decir, haga el XOR de los codewords y cuenta el número de unos.
- Normalmente todos los 2^m mensajes de dato son legales, pero no los 2^n codewords debido a la manera en que se calcula los bits de chequeo.
- Se pueden construir todos los codewords legales y entonces encontrar los dos con la distancia de Hamming mínima. Esta es la *distancia de Hamming del código.*
- Para **detectar** d errores se necesita un código de distancia de Hamming de d+1, porque entonces d errores únicos de bit no pueden cambiar un codeword válido a otro codeword válido.
- Para **corregir** d errores se necesita una distancia de 2d+1. Aun cuando hay d cambios, el codeword original todavía está más cerca que cualquier otro.
- Ejemplos:
    - Un código usa un solo bit de paridad que se añade así que el número de unos es par. Tiene una distancia de dos y puede detectar los errores únicos.
    - Un código tiene los cuatro codewords 0000000000, 0000011111, 1111100000, y 1111111111. La distancia es cinco; el código puede corregir dos errores. Por ejemplo, interpreta 0000000111 como 0000011111.
- Supon que queremos corregir los errores de un bit. Necesitamos un código con una distancia de tres. Dado m, ¿qué debe ser r?
    - Hay 2^m mensajes legales. Cada uno tiene n codewords ilegales a una distancia de uno que se forman invirtiendo individualmente cada uno de los n bits del codeword.

    - Entonces cada uno de los 2^m mensajes necesita n+1 patrones de bit propios.
    - Tenemos 2^n codewords posibles. Entonces, $2^m(n+1) <= 2^n$, o usando n = m+r, $m+r+1 <= 2^r$.
    - Dada m tenemos un límite inferior del número de bits de chequeo r. El código de Hamming logra este límite.
- Se pueden usar un código de corrección para los errores de un bit para corregir los errores en grupo usando un truco: Transmite los datos como las columnas de un matriz donde cada fila tiene su propio bits de chequeo.

Códigos de detección de errores

- Aunque se usa a veces la corrección de errores, normalmente se prefiere la detección de errores ya que es más eficiente.
- Por ejemplo, asume que tenemos un canal con una taza de errores de 10^{-6} por bit (es decir, un bit en cada 10^6). Usamos mensajes de 1000 bits de dato.
    - Para la corrección debemos añadir 10 bits por mensaje. En la transmisión de 10^6 bits de dato mandamos 10.000 bits de chequeo para detectar y corregir el un bit de error que esperamos.
    - Para la detección usamos solamente un bit de paridad por mensaje. Para 10^6 bits de dato usamos solamente 1000 bits. Pero uno de los mensajes tiene un error, así que tenemos que retransmitirlo con su bit de paridad (1001 bits). En total usamos 2001 bits para este esquema.
- Si usamos un solo bit de paridad y tenemos un grupo de errores, la probabilidad de detección es solamente 1/2. Podemos aumentar la capacidad a detectar un grupo de errores usando el truco de la transmisión de un matriz de k×n: Se transmiten los datos por columna, y cada fila tiene un bit de paridad. Podemos detectar los errores en grupo hasta n, el número de filas. No podemos detectar n+1 si el primer bit y el último bit son invertidos y todos los otros son correctos. Si tenemos muchos errores en el bloque la probabilidad de aceptarlo es solamente $(1/2)^n$.

Códigos de CRC

- Un método más usado es el código polinomial (también llamado el *cyclic redundancy code*, o CRC). Se trata los strings de bits como polinomios con coeficientes de solamente 0 y 1. Un mensaje de k bits con un *grado* de k-1 corresponde a

  $$bit_0x^{k-1} + ...+ bit_nx^{k-1-n} + ...+ bit_{k-1}x^0$$

  La aritmética con estos polinomios es módulo 2 sin llevar, es decir la adición y la sustracción son equivalentes a XOR. La división usa XOR en ves de sustracción y A se divide en B si el número de bits en B es mayor de o igual a el número en A.
- El remitente y el recibidor usan el mismo *polinomio de generación*, G(x), con bits alto y bajo de 1.
- Para calcular el checksum de r bits, que también es el grado de G(x),
    - Añade r bits de 0 a M(x), el mensaje, produciendo $x^rM(x)$.
    - Divide $x^rM(x)$ por G(x), produciendo un resto.
    - Transmite $T(x) = x^rM(x)$ - *resto*. T(x) es divisible por G(x). Sus últimos r bits son el checksum.
- Si hay errores en la transmisión recibiremos T(x)+E(x) en vez de T(x). El recibidor divide T(x)+E(x) por G(x). Ya que el resto debido a T(x) es 0, el resto obtenido es completamente debido a E(x). Si E(x) tiene G(x) como un factor, el resto será 0 y no detectaremos el error, de otro modo, sí.
- Si hay un error de un bit, $E(x) = x^i$. Si G(x) tiene más de un término, no puede dividir E(x). Entonces podemos detectar todos los errores de un bit.

- Con dos errores tendremos $E(x) = x^i + x^j = x^j(x^{i-j}+1)$. Podemos usar un G(x) que no divide x^k+1 para cualquier k hasta el valor máximo de i-j (que es la longitud del marco). Por ejemplo, $x^{15}+x^{14}+1$ no divide x^k+1 para k<32768.
- Si x+1 es un factor de G(x), podemos detectar todos los errores que consisten en un número impar de bits invertidos. Prueba por contradicción: Asume que E(x) tiene un número impar de términos y es divisible por x+1. Entonces E(x) = (x+1)Q(x) por algún Q(x). E(1) = (1+1)Q(1) = (0)Q(1) = 0. Pero E(1) debe ser 1 porque consiste en la suma de un número impar de 1's.
- Podemos detectar todos los errores en grupo con longitudes menos de o igual a r. Si el grupo tiene una longitud de k, lo podemos escribir como $x^i(x^{k-1}+...+1)$ (i ubica el grupo en el marco). Si G(x) contiene un término de x^0, x^i no puede ser un factor y G(x) no puede ser igual a $x^{k-1}+...+1$ (el grado k-1 es menos de r). Si el grupo tiene una longitud de r+1, la probabilidad que el grupo es G(x) es la probabilidad que los r-1 bits intermedios del grupo son iguales (por definición el primer y el último bits del grupo son 1), que es $(1/2)^{r-1}$.
- Para los grupos con longitudes mayor de r+1, la probabilidad es $(1/2)^r$.
- Estándares internacionales:
  - CRC-12 = $x^{12} + x^{11} + x^3 + x^2 + x + 1$
  - CRC-16 = $x^{16} + x^{15} + x^2 + 1$
  - CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$

  Los dos últimos detectan todos los errores de uno y dos bits, los errores con un número impar de bits invertidos, los grupos de errores con longitudes menos de o igual a 16, 99,997% con longitudes de 17, y 99,998% con longitudes mayor o igual a 18.

Protocolos elementales de enlace

- Suposiciones iniciales:
  - Máquina A quiere mandar un flujo de datos a B usando un servicio confiable orientada a la conexión. El servicio es simplex.
  - Los niveles de red de A y B están siempre listos. Por ejemplo, A siempre tiene datos listos para mandar.
  - El canal nunca pierde ni daña los marcos.
  - El hardware hace el checksum (necesario cuando eliminamos la condición precedente).

```
sender1()
{
  frame s;
  packet buffer;

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

```
receiver1()
{
  frame r;
  event_type event;

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

- Cambio: El nivel de red de remitente no puede aceptar siempre los datos. Debemos prevenir que el remitente satura el recibidor.
  - Insertar un retraso fijo no es deseable, porque fija el caso peor como el caso normal.
  - El recibidor manda un acuse de recibo después de cada marco; el remitente tiene que esperar el acuse (un protocolo de *parar-y-esperar*).
  - El medio tiene que ser half-duplex aunque el servicio todavía es simplex.

```
sender2:
  while (true)
```

```
receiver2:
  while (true)
```

```
    from_network_layer(&buffer)
    s.info = buffer
    to_physical_layer(&s)
    wait_for_event(&event)
```

```
    wait_for_event(&event)
    from_physical_layer(&r)
    to_network_layer(&r.info)
    to_physical_layer(&s)
```

- Cambio: Se pueden dañar o perder los marcos.
  - Una posibilidad: Añadir un temporizador al remitente. El remitente manda el marco, y el recibidor manda un acuse solamente si los datos fueron recibidos correctamente (se descartan los marcos dañados). Después de algún tiempo el remitente manda el marco de nuevo. ¿Cuál es el problema? Se pueden duplicar los marcos si un acuse es perdido.
  - Solución: El remitente inserta un número de secuencia en en encabezamiento del marco. Este permite que el recibidos pueda distinguir entre marcos nuevos y duplicados.
  - ¿Cuántos bits necesitamos para el número de secuencia? Necesitamos distinguir solamente entre m y m+1 después de que el recibidor manda el acuse de m (depende de que el acuse es recibido por el remitente o no). No podemos recibir el marco m+2 porque el recibidor no va a mandar el acuse de m+1 (que permite al remitente mandar el marco m+2) hasta que reciba el marco m+1. Entonces se necesita solamente un bit para distinguir entre las dos posibilidades.

```
sender3:
  next_frame_to_send = 0
  from_network_layer(&buffer)
  while (true)
    s.info = buffer
    s.seq = next_frame_to_send
    to_physical_layer(&s)
    start_timer(s.seq)
    wait_for_event(&event)
    if (event == frame_arrival)
      from_network_layer(&buffer)
      inc(next_frame_to_send)
```

```
receiver3:
  frame_expected = 0
  while (true)
    wait_for_event(&event)
    if (event == frame_arrival)
      from_physical_layer(&r)
      if (r.seq == frame_expected)
        to_network_layer(&r.info)
        inc(frame_expected)
      to_physical_layer(&s) // ACK
```

El período de espera en el remitente tiene que ser suficiente largo para prevenir las expiraciones prematuras. ¿Por qué?

Protocolos de *ventana deslizante*

- Hemos tenido la restricción que la transmisión de datos es simplex, aunque el canal de transmisión es dúplex o full-dúplex. Esta restricción malgasta la mitad del ancho de banda del canal.
- Nuestro primer paso es eliminar esta restricción. Ambos los marcos de datos y los acuses de recibo pueden fluir en ambas direcciones. Podemos usar un campo en el encabezamiento de cada marco que especifica su clase (datos o acuse).
- Un segundo mejoramiento es lo siguiente: Cuando un marco de datos llega, el recibidor no manda inmediatamente un acuse. En vez de esto, espera hasta que su nivel de red le pase un paquete de datos (recuerda que el flujo de datos ahora es full-dúplex). Cuando el paquete está listo, se añade el acuse para el marco recibido antes. Entonces, se combinan en un solo marco un acuse y un paquete de datos. Se llama este proceso *piggybacking*.
- Ventajas principales de piggybacking:
  - Mejor uso del ancho de banda. El campo de acuse de recibo en un encabezamiento de marco de datos es solamente pocos bits, mientras que un marco que es solamente un acuse consiste en los bits, un encabezamiento, y un checksum.
  - Menos marcos mandados implica menos interrupciones de llegada de marco en el recibidor, y quizás menos buffers en el recibidor también.

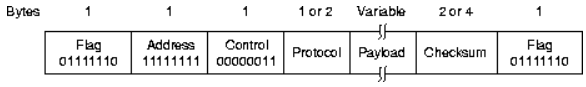
- ¿Cuál es el lado negativo de piggybacking? Introduce una nueva complicación. ¿Por cuánto tiempo debiera esperar el nivel de enlace para un paquete de datos antes de que mande un acuse? Si espera demasiado largo, el remitente retransmitirá el marco.
- En protocolo 3 teníamos también problemas si el mandador no espera por suficiente tiempo. Queremos un protocolo que puede continuar funcionando en tal caso. Una clase de protocolos que son mas robustos son los protocolos de *ventana deslizante.*
- En estos protocolos cada marco tiene un número de secuencia con un rango de 0 hasta algún n. En cualquier instante, el mandador mantiene un conjunto de números de secuencia que corresponden a los marcos que puede mandar (los marcos dentro de la *ventana del mandador*). El recibidor mantiene una *ventana del recibidor* que es el conjunto de marcos que puede aceptar. Las dos ventanas pueden tener tamaños distintos.
    - Los números de secuencia en la ventana del mandador representan los marcos mandados pero no dados acuses. Cuando un paquete llega desde el nivel de red, se le asigna el próximo número de secuencia (modulo n) y la ventana crece por uno. Cuando un acuse llega, la ventana disminuye por uno. En esta manera la ventana mantiene siempre la lista de marcos sin acuses. (Nota: Requerimos que el nivel físico entrega marcos en el orden mandado.)
    - Ya que es posible que se pierdan o se dañen los marcos en la ventana del mandador, el mandador los tiene que guardar para una retransmisión eventual. Con una ventana de tamaño n el mandador necesita n buffers. Si la ventana crece a su tamaño máximo, el nivel de enlace de mandador no puede aceptar paquetes nuevos desde el nivel de red hasta que un buffer está vacío.
    - En el recibidor la ventana siempre mantiene el mismo tamaño y representa los marcos que se pueden aceptar. El recibidor tiene un buffer para cada posición en la ventana; el propósito de estos buffers es permitir que se pueden entregar los marcos en orden al nivel de red aun cuando los marcos llegan en un orden distinto (debido a retransmisiones, marcos perdidos, etc.). Se descartan los marcos que llegan con números de secuencia fuera de la ventana.
- El caso más sencillo es un protocolo con ventanas de tamaños máximos de 1. Cuando un marco llega cuyo número de secuencia es igual al número único en la ventana, se genera un acuse de recibo, se entrega el paquete a nivel de red, y se desplaza la ventana arriba por uno (es decir, invertir el número en la ventana).
- Tal protocolo es de tipo parar-y-esperar, ya que el mandador transmite un marco y espera su acuse antes de mandar el próximo. ¿Qué pasa en este tipo de protocolo cuando el tiempo de ida y vuelta es largo (por ejemplo, con una conexión de satélite)? El mandador malgasta ancho de banda esperando los acuses de recibo. Ejemplo:
    - Canal de satélite de 50 kbps con una latencia de 250 msegs y marcos de 1000 bits.
    - En t=0 mandamos un marco. En t=20 msegs es completamente mandado.
    - Llega completamente en t=270. El acuse llega en t=520.
    - El mandador espera 500/520 = 96% del tiempo.
- Solución: *pipelining*. El mandador puede mandar hasta n marcos sin esperar. Si n es suficiente grande los acuses llegarán siempre antes de que el mandador haya mandado n marcos (es decir, el tamaño de su ventana).
- Con el pipelining tenemos un problema si se daña o se pierde un marco en la mitad del flujo. Muchos marcos llegarán en el recibidor antes de que el mandador sepa que hay un error. Cuando el recibidor nota un marco dañado o faltando, ¿qué debiera hacer con los marcos correctos que siguen?
    - **Repetir n.** En esta estrategia el recibidor descarta los marcos que siguen y no manda ningún acuse. Es decir, el recibidor tiene una ventana de tamaño 1 y no acepta ningún marco excepto el próximo que le debe dar a nivel de red. Con tiempo el mandador notará la ausencia de acuses y retransmitirá los marcos en orden. Este enfoque malgasta ancho de banda si la taza de errores es alta.
    - **Repetir selectivamente.** En esta estrategia el recibidor guarda los marcos correctos después del malo. Cuando el mandador nota que falta un acuse, solamente retransmitirá el marco malo. Si esta retransmisión tiene éxito, el recibidor tendrá muchos marcos correctos en secuencia y le podrá pasar al nivel de red y mandar un acuse de recibo para el número de secuencia más alto. Este enfoque requiere una ventana del recibidor con un tamaño más de 1, y su costo es el espacio para

los buffers. El tamaño de la ventana debiera ser la mitad del número de números de secuencia. ¿Por qué? El recibidor tiene que poder distinguir entre marcos duplicados y marcos nuevos.

- Se pueden usar acuses negativos de recibo (NAKs), los cuales el recibidor puede usar para acelerar la retransmisión de marcos.

SLIP y PPP

- Dos protocolos populares para las conexiones de Internet temporales entre los PCs de usuarios y los proveedores de servicios de Internet son SLIP y PPP. Se pueden usar estos protocolos también para las conexiones entre ruteadores sobre líneas dedicadas en la subred de la Internet.
- **SLIP** (de 1984) es muy sencillo. Se mandan paquetes crudos de IP (es decir, paquetes del nivel de la red) sobre la línea, con un byte de flag al fin para indicar los marcos. Se usa el relleno de caracteres para el caso donde el byte de flag ocurre dentro de los datos de un paquete. Algunas versiones nuevas de SLIP apoyan la compresión de encabezamientos de TCP y IP, aprovechando del hecho que los paquetes consecutivos normalmente tienen muchos campos comunes. Pero SLIP tiene algunos problemas:
    - No provee ninguna detección de errores.
    - Apoya solamente IP.
    - Cada lado tiene que saber la dirección IP del otro antes.
    - No provee ningún sistema de autenticación.
    - Hay muchas versiones incompatibles.
- **PPP** se aplica a las deficiencias de SLIP. Provee tres cosas:
    - Un sistema de enmarcar que distingue el fin de un marco y el inicio del próximo, que también maneja la detección de errores.
    - Un protocolo de control de enlace (LCP, Link Control Protocol) para subir una conexión, probarla, negociar opciones, y bajarla.
    - Un método para negociar opciones del nivel de red que es independiente del protocolo de nivel de red usado. Hay un protocolo de control de red (NCP, Network Control Protocol) para cada clase de nivel de red apoyada.



- Un marco de PPP:

    - El campo de dirección es normalmente constante.
    - El campo de control también es normalmente constante y indica un servicio de datagrama (no confiable y sin acuses de recibo), pero se pueden negociar servicios confiables con marcos enumerados en ambientes ruidosos.
    - El campo de protocolo indica el tipo de paquete (por ejemplo, LCP, NCP, IP, AppleTalk, etc.).
    - El campo de datos tiene una longitud variable hasta algún máximo negociado.

- Se establece una conexión de PPP en unos pasos. La línea empieza en el estado *DEAD* (muerto) con ninguna conexión física. Después de que la conexión física exista, el estado es *ESTABLISHED* (establecido). Ahora se negocian las opciones de LCP, que (si tiene éxito) produce el estado *AUTHENTICATE* (autenticar). Si este tiene éxito se entra el estado de *NETWORK* (red), donde el NCP configura el nivel de la red. Este produce el estado de *OPEN* (abierto), y se pueden intercambiar datos. Después del transporte de datos se entra el estado de *TERMINATE* (terminar) y desde allí a *DEAD* de nuevo.

El nivel de enlace de ATM

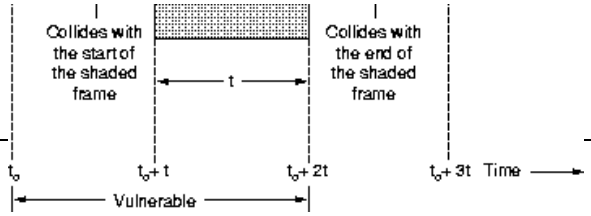
- Este nivel se llama el subnivel de TC (*transmission convergence*, o convergencia de transmisión).
- Este nivel recibe celdas del nivel arriba (que se llama el nivel de ATM). Ellos tienen un encabezamiento de 4 bytes seguido por un checksum de 1 byte. Este checksum de CRC es solamente para el encabezamiento, no para los datos. ¿Por qué?
    - Se quiero reducir la probabilidad que una celda es entregada al destino equivocado.
    - Se quiero evitar el costo de hacer un checksum para los datos. En algunas aplicaciones (por ejemplo, transmisión de video) errores en los datos no son tan importantes. Si se necesita más confianza en los datos, los niveles más altos tienen que proveerla.
    - ATM es dirigido a la fibra, que tiene pocos errores.
- Si el medio físico es síncrono, el subnivel de TC controla la inyección de las celdas en él. También genera cualquiera información de marco necesitada por el medio; esto es necesario con SONET, por ejemplo.
- La tarea más difícil del subnivel de TC del recibidor es identificar los límites de las celdas que llegan. No hay ningún byte de flag. A veces el nivel físico ayuda (por ejemplo, en SONET) pero no siempre. La solución:
    - Se usan el encabezamiento y su checksum. El subnivel de TC mantiene un registro de desplazamiento de 40 bits. Desplaza un bit a la vez hasta que tiene un checksum válido para el encabezamiento.
    - Ya que la probabilidad de encontrar un checksum válido en una secuencia aleatoria de bits es solamente 1/256, se repite la prueba con n celdas siguientes antes de decidir que se hayan encontrado la sincronización.
    - Para identificar la pérdida de la sincronización, el subnivel de TC espera hasta que m celdas tienen checksums malos.
    - Para prevenir que un usuario inserte una secuencia de bits que parecen como encabezamientos y checksum, se arreglan de otro modo los bits de dato para la transmisión.
- Nota que el subnivel de TC usa un encabezamiento creado por el nivel arriba para encontrar los marcos. Esto es una violación de las reglas de buen diseño de protocolos.

Redes de broadcast

- En una red de broadcast la cuestión principal es como determinar quien usa un canal para el cual existe competencia. Los protocolos para esto pertenecen a un subnivel del nivel de enlace que se llama el subnivel de *MAC* (Medium Access Control, o control de acceso al medio). Es muy importante en las LANs, que normalmente usan canales de broadcast.
- Se puede asignar un solo canal de broadcast usando un esquema *estático* o *dinámico.*
- **Asignación estática.** Se usa algún tipo de multiplexación (MDF o MDT) para dividir el ancho de banda en N porciones, de que cada usuario tiene uno. Problemas:
    - Si menos de N usuarios quieren usar el canal, se pierde ancho de banda.
    - Si más de N usuarios quieren usar el canal, se niega servicio a algunos, aun cuando hay usuarios que no usan sus anchos de banda alocados.
    - Porque el tráfico en sistemas computaciones ocurre en ráfagas, muchos de los subcanales van a estar desocupados por mucho del tiempo.
- **Asignación dinámica.** Usa el ancho de banda mejor. Hay muchos protocolos basados en cinco suposiciones principales:
    - **Modelo de estación.** Hay N estaciones independientes que generan marcos para la transmisión. La probabilidad de generar un marco en el período delta t es lambda delta t, donde lambda es un constante. Después de generar un marco una estación hace nada hasta que se transmita el marco con éxito.
    - **Canal único.** Hay un solo canal disponible para la comunicación. Todos pueden transmitir usándolo y pueden recibir de él.
    - **Choques.** Si se transmiten dos marcos simultáneamente, se chocan y se pierden ambos. Todas las estaciones pueden detectar los choques.
    - **Tiempo continuo o dividido.** En el primer caso se puede empezar con la transmisión de un marco en cualquier instante. En el segundo se parte el tiempo con un reloj de maestro que las transmisiones empiezan siempre al inicio de una división.
    - **Detección del portador o no.** Las estaciones pueden detectar que el canal está en uso antes de tratar de usarlo, o no. En el primer caso ninguna estación trataré transmitir sobre una línea ocupada hasta que sea desocupada. El el último las estaciones transmiten y solamente luego pueden detectar si hubo un choque.

ALOHA

- Desarrollado en los años 70 en Hawaii, ALOHA es un sistema de broadcast que usa el radio. Hay dos versiones, ALOHA puro y ALOHA dividido, que son distintos en el tratamiento del tiempo.
- **ALOHA puro.** Los usuarios pueden transmitir marcos en cualquier instante. Habrá choques, y los marcos que chocan se destruirán. Empero, un mandador siempre puede detectar un choque escuchando al canal.
    - Con una LAN el feedback es instantáneamente, pero con un satélite hay un retraso de 270 msegs.
    - Si hubo un choque el mandador espera un período aleatorio y manda el marco de nuevo. ¿Por qué aleatorio? Porque si no, los mismos marcos chocarán repetidamente.
    - Este tipo de sistema donde usuarios múltiples comparten un canal en una manera que puede producir conflictos se llama un *sistema de contienda.*
- En ALOHA puro un traslapo de solamente un bit entre dos marcos es suficiente para destruir ambos. Dado que los usuarios pueden transmitir en cualquier instante, una pregunta interesante es cuál es la utilización máxima del canal.



- Asume que hay un número infinito de usuarios que pueden transmitir marcos. Todos los marcos tienen el mismo tamaño, y el tiempo para transmitir un marco es t.
- Asume que en un período t el número de intentos de transmisión tiene una distribución Poisson con una media G. Nota que las transmisiones pueden ser tanto marcos nuevos como retransmisiones de marcos que chocaron.
- Si la probabilidad que un marco no choque en un período t es P_0, la utilización es $S = GP_0$. Podemos ver S como el número de marcos *nuevos* que los usuarios pueden producir por t para un estado en equilibrio.
- Si queremos transmitir con éxito un marco en tiempo t_0+t, necesitamos que no se inicie la transmisión de otro marco entre t_0 y t_0 + 2t (de otra manera habría un traslapo). Ve el dibujo.

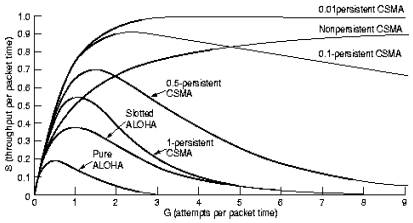
- La probabilidad que se genera 0 marcos en un período t es dada por la distribución Poisson como e^{-G}. En un período de 2t es $e^{-G}e^{-G} = e^{-2G}$. Esto es P_0. Por lo tanto, $S = Ge^{-2G}$.
- La utilización máxima ocurre con G = 0,5, con S = 1/2e, o 0,184. Es decir, la utilización máxima del canal es solamente 18% cuando todos pueden transmitir en cualquier instante.

- **ALOHA dividido.** Este sistema dobla la capacidad de ALOHA puro. Se divide el tiempo en intervalos discretos; cada intervalo corresponde a un marco. Los usuarios están de acuerdo en los límites de los intervalos (por ejemplo, tienen un reloj de maestro). Se pueden transmitir los marcos solamente a los inicios de los intervalos.
  - Para evaluar la utilización de este método, podemos usar el mismo análisis como con ALOHA puro.
  - Ahora para evitar un traslapo solamente necesitamos que no se inicie la transmisión de otro marco en el intervalo de t_0+t (es decir, en el mismo intervalo en que transmitimos el marco de interés).
  - Por lo tanto la probabilidad que se genera ningún marco en el período vulnerable es e^{-G}. Entonces $S = Ge^{-G}$. Si G=1, S=0,368. Entonces 37% de los intervalos están vacíos, 37% tienen un marco, y 27% son choques.
  - Con G mayor tenemos menos intervalos vacíos pero el número de choques crece de manera exponencial.

Protocolos de acceso múltiple con sentido de portador

- La mejor utilización con ALOHA fue solamente 1/e. Esto quizás no sea sorpresa, cuando consideramos que nadie presta atención a las acciones de otros. En las redes de área local las estaciones pueden detectar que pasa en la media y adaptar su comportamiento de acuerdo con esto. Los protocolos de este tipo se llaman *protocolos con sentido de portador.*
- **CSMA de persistencia 1.** CSMA es Carrier Sense Multiple Access, o acceso múltiple con sentido de portador. Cuando una estación tiene datos para mandar, primer examina si alguien está usando el canal. Espera hasta que el canal esté desocupado y entonces transmite un marco. Si hay un choque, espera un período aleatorio y trata otra vez.
  - *Persistencia 1* significa que la estación transmite con una probabilidad de 1 cuando encuentra el canal desocupado.
  - El retraso de propagación tiene gran afecto al rendimiento del protocolo. Hay una probabilidad que poco después una estación manda un marco otra estación también trata de mandar un marco. Si la señal de la primera no ha llegado a la segunda, la segunda detectará un canal desocupado, mandará su marco, y producirá un choque. Cuanto más grande el retraso, tanto peor el rendimiento del protocolo.

    - Aunque todavía hay la probabilidad de choques, este protocolo es mejor que ALOHA porque las estaciones no transmiten en la mitad de otra transmisión.
- **CSMA sin persistencia.** Es menos ávido que el protocolo anterior. Antes de mandar prueba el canal y manda si nadie lo está usando. Si el canal está ocupado, no lo prueba constantemente hasta que esté desocupado, sino espera un período aleatorio y repite el algoritmo. La utilización es mejor pero los retrasos para mandar los marcos son más largos.
- **CSMA de persistencia p.** Es para los canales con tiempo dividido. Si el canal está desocupado, transmite con una probabilidad de p. Con una probabilidad de 1-p espera hasta el próximo intervalo y repite el proceso. Se repite el proceso hasta que se mande el marco o haya un choque, en cual caso espera un período aleatorio y empieza de nuevo. Si el canal originalmente está ocupado el protocolo espera hasta el próximo intervalo y entonces usa el algoritmo.



Protocolos de CSMA con la detección de choques

- Los protocolos de CSMA son un mejoramiento sobre ALOHA porque aseguran que ninguna estación transmite cuando detecta que el canal está ocupado. Un segundo mejoramiento es que las estaciones terminan sus transmisiones tan pronto como detectan un choque. Esto ahorra tiempo y ancho de banda. Los protocolos de esta clase se llaman CSMA/CD (Carrier Sense Multiple Access with Collision Detection, o CSMA con la detección de choques).
- Después de detectar un choque, una estación termina su transmisión, espera un período aleatorio, y trata de nuevo.
- Los choques ocurren en el *período de contienda*. La duración de este período determina el retraso y la utilización del canal. Para asegurar que tiene control del canal, ¿cuánto tiempo debe esperar una estación? Supongamos que el tiempo para una señal propagar entre las dos estaciones más separadas es t.
    - Esperar solamente t no es suficiente. ¿Por qué?
    - Supon que una estación transmite un bit al tiempo t_0. Poco antes t_0+t, otra estación, que no ha recibido el bit transmitido, manda su bit propio. Inmediatamente detecta que hay un choque, pero la primera estación no detectará el choque hasta t_0+2t.
    - En un cable coaxial de 1 km, t = 5 microsegundos.

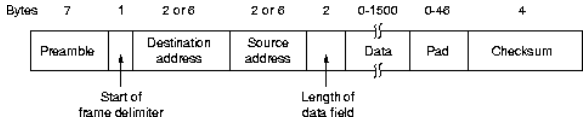
Protocolos libre de choques

- Los períodos de contienda son un problema mayor cuando los cables son largos (retraso de propagación es mayor) y los marcos son cortos (el overhead del los periodos de contienda juega un papel mayor). Esto es el caso en las redes de fibra óptica.
- Podemos eliminar completamente la posibilidad de choques en el período de contienda. En estos protocolos asumimos que las N estaciones tienen direcciones únicas de 0 a N-1.
- **Protocolo de bit-map.** El período de contienda consiste en N intervalos. Si la estación 0 tiene un marco para mandar, transmite un bit de 1 en el intervalo 0. Ninguna otra estación puede transmitir en este intervalo. En general, la estación j transmite un bit de 1 en el intervalo j si tiene un marco para mandar. Después de los N intervalos del período de contienda, todas las estaciones saben cuales quieren transmitir marcos. En este punto transmiten en orden.
    - Porque todas están de acuerdo en el orden de transmisiones, nunca habrá choques.
    - Después de la última transmisión de un marco, un nuevo período de contienda empieza.
    - El overhead es solamente un bit por marco.
- **Protocolo de cuenta atrás binaria.** Reduce el overhead usando direcciones binarias para las estaciones. El período de contienda ahora tiene solamente $\log_2 N$ intervalos, que es la longitud de la dirección de cada estación (por ejemplo, direcciones son 0000, 0001, etc.). Cada estación que quiere mandar un marco transmite el bit más alta de su dirección en intervalo 0, el próximo en intervalo 1, etc. Se hacen un OR de los bits en el canal. Cuando una estación ve un 1 en un intervalo en que el bit que transmitió fue 0, abandona el intento de transmitir en este turno. Finalmente solamente la estación con la dirección más alta transmite su marco después del período de contienda.
    - Si en el formato de los marcos la dirección del mandador es el primer campo, ¡este protocolo tiene ningún overhead!
    - Para mantener la justicia del algoritmo se cambia la asignación de las direcciones. Después de transmitir se asigna una estación la dirección 0 y aumentan las otras.

IEEE 802.3 y Ethernet

- IEEE 802.3 es un protocolo de CSMA/CD con persistencia de 1 para las LANs.
    - Cuando una estación quiere transmitir, escucha al cable.
    - Si el cable está ocupado, la estación espera hasta que esté desocupado; de otra manera transmite inmediatamente.
    - Si hay un choque, las estaciones involucradas esperan por períodos aleatorios.
- Historia:
    - Después de ALOHA y el desarrollo del sentido de portador, Xerox PARC construyó un sistema de CSMA/CD de 2,94 Mbps para conectar más de 100 estaciones de trabajo en un cable de 1 km. Se llamaba *Ethernet* (red de éter).
    - Xerox, DEC, y Intel crearon un estándar para un Ethernet de 10 Mbps. Esto fue el baso para 802.3, que describe una familia de protocolos de velocidades de 1 a 10 Mbps sobre algunos medios.
- Cables:
    - **10Base5** (Ethernet gruesa). Usa un cable coaxial grueso y tiene una velocidad de 10 Mbps. Los segmentos pueden ser hasta 500 m en longitud con hasta 100 nodos. Se hacen las conexiones usando *derivaciones de vampiro:* se inserta un polo hasta la mitad del cable. La derivación es dentro un *transceiver*, que contiene la electrónica para la detección de portadores y choques. Entre el transceiver y el computador es un cable de hasta 50 m. A veces se pueden conectar más de un computador a un solo transceiver. En el computador hay un controlador que crea marcos, hace checksums, etc.
    - **10Base2** (Ethernet delgada). Usa un cable coaxial delgado y dobla más fácilmente. Se hacen las conexiones usando conectores de T, que son más fáciles para instalar y más confiables. Ethernet

delgada es más barata y más fácil instalar pero los segmento pueden ser solamente 200 m con 30 nodos. En 10Base2 el transceiver está en el computador con el controlador.
    - La detección de derivaciones malas, rupturas, y conectores flojos es un gran problema con ambas. Un método que se usa es la medición de la propagación y la reflexión de un pulso en el cable.
    - **10Base-T.** Simplifica la ubicación de rupturas. Cada estación tiene una conexión con un *hub* (centro). Los cables normalmente son los pares trenzados. La desventaja es que los cables tienen un límite de solamente 100 m, y también el costo de un hub puede ser alto.
    - **10Base-F.** Usa la fibra óptica. Es cara pero buena para las conexiones entre edificios (los segmentos pueden tener una longitud hasta 2000 m).
    - Para eliminar el problema con las longitudes máximas de los segmentos, se pueden instalar *repetidores* que reciben, amplifican, y retransmiten las señales en ambas direcciones. La única restricción es que la distancia entre cualquier par de transceivers no puede ser más de 2,5 km y no puede haber más de cuarto repetidores entre transceivers.
- **Codificación de Manchester.** En 802.3 no hay ningún reloj de maestro. Este produce un problema en la detección de bits distintos (por ejemplo, ¿cómo se detectan dos bits de 0 en vez de tres?). En la *codificación de Manchester* se usan dos señales para cada bit. Se transmite un bit de 1 estableciendo un voltaje alto en el primer intervalo y un voltaje bajo en el segundo (un bit de 0 es el inverso). Porque cada bit contiene una transición de voltajes la sincronización es sencilla.



- Marcos:

    - El preámbulo es 7 bytes de bits que se alternan. La codificación de Manchester de esto produce una onda que el recibidor puede usar para sincronizar su reloj con el mandador. Después está el inicio del marco.
    - La dirección de destino puede tener un bit alto de 1, que indica la dirección de un grupo. Todas las estaciones reciben los marcos que tienen este bit encendido, lo que permite el *multicast*. Una dirección de todos unos es para el broadcast. El próximo bit distingue entre las direcciones locales y las globales, que son únicas en el mundo.
    - La longitud no puede ser 0; un marco debe ser por lo menos 64 bytes. Hay dos razones. Simplifica la distinción entre marcos válidos y basura producida por choques. Más importante permite que el tiempo para mandar un marco es suficiente para detectar un choque con la estación más lejana. Para una LAN de 10 Mbps con una longitud máxima de 2500 metros y cuatro repetidores, el marco mínimo debe tomar 51,2 microsegs, que corresponde a 64 bytes. Se rellena si no hay suficientes datos. Nota que con redes más rápidas se necesitan marcos más largo o longitudes máximas más cortas.
    - El checksum es CRC.
- **Algoritmo de retiro de manera exponencial binaria.** Después de un choque se divide el tiempo en intervalos de 2t, que es 51,2 microsegs. Después del choque i cada estación elige un número aleatorio entre 0 y 2^i-1 (pero con un máximo de 1023) y espera por un período de este número de intervalos. Después de

16 choques el controlador falla. Este algoritmo adapta automáticamente al número de estaciones que están tratando de mandar.

- Con más y más estaciones y tráfico en una LAN de 802.3, se satura la LAN. Una posibilidad para aumentar el rendimiento del sistema sin usar una velocidad más alta es una LAN 802.3 conmutada.
    - El conmutador consiste en un backplane en que se insertan 4 a 32 tarjetas que tienen uno a ocho puertas de (por lo general) 10BaseT.
    - Cuando un marco llega en la tarjeta, o se reenvía a una estación conectada a la misma tarjeta o se reenvía a otra tarjeta.
    - En un diseño cada tarjeta forma su propio *dominio de choques.* Es decir, cada tarjeta es un LAN, y todas las tarjetas pueden transmitir paralelamente.
    - Otro diseño es que cada puerta forma su propio dominio de choques. La tarjeta guarda los marcos que llegan en RAM y los choques son raros. Este método puede aumentar el rendimiento de la red un orden de magnitud.
    - Se pueden conectar un hub a una puerta también.
- Además de 802.3, existen 802.4 (bus de token) y 802.5 (anillo de token). La idea es que las estaciones alternan en el uso del medio (intercambiando un *token*, que representa el turno). La ventaja es que el tiempo máximo de espera para mandar un marco tiene un límite. En el bus de token se usa un medio de broadcast, mientras que en el anillo de token se usan enlaces de punto-a-punto entre las estaciones.

Bridges

- Los bridges (puentes) son dispositivos que conectan las LANs. Razones para tener LANs múltiples:
    - Dueños autónomos (por ejemplos, departamentos distintos en una empresa)
    - Distancia entre grupos
    - Carga
    - Distancia entre computadores que debieran estar en la misma LAN
    - Confiabilidad: por contraste con un repetidor, un bridge puede rechazar basura de un nodo defectivo
    - Seguridad (restringir la propagación de marcos confidenciales)
- Se necesitan bridges distintos para conectar cada combinación de 802.x y 802.y. Los protocolos tienen formatos de marco distintos, velocidades distintas, y longitudes máximas de marco distintas.
- **Bridge transparente.** Este tipo de bridge no requiere ningún cambia a hardware o software. Hay que conectarlo y no más.
    - El bridge opera en un modo donde acepta todos los paquetes de la LAN (*modo promiscuo*). Con cada marco el bridge tiene que decidir si reenviarlo o descartarlo. Busca la dirección del destino en una tabla de hash dentro del bridge para determinar la línea de salida.
    - Al principio las tablas de todos los bridges son vacías. Porque no saben dónde están los destinos, los bridges reenvían marcos a todas las LANs.
    - Para llenar las tablas se usan un algoritmo de *aprender atrás.* Porque los bridges aceptan todos los marcos transmitidos en sus LANs, pueden notar las direcciones de fuente en los marcos. Con estas pueden guardar las LANs de las cuales los marcos originan. Entonces un bridge puede determinar a través de qué LAN se puede alcanzar una estación.
    - Porque la topología de las LANs puede cambiar, se expiran las entradas en las tablas después de algunos minutos. Si una máquina no transmite nada por algunos minutos, para mandar un paquete a él se deben inundar la red.
    - Para aumentar la confiabilidad del sistema a veces se usan más de un bridge para conectar dos LANs, que puede producir problemas con la inundación de marcos (ciclos son posibles).
    - La solución de este problema es que los bridges se comunican para construir un árbol. Los bridges hacen broadcasts de sus números de serie para elegir la raíz, y entonces pueden construir el árbol.

- **Bridge de ruteo de fuente.** La desventaja con los bridges transparentes es que malgastan ancho de banda (usan solamente un subconjunto de la topología, el árbol). Con el ruteo de fuente, cada fuente sabe el camino óptimo a cada destino posible.
    - Para encontrar las rutas las estaciones mandan un *marco descubridor* que es reenviado por cada bridge. Las respuestas incluyen el camino tomado por el marco.
    - Un problema es que este método puede producir una explosión en el número de marcos descubridores.

LANs de velocidad alta

- **FDDI (Fiber Distributed Data Interface).** Es una LAN de anillo de token que corre con una velocidad de 100 Mbps sobre distancias hasta 200 km con hasta 1000 estaciones conectadas. Se lo puede usar como un LAN normal pero el uso más común es para conectar LANs de cobre.
    - Consiste en dos anillos que transmiten en sentidos contrarios. Si hay una ruptura (por ejemplo, debido a un fuego) se pueden conectar los dos anillos en uno.
    - En vez de la codificación de Manchester usa un esquema que se llama **4 de 5:** se usan cinco bits para codificar cada cuatro. Dieciséis combinaciones son datos y otras son para control. Para sincronizar se usa un preámbulo largo y se requiere que los relojes son estables dentro de 0,005%.
    - Debido a la longitud potencial del anillo una estación puede generar un nuevo marco inmediatamente después de transmitir un marco, en vez de esperar su vuelta (como en 802.5). Pueden estar algunos marcos en el anillo a la vez.
    - FDDI también tiene un modo síncrono donde cada marco contiene cuatro canales de T1; puede haber hasta 16 marcos síncronos cada 125 microsegundos.
- **Fast Ethernet (Ethernet rápida).** Ya que FDDI no tenía éxito en el mercado de LANs (sino en el mercado de backbones), se desarrolló 802.3u, o Fast Ethernet. La idea es preservar los formatos de paquetes, las interfaces, etc., pero reducir el tiempo por bit de 100 nsegs a 10 nsegs (es decir, 100 Mbps).
    - Se usan solamente los cables de 10Base-T porque tienen muchas ventajas. El problema es que el par trenzado de clase 3 (el más común) no puede portar señales de 200 megabaud (100 Mbps con la codificación de Manchester) sobre 100 metros.
    - La solución para clase 3 es usar cuatro pares trenzados que tienen una señalización de 25 MHz, solamente 25% más rápida que con 802.3. De los cuatro, uno es siempre al hub, uno es siembre desde el hub, y los dos otros son conmutables. Se eliminan la codificación de Manchester (no es necesario en este caso con los relojes de hoy y distancias menos de 100 metros) y se usa la señalización ternaria. Entonces con tres pares se pueden transmitir 27 símbolos, o 4 bits a la vez.
    - Con la clase 5 el sistema es más sencillo porque el cable puede manejar 100 Mbps. Es full dúplex y compatible con FDDI. Hay también un estándar para fibra óptica que permite distancias hasta 2 km entre la estación y el hub.
    - Los hubs normalmente apoyan ambos 10 Mbps y 100 Mbps, que permite las instalaciones mezcladas.
- **HPPI (High-Performance Parallel Interface).** Fue desarrollado por Los Alamos para conectar los supercomputadores. Principios de diseño: chips estándares, ninguna opción, y rendimiento.
    - Tiene velocidades de 800 Mbps y 1600 Mbps. El primer es suficiente para 30 marcos por segundo de 1024×1024 pixels de 24 bits cada uno. Usa un conmutador de crossbar.
    - El cable contiene 50 pares trenzados (32 de datos, otros de control), es simplex, y tiene una longitud máxima de 25 metros. Se transfiere una palabra cada 40 nsegs. Se usan dos cables para la velocidad más alta.
    - Los marcos tienen 256 palabras. Se limita la detección de errores a un bit de paridad por palabra y una palabra de paridad por marco; otros checksums eran demasiado lentos.

- **Fibre channel.** La idea fue reemplazar los pares trenzados de HPPI con una sola fibra. Por desgracia es mucho más complicado, y por lo tanto más caro y difícil de implementar. Apoya velocidades de 100, 200, 400, y 800 Mbps.

El nivel de red

- Rutea los paquetes de la fuente al destino final a través de ruteadores intermedios. Tiene que saber la topología de la subred, evitar la congestión, y manejar los casos cuando la fuente y el destino están en redes distintas.
- El nivel de red normalmente es la interfaz entre el portador y el cliente. Sus servicios son los servicios de la subred. Fines:
    - Los servicios debieran ser independientes de la tecnología de la subred.
    - Se debiera resguardar el nivel de transporte de las características de las subredes.
    - Las direcciones de red disponibles al nivel de transporte debieran usar un sistema uniforme.
- La gran decisión en el nivel de red es si el servicio debiera ser orientado a la conexión o sin conexión.
    - **Sin conexión (Internet).** La subred no es confiable; porta bits y no más. Los hosts tienen que manejar el control de errores. El nivel de red ni garantiza el orden de paquetes ni controla su flujo. Los paquetes tienen que llevar sus direcciones completas de destino.
    - **Orientado a la conexión (sistema telefónico).** Los pares en el nivel de red establecen conexiones con características tal como la calidad, el costo, y el ancho de banda. Se entregan los paquetes en orden y sin errores, la comunicación es dúplex, y el control de flujo es automático.
- El punto central en este debate es donde ubicar la complejidad. En el servicio orientado a la conexión está en el nivel de red, pero en el servicio sin conexión está en el nivel de transporte. Se representan los dos enfoques en los ejemplos de la Internet y ATM.

Estructura interna de la subred

- Hay dos posibilidades que son independientes del servicio que se ofrece.
- **Circuitos virtuales.** Dentro de la subred normalmente se llama una conexión un circuito virtual. En un circuito virtual uno evita la necesidad de elegir una ruta nueva para cada paquete. Cuando se inicializa la conexión se determina una ruta de la fuente al destino que es usada por todo el tráfico.
    - Cada ruteador tiene que guardar adónde debiera reenviar los paquetes para cada uno de los circuitos que lo pasan. Los paquetes tienen un campo de número de circuito virtual en sus encabezamientos, y los ruteadores usan este campo, la línea de entrada, y sus tablas de ruta para reenviar el paquete en la línea de salida propia.
    - Se cobra el tiempo que la conexión existe, que corresponde a la reservación de entradas de tabla, ancho de banda, etc.
- **Datagramas.** Son paquetes que se rutean independientemente.
    - Los ruteadores tienen solamente las tablas que indican qué línea de salida usar para cada ruteador de destino posible. (Se usan estos tablas en los circuitos virtuales también, durante la inicialización de un circuito.)
    - Cada datagrama tiene la dirección completa del destino (estas pueden ser largas).
    - El establecimiento de las conexiones en el nivel de red o de transporte no requiere ningún trabajo especial de los ruteadores.
- Balanzas:
    - Entre el espacio de memoria en los ruteadores y el ancho de banda. Con paquetes cortos las direcciones usan un gran porcentaje del ancho de banda; el espacio de memoria quizás esté más barato.
    - Entre el tiempo de inicialización y el tiempo a analizar las direcciones.
    - Evitación de congestión. Es más fácil con los circuitos virtuales.

    - Vulnerabilidad. La falla de un ruteador en un circuito virtual destruye el circuito. Los datagramas están más robustos.
- Cada combinación de servicio y estructura interna es posible y existe. Ejemplos: UDP sobre IP (sin conexión, datagrama), TCP sobre IP (conexión, datagrama), UDP sobre IP sobre ATM (sin conexión, circuito virtual), ATM AAL1 sobre ATM (conexión, circuito virtual).

Algoritmos de ruteo

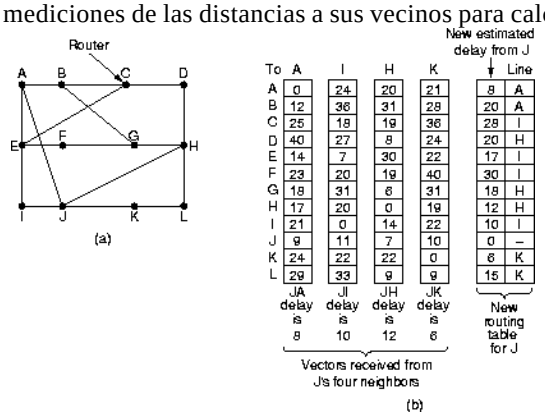
- El *algoritmo de ruteo* decide en qué línea de salida se debiera transmitir un paquete que llega. Propiedades deseables:
    - Correctitud y sencillez.
    - Robustez. Una red puede tener que operar por años y experimentará fallas de software y hardware. El algoritmo de ruteo no debe requerer que se reinicializa la red después de fallas parciales.
    - Estabilidad. Debiera tener un equilibrio.
    - Justicia y optimalidad. Están frecuentemente contradictorias. Se necesita una balanza entre la eficiencia global y la justicia al individual. ¿Qué podemos optimizar? El retraso por paquete o la utilización global de la red son posibilidades. Estos también están contradictorios, porque con 100% utilización los retrasos aumentan. Una solución intermedia es minimizar el número de saltos.
- Los algoritmos pueden ser adaptativos o no. Los primeros cambian sus decisiones de ruteo para reflejar la topología y el tráfico en la red. Los últimos son estáticos.
- **El principio de optimalidad.** Si el ruteador J está en el camino óptimo desde ruteador I a ruteador K, entonces la ruta óptima desde J a K está en la misma ruta. El conjunto de rutas óptimas forma el *árbol de hundir (sink tree)*. El fin de los algoritmos de ruteo es descubrir y usar los árboles de hundir de todos los ruteadores. Un problema es que la topología cambia.

Algoritmos estáticos

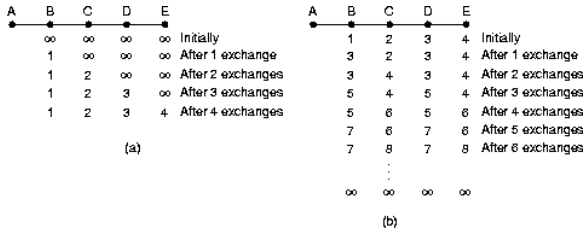
- **Camino más corto.** Se calculan los caminos más cortos usando alguna métrica. Posibilidades: el número de saltos, la distancia física, el retraso de transmisión por un paquete de prueba, el ancho de banda, el tráfico promedio, el costo de comunicación, etc.
- **Inundación.** Se manda cada paquete que llega sobre todas las otras líneas. Puede generar un número infinito de paquetes, así que se necesita un método para restringir la inundación.
    - Se puede usar un contador de saltos en cada paquete que se decrementa después de cada salto. Cuando el contador es cero se descarta el paquete.
    - Se pueden guardar números de secuencia agregados por cada ruteador a los paquetes. Los ruteadores mantienen listas de los números de secuencia más altos vistos y descartan los paquetes que son duplicados.
    - En la *inundación selectiva* se mandan los paquetes solamente sobre las líneas que salen más o menos en la dirección correcta.
- **Ruteo basado en el flujo.** Usa la topología y la carga para determinar las rutas óptimas. Si el tráfico entre nodos es conocido, se lo puede analizar usando la teoría de colas. Probando conjuntos distintos de rutas se puede minimizar el retraso promedio de la red.
- En general las redes modernas usan los algoritmos dinámicos en vez de los estáticos.

Ruteo de vector de distancia

- Se llaman estos algoritmos también *Bellman-Ford* y *Ford-Fulkerson*. Eran los algoritmos originales de ruteo de la ARPANET.
- Cada ruteador mantiene una tabla (un vector) que almacena las mejores distancias conocidas a cada destino y las líneas a usar para cada destino. Se actualizan las tablas intercambiando información con los vecinos.
- La tabla de un ruteador almacena una entrada para cada uno de los ruteadores en la subred (los ruteadores son los índices). Las entradas almacenan la línea preferida de salida y una estimación del tiempo o la distancia al destino. Se pueden usar métricas distintas (saltos, retrasos, etc.).
- Cada ruteador tiene que medir las distancias a sus vecinos. Por ejemplo, si la métrica es el retraso, el ruteador la puede medir usando paquetes de eco.
- Cada T msegs los ruteadores intercambian sus tablas con sus vecinos. Un ruteador usa las tablas de sus vecinos y sus mediciones de las distancias a sus vecinos para calcular una nueva tabla.



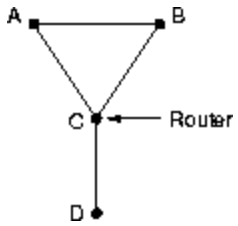
- El ruteo de vector de distancia sufre el problema que incorpora buenas noticias rápidamente pero malas noticias muy lentamente. Por ejemplo, en la parte (a) del dibujo siguiente el ruteador A acaba de subir. En esta red lineal la distancia nueva a A (en saltos) se propaga un salto por intercambio. Por contraste, en la parte (b) A acaba de bajar. Aunque B ahora no tiene una ruta a A, cree que hay una ruta a través de C, que cree que hay una ruta a través de B. Los ruteadores no pueden detectar el ciclo, y el número de saltos a A crece por solamente uno en cada turno. Este problema se llama *contar a infinito*, y hay que establecer un



valor de infinito suficiente pequeño para detectarlo. Por ejemplo, el valor puede ser la longitud del camino más largo más uno.

- Hay muchas soluciones a este problema, pero ninguna lo soluciona completamente. Una usada frecuentemente es la del *horizonte partido*. En esta variación del algoritmo la única diferencia es que siempre se reporta una distancia infinita a ruteador X sobre la línea que se usa para rutear a X.
    - En nuestro ejemplo esta modificación permite que las malas noticias se propagan un salto por intercambio.

- Por desgracia no funciona siempre. En el dibujo siguiente, A y B cuentan a infinito cuando D baja.



Ruteo de estado de enlace

- En 1979 se reemplazó el uso del ruteo de vector de distancia en la ARPANET. Tenía dos problemas principales:
  - La métrica era la longitud de las colas y no consideraba los anchos de banda de las líneas (originalmente todos eran 56 kpbs).
  - El tiempo para converger era demasiado grande.
- El nuevo algoritmo que se usa es el *ruteo de estado de enlace.* Tiene cinco partes. Cada ruteador tiene que
  - Descubrir sus vecinos y sus direcciones.
  - Medir el retraso o costo a cada vecino.
  - Construir un paquete con la información que ha averiguado.
  - Mandar este paquete a todo los ruteadores.
  - Calcular la ruta mínima a cada ruteador.
- **Descubrir los vecinos.** Cuando se bootea un ruteador, manda paquetes especiales de saludos sobre cada línea punto-a-punto suya. Los vecinos contestan con sus direcciones únicas. Si más de dos ruteadores están conectados por la LAN, se modela la LAN como un nodo artificial.
- **Medir el costo.** El ruteador manda paquetes de eco que los recipientes tienen que contestar inmediatamente. Se divide el tiempo por el viaje de ida y vuelta para determinar el retraso.
  - Un punto interesante es si debiera incluir en el retraso la carga de la línea. Esto corresponde a iniciar el reloj del viaje cuando se pone el paquete en la cola o cuando el paquete alcanza la cabeza de la cola.
  - Si incluimos la carga, se usan las líneas menos cargadas, que mejora el rendimiento.
  - Empero, en este caso es posible tener oscilaciones grandes en el uso de las líneas.
- **Construir el paquete.** El paquete consiste en la identidad del mandador, un número de secuencia, la edad, y la lista de vecinos y retrasos. Se pueden construir los paquetes periódicamente o solamente después de eventos especiales.
- **Distribuir los paquetes de estado de enlace.** Esto es la parte más difícil del algoritmo, porque las rutas en los ruteadores no cambian juntas. La idea fundamental es usar la inundación.
  - Para restringir la inundación se usan los números de secuencia que se incrementan cada vez se reenvía un paquete. Los ruteadores mantienen pares del ruteador de fuente y el número de secuencia que han visto, y descartan los paquetes viejos. Los paquetes nuevos se reenvían sobre todas las líneas salvo la de llegada.
  - Para evitar que el número de secuencia se desborda, se usan 32 bits.
  - Para evitar que los paquetes pueden vivir por siempre, contienen un campo de edad que se decrementa.

    - Si un ruteador cae o un número de secuencia se convierte malo, se perderán paquetes. Por lo tanto se incluye un campo de edad en cada entrada en la lista. Se decrementa este campo cada segundo y se descarta la información que tiene una edad de cero.
- **Calcular las rutas.** Se usa el algoritmo de Dijkstra. Un problema es que, debido a errores en los ruteadores, puede haber problemas en las rutas.

Ruteo jerárquico

- Las tablas de ruta crecen con la red. Después de algún punto no es práctico mantener toda la información sobre la red en cada ruteador.
- En el ruteo jerárquico se divide la red en regiones. Los ruteadores solamente saben la estructura interna de sus regiones.
- Para una subred de N ruteadores el número óptimo de niveles es lnN.

Ruteo de broadcast

- Para el broadcast de información hay algunas posibilidades.
- La más sencilla es mandar un paquete distinto a cada destino, pero esta malgasta ancho de banda.
- Otra posibilidad es la inundación pero genera demasiado paquetes y consume demasiado ancho de banda.
- En el *ruteo de destinos múltiples,* cada paquete almacena la lista de destinos. El ruteador divide el paquete en nuevos para cada línea de salida. Cada paquete tiene una nueva lista de destinos. Se divide la lista original sobre las líneas de salida.
    - Se puede usar el árbol de hundir o cualquier árbol de cobertura para la red, pero esto requiere que los ruteadores saben el árbol (que no es el caso en el ruteo de vector de distancia).
- En el *algoritmo que reenvía usando el camino inverso (reverse path forwarding),* se aproxima el comportamiento del uso de un árbol de cobertura. Cuando un paquete llega, se lo reenvía solamente si llegó sobre la línea que se usa para mandar paquetes a su fuente. Es decir, si el paquete llegó sobre esta línea, es probable que tome la ruta mejor a esta ruteador. Si no, es probable que sea un duplicado.

Algoritmos de control de congestión

- Cuando hay demasiado paquetes en alguna parte de la subred, el rendimiento baja. Esta situación se llama la *congestión*.
- Razones para la congestión:
    - Contienda para las líneas.
    - Memoria insuficiente en los ruteadores. Más memoria puede ayudar hasta un punto, pero aun cuando los ruteadores tienen memoria infinita, la congestión empeora, ya que los paquetes expiran antes de llegar a la cabeza de la cola.
    - Procesadores lentos. Si las CPUs no son suficiente rápidas, las colas pueden aumentar aun cuando hay capacidad en las líneas.

  La congestión propaga hacía arriba porque los mandadores tienen que guardar mensajes que no pueden entregar a un ruteador sobrecargado.
- El control de la congestión no es el mismo que el control de flujo. En el último el problema es evitar que el mandador manda más datos que el recibidor puede procesar, mientras que en el control de congestión el problema es evitar sobrecargar la capacidad de la red.
- Se dividen las soluciones al problema de control de congestión en dos clases:

- **Loop abierto.** Estas intentan evitar el incidente de congestión. Usan algoritmos para decidir cuándo aceptar más tráfico, cuándo descartar paquetes, etc. Los algoritmos no utilizan el estado actual de la red.
- **Loop cerrado.** Estas monitorean el sistema para la congestión y su ubicación, pasan esta información a los lugares donde se la pueden utilizar, y ajustan la operación del sistema para corregir el problema.

Formación del tráfico

- Una razón principal de la congestión es que el tráfico viene frecuentemente en ráfagas. Un método de loop abierto para manejar la congestión es forzar que el tráfico es más predecible. Se lo llama la *formación del tráfico.*
- La idea en la formación del tráfico es controlar la velocidad promedia de la transmisión de datos y la incidencia de ráfagas.
- Un cliente puede solicitar servicio para tráfico con algún patrón. Entonces el portador tiene que vigilar que el tráfico del cliente tiene este patrón (la *vigilancia del tráfico*). Esto es más sencillo con los circuitos virtuales que con los datagramas.
- **Algoritmo de cubo agujereado.** Cada host es conectado a la red por una interfaz que contiene un cubo agujereado, es decir, una cola interna finita. Si un paquete llega cuando la cola está llena, se lo descarta. El host puede poner un paquete en la red en cada intervalo. En esta manera se convierte un flujo ondulado en uno uniforme.
    - Con paquetes de tamaños variables se cambia el algoritmo a poner un número fijo de bytes en la red en cada intervalo. Si no quedan suficientes bytes para un paquete, se pierde el residuo y el paquete tiene que esperar hasta el próximo intervalo.
    - Ejemplo: Un computador puede producir 25 MB/seg, que es también la velocidad de la red. Empero los ruteadores solamente pueden manejar esta velocidad por intervalos cortos, y su rendimiento mejor es con velocidades de 2 MB/seg. Los datos llegan en ráfagas de 1 MB (40 mseg), una cada segundo. Entonces se puede usar un cubo de una capacidad de 1 MB y una velocidad de salida de 2 MB/seg.
- **Algoritmo de cubo de token.** El algoritmo de cubo agujereado nunca permite ráfagas en la salida. Para muchas aplicaciones es deseable que se permitan más variaciones en la velocidad de salida. En el algoritmo de cubo de token, el cubo contiene tokens en vez de paquetes.. Se añade un token nuevo cada intervalo al cubo hasta algún máximo. Para transmitir un paquete se necesita sacar un token del cubo.
    - No se descartan paquetes cuando el cubo está lleno, sino tokens.
    - En vez del derecho de mandar un paquete, los tokens pueden significar el derecho de mandar algún número de bytes.
    - El algoritmo permite las ráfagas pero hasta alguna longitud limitada S. Si la capacidad del cubo es C bytes, los tokens llegan con una velocidad de p bytes/seg, y la velocidad de salida máxima es M, tenemos $C+pS = MS$, o $S = C/(M-p)$. Si C = 250 KB, M = 25 MB/seg, y p = 2 MB/seg, el tiempo de ráfaga máxima es 11 mseg y su tamaño es 272 KB.
- La vigilancia de estos esquemas requiere que la red simula los algoritmos.

Control de congestión en subredes de circuitos virtuales

- Un algoritmo de loop cerrado que se puede usar en las subredes de circuitos virtuales es el *control de entrada.* Cuando se ha señalizado la congestión, no se establecen más circuitos virtuales.
- Otro enfoque es rutear los circuitos virtuales nuevos alrededor de las áreas con problemas.

- También se pueden negociar las características de la conexión durante su establecimiento y reservar el ancho de banda necesario. Pero malgasta recursos.

Paquetes de bloqueo

- Cada ruteador puede monitorear las utilizaciones de sus líneas. Puede mantener una variable u para la utilización que se actualiza según

  $$u_{nuevo} = au_{viejo} + (1-a)f$$

  donde a es algún constante y f es la utilización del instante.
- Cuando u sube sobre algún límite, la línea de salida entra en un estado de aviso. Si la línea de salida de un paquete nuevo está en un estado de aviso, se manda un *paquete de bloqueo* a su host de fuente original. Este paquete de bloqueo lleva el destino encontrado en el paquete, que permite que la fuente pueda identificar el paquete y la ruta que generaron el paquete de bloqueo. Se establece un bit en el encabezamiento del paquete que previene la generación de nuevos paquetes de bloqueo y lo reenvía como normal.
- Cuando un host recibe el paquete de bloqueo, debe reducir el tráfico al destino especificado por X por ciento. Porque ya hay otros paquetes en la ruta que generarán más paquetes de bloqueo, el host los debiera ignorar por algún intervalo. Después del intervalo, si hay más paquetes de bloqueo, debe reducir el flujo de nuevo.
- Si ningún paquete de bloqueo llega después de algún tiempo, el host puede aumentar el flujo. Normalmente los decrementos son 0,50, 0,25, etc., de la velocidad original. Los incrementos son más pequeños para evitar una nueva incidente de congestión.
- Un problema con este algoritmo es que los hosts cambian sus comportamientos voluntariamente. Una variación entonces es el *algoritmo con colas justas.* Cada línea de salida en un ruteador tiene un conjunto de colas, una para cada línea de entrada. El ruteador transmite un paquete por cola en turno. Si un host manda demasiado paquetes, no afecta a otros.
  - Una modificación del algoritmo es tomar en cuenta los tamaños de los paquetes. En esto se elige el próximo paquete usando un round robin que opera byte-por-byte.
  - En el *algoritmo con colas justas con pesos,* se permiten prioridades distintas para las colas.
- Un problema con los paquetes de bloqueo es el tiempo por su propagación. En este tiempo el host puede mandar muchos paquetes. Una solución es los *paquetes de bloque de salto-por-salto.* En esta cada ruteador intermedio tiene que reducir el flujo inmediatamente, que alivia la situación del ruteador más abajo. Este esquema alivia la congestión antes de que pueda desarrollar, pero al costo de requerir más buffers más arriba en el flujo.

Pérdida de carga

- Cuando todavía hay demasiado paquetes, los ruteadores pueden elegir paquetes a descartar.
- En algunas aplicaciones es mejor descartar los paquetes nuevos (por ejemplo, en la transferencia de archivos). En otras (la multimedia), los nuevos tienen más valor. Se llama la primera política *vino* y la última *leche.*
- En general se necesita la ayuda de los mandadores. Puede haber dos clases de paquetes con costos distintos (por ejemplo, en ATM esto es el caso).
- Generalmente es mejor que un ruteador empieza con descartar paquetes temprano en vez de tarde.

Internets

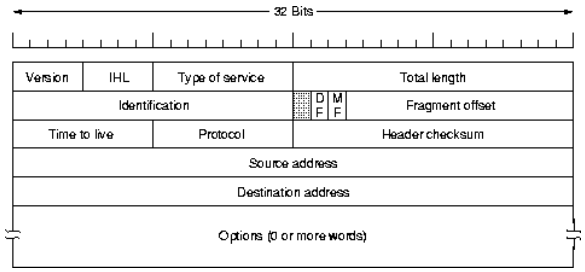
- La heterogeneidad en las clases de redes no parece que cambie pronto:
    - La base de redes distintas instaladas está creciendo.
    - El punto de decisiones de compras de redes se está bajando en las organizaciones.
    - Las redes distintas tienen tecnología distinta, así que no es una sorpresa que haya software distinto.
- Dispositivos para conectar las redes:
    - **Repetidores.** Amplifican o regeneran las señales para permitir cables más largos.
    - **Bridges.** Son dispositivos de guardar-y-reenviar. Operan en el nivel de enlace y pueden cambiar los campos de los marcos.
    - **Ruteadores de protocolos múltiples.** Son como los bridges pero funcionan en el nivel de red. Pueden conectar redes de protocolos distintos.
    - **Gateways (puertas) de transporte.** Conectan las redes a nivel de transporte.
    - **Gateways de aplicación.** Conectan dos partes de una aplicación (por ejemplo, correo electrónico) que usan formatos distintos.
- Porque los gateways frecuentemente conectan dos redes que pertenecen a organizaciones distintas, a veces se divide un gateway en dos partes (*gateways medios*). Las organizaciones no tienen que estar de acuerdo sobre la administración de una máquina ya que cada uno administra su máquina propia. Solamente tienen que estar de acuerdo sobre el protocolo entre los gateways medios.
- Algunas diferencias entre las redes:
    - **Clase de servicio.** Orientado a la conexión o sin conexión.
    - **Protocolos.** IP, IPX, CLNP, AppleTalk, DECnet, etc.
    - **Direcciones.** De un nivel (802) o jerárquicas (IP).
    - **Tamaño de paquete.** Cada red tiene su máximo propio.
    - **Control de errores.** Entrega confiable y en orden o sin orden.
    - **Control de flujo.** Ventana deslizante, control de velocidad, o ningún.
    - **Control de congestión.** Cubo agujereado, paquetes de bloqueo, etc.
    - **Seguridad.**
    - **Contabilidad.** Por tiempo conectado, por paquete, por byte, o ninguna.
- Dos enfoques a la creación de las internets:
    - **Circuitos virtuales concatenados.** Se establece una ruta a través de muchos gateways que convierten los paquetes. No se los pueden implementar si una de las redes es de datagrama.
    - **Internet sin conexión.** Los paquetes tienen que encontrar una ruta al destino. Las direcciones crean problemas. La principal ventaja de las internets basadas en los datagramas es que se las pueden usar sobre redes que no soportan los circuitos virtuales.
- **Túneles.** Si la fuente y el destino están en la misma clase de red, es sencillo conectarlos a través de algún tipo distinto de red. Se insertan los paquetes de la primera red en paquetes de la red de conexión y se extraen los paquetes de nuevo en la red de destino.
- **Fragmentación.** Un problema grande en las internets es el tamaño máximo de los paquetes. Si un paquete es demasiado grande para la próxima red que tiene que atravesar, el gateway tiene que partirlo en fragmentos.
    - **Fragmentación transparente.** El gateway parte el paquete. Se mandan todos los fragmentos al mismo gateway de salida, donde se los montan de nuevo. El gateway de salida tiene que saber cuando tiene todos los fragmentos. Todos los paquetes tienen que salir a través del mismo gateway. Hay que pagar el overhead de partir y montar en cada red de paquetes pequeños.
    - **Fragmentación no transparente.** El host de destino tiene que montar el paquete de nuevo. Hay más overhead porque los fragmentos persisten hasta el fin del viaje. Empero, se pueden usar gateways múltiples de salida.
    - Una manera para enumerar los fragmentos es que cada encabezamiento tiene el número del paquete original, el número del primer *fragmento elemental* en el paquete, y un bit que indica el

fragmento final. Los fragmentos consisten en conjuntos de fragmentos elementales que son suficientes pequeños para cualquiera red en la internet. Se los dividen cuando sea necesario.

- **Firewalls.** Con un firewall, todos los paquetes que entran o salen de un dominio son examinados. Consiste en dos filtros de paquete (son ruteadores con alguna funcionalidad extra) y un gateway de aplicación entre ellos.
    - Se usan dos filtros para asegurar que no hay ningún camino que sale afuera o entra hacia dentro sin pasar el gateway de aplicación.
    - Los filtros chequean las puertas de los paquetes de TCP para determinar si dejan pasar un paquete. Porque es difícil determinar los propósitos de los paquetes de UDP, a veces se los prohiben completamente.
    - El gateway de aplicación puede filtrar los paquetes en una manera más sofisticada.

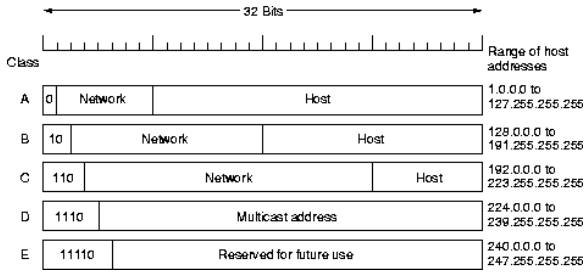
El nivel de red en la Internet

- El protocolo de IP (Internet Protocol) es la base fundamental de la Internet. Porta datagramas de la fuente al destino. El nivel de transporte parte el flujo de datos en datagramas. Durante su transmisión se puede partir un datagrama en fragmentos que se montan de nuevo en el destino.
- Paquetes de IP:



- **Versión.** Es 4. Permite las actualizaciones.
- **IHL.** La longitud del encabezamiento en palabras de 32 bits. El valor máximo es 15, o 60 bytes.
- **Tipo de servicio.** Combinaciones varios de la confiabilidad y la velocidad son posibles. No usado.
- **Longitud total.** Hasta un máximo de 65.535 bytes.
- **Identificación.** Para determinar a qué datagrama pertenece un fragmento.
- **DF (Don't Fragment).** El destino no puede montar el datagrama de nuevo.
- **MF (More Fragments).** No establecido en el fragmento último.
- **Desplazamiento del fragmento.** A qué parte del datagrama pertenece este fragmento. El tamaño del fragmento elemental es 8 bytes.
- **Tiempo para vivir.** Se lo decrementa cada salto.
- **Protocolo.** Protocolo de transporte a que se debiera pasar el datagrama.
- Las opciones incluyen el ruteo estricto (se especifica la ruta completa), el ruteo suelto (se especifican solamente algunos ruteadores en la ruta), y grabación de la ruta.

- Las direcciones de IP consisten en 32 bits. Hay 126 redes de 16 millón hosts cada una, 16.382 de 65.536, y 2 millón de 254. Los valores de 0 indican esta red y los de -1 la dirección de broadcast. Las direcciones que



empiezan con 127 son para pruebas de loopback.

- Normalmente se dividen una dirección de una clase en subredes. Esto permite que una organización tiene, por ejemplo, una dirección de clase B pero muchas subredes dentro de esto. Se divide la parte de la dirección que es normalmente el host en una dirección de red y una dirección de host.

- Hay una escasez de direcciones. El problema es que la mayoría de las organizaciones eligen direcciones de clase B, aunque la mitad de las redes de clase B tienen menos de 50 hosts. Ahora se usa CIDR (Classless InterDomain Routing) para asignar las redes de clase C en bloques de tamaños variables. Hay cuatro bloques grandes para Europa, Norteamérica, Sudamérica, y Asia/Pacífico. En cada uno se combinan en los ruteadores una dirección y una máscara para determinar a qué organización pertenece una dirección.

Protocolos de control

- La Internet tiene varios protocolos de control al nivel de red.
- **ICMP (Internet Control Message Protocol).** Ejemplos de paquetes: No se puede alcanzar el destino, la vida de un paquete expiró, valor ilegal en el encabezamiento, paquete de bloqueo (no usado más), paquete de eco o respuesta.
- **ARP (Address Resolution Protocol).** En una LAN es difícil mantener la correspondencia entre las direcciones de IP y las direcciones de LAN (por ejemplo, en una Ethernet hay direcciones de 48 bits). El protocolo ARP permite que una máquina haga un broadcast para preguntar qué dirección local pertenece a alguna dirección de IP. En esta manera no se necesita una tabla de configuración, que simplifica la administración.

- **RARP (Reverse ARP).** Permite que una máquina que acaba de bootear pueda encontrar su dirección de IP. Hay también el protocolo BOOTP, cuyos mensajes son de UDP y se pueden reenviar sobre ruteadores.

IPv6

- Fines:
    - Soportar miles de millones de hosts, incluso con la asignación ineficiente de direcciones.
    - Reducir el tamaño de las tablas de ruteo.
    - Simplificar el protocolo, que permite un procesamiento más rápida.
    - Proveer más seguridad.
    - Usar tipos distintos de servicio.
    - Mejorar el multicasting.
    - Permitir que un host puede viajar sin cambiar su dirección.
    - Permitir que el protocolo pueda cambiar en el futuro.
    - Permitir que los protocolos nuevos y antiguos puedan coexistir.
- Puntos principales del diseño aceptado:
    - Direcciones de 16 bytes, que implica 7×10^{23} por metro cuadrado de la tierra.
    - Un encabezamiento de 7 campos en vez de 13.
    - Mejor apoyo para las opciones.
    - Mejor seguridad con la autenticación y la privacidad.
    - Más tipos de servicio.
- IPv6 no usa la fragmentación. Los ruteadores tienen que manejar paquetes de 576 bytes. Si un paquete es mayor que una red puede manejar, se rechaza el paquete y el host tiene que fragmentarlo.
- Se eliminó el checksum.
- Se permiten datagramas de tamaños grandes (*jumbograms*), que es importante para las aplicaciones de supercomputador.

El nivel de transporte

- Utiliza los servicios del nivel de red para proveer un servicio eficiente y confiable a sus clientes, que normalmente son los procesos en el nivel de aplicación. El hardware y software dentro del nivel de transporte se llaman la *entidad de transporte.* Puede estar en el kernel, en un proceso de usuario, en una tarjeta, etc.
- Hay los mismos dos tipos de servicio que en el nivel de red: orientado a la conexión o sin conexión. Son muy semejantes a los del nivel de red. Las direcciones y el control de flujo son semejantes también.
- Por lo tanto, ¿por qué tenemos un nivel de transporte? ¿Por qué no solamente el nivel de red? La razón es que el nivel de red es una parte de la subred y los usuarios no tienen ningún control sobre ella. El nivel de transporte permite que los usuarios puedan mejorar el servicio del nivel de red (que puede perder paquetes, puede tener ruteadores que bajan a veces, etc.).
- El nivel de transporte permite que tengamos un servicio más confiable que el nivel de red. También, las primitivas del nivel de transporte pueden ser independiente de las primitivas del nivel de red. Las aplicaciones pueden usar estas primitivas y funcionar en cualquier tipo de red.

Primitivas del servicio de transporte

- El servicio de transporte orientado a la conexión es confiable, como los pipes en Unix.
- El nivel de transporte puede proveer también un servicio no confiable de datagrama, pero eso no es muy interesante.
- Porque muchos programadores usan las primitivas del servicio de transporte, deben ser convenientes y fáciles de usar.
- Ejemplo: Las entidades de transporte se comunican con TPDUs (Transport Protocol Data Units). Un conjunto de primitivas:

Primitiva	TPDU mandado
LISTEN	ninguno
CONNECT	CONNECTION REQ
SEND	DATA
RECEIVE	ninguno
DISCONNECT	DISCONNECTION REQ

El cliente hace un CONNECT. Cuando el CONNECTION REQUEST (solicitud de conexión) llega, la entidad de transporte del servidor confirma que el servidor está bloqueado en un LISTEN. Entonces manda un TPDU de CONNECTION ACCEPTED al cliente. Los partidos pueden comunicarse con SEND y RECEIVE, que son confiables. Para desconectar es posible que solamente un partido tiene que mandar un DISCONNECT (caso asimétrico) o ambos (caso simétrico).

- Primitivas de sockets de Berkeley:

Primitiva	Significado
SOCKET	Crear nuevo punto de comunicación
BIND	Ligar el socket con una dirección local
LISTEN	Alocar cola para llamadas
ACCEPT	Bloquear hasta que una solicitud llegue
CONNECT	Bloquear y establecer la conexión
SEND	Mandar datos

RECEIVE	Recibir datos
CLOSE	Cerrar la conexión

BIND permite la asignación de una dirección explicita (por ejemplo, una dirección bien conocida) con un socket. El servidor llama a las cuatro primeras primitivas en orden. El cliente llama a SOCKET y entonces CONNECT. La desconexión es simétrico.

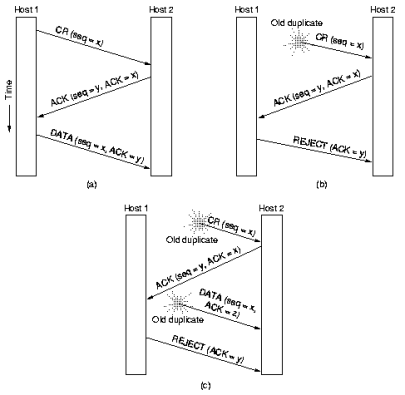
Protocolos de transporte

- Los protocolos de transporte se parecen los protocolos de enlace. Ambos manejan el control de errores, el control de flujo, la secuencia de paquetes, etc. Pero hay diferencias:
    - En el nivel de transporte, se necesita una manera para especificar la dirección del destino. En el nivel de enlace hay solamente el enlace.
    - En el nivel de enlace es fácil establecer la conexión; el host en el otro extremo del enlace está siempre allí. En el nivel de transporte este proceso es mucho más difícil.
    - En el nivel de transporte, se pueden almacenar paquetes dentro de la subred. Los paquetes pueden llegan cuando no son esperados.
    - El nivel de transporte requiere otro enfoque para manejar los buffers, ya que hay mucho más que conexiones que en el nivel de enlace.
- Cuando una aplicación quiere establecer una conexión con otra aplicación, necesita dar la dirección. Esta dirección es una dirección del nivel de transporte, y se llama un TSAP (Transport Service Access Point). Las direcciones del nivel de red se llaman NSAPs (Network Service Access Points). ¿Cómo sabe una aplicación la dirección del destino?
    - Algunos servicios han existido desde años y tienen direcciones bien conocidos. Para evitar que todos los servicios tienen que correr todo el tiempo (algunos se usan rara vez), puede ser un servidor de procesos que escucha a muchas direcciones a la vez y crea instancias de servicios cuando sea necesario.
    - Para otros servicios se necesitan un *servidor de nombres*. Esto tiene un TSAP bien conocido y mantiene una lista de nombres (strings) y direcciones. Los servidores tienen que registrarse con el servidor de nombres.
- Dado una dirección TSAP, todavía se necesita una dirección de NSAP. Con una estructura jerárquica para los TSAPs, la dirección NSAP es una parte de la dirección TSAP. Por ejemplo, en la Internet un TSAP es un par que consiste en la dirección de IP (NSAP) y la puerta.

Establecimiento de una conexión

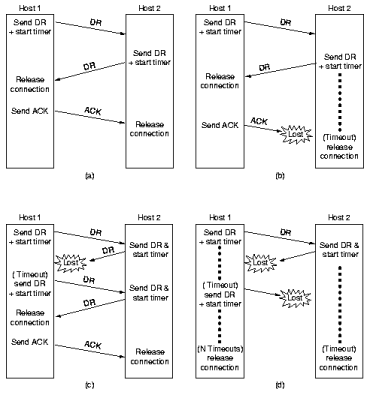
- El problema principal en el establecimiento de una conexión es que la subred puede perder, almacenar, y duplicar los paquetes. En el caso peor un cliente podría abrir una conexión para una transacción, y después la subred podría duplicar todos.
- Para eliminar los paquetes duplicados atrasados, se puede limitar el tiempo de vida máximo de un paquete en la subred (por ejemplo, usando un contador de saltos). Con un límite de los tiempos de vida de los paquetes y sus acuses de recibo, podemos usar cualquier protocolo de ventana deslizante para intercambiar paquetes entre dos hosts y detectar los paquetes duplicados.
- Un problema es que se necesita un método para elegir los números de secuencia que puede tolerar una caída (es decir, hay que asegurar que dos TPDUs con el mismo número de secuencia pueden existir a la vez). Cada host tiene un reloj; no se necesitan sincronizar los relojes de los hosts distintos. Un reloj es un contador binario que se incrementa en intervalos uniformes. El número de bits en el contador tiene que ser igual a o mayor del número de bits en los números de secuencia. Un reloj continúa corriendo aun cuando su host se baja.

- Un host usa los k bits más bajos del contador como el valor inicial de su número de secuencia. Los números de secuencia son suficiente grandes que no pueden repetir dentro de la vida de un paquete.
- Después de una caída, un host no sabe donde está en la secuencia. Puede esperar el tiempo de vida máximo de un paquete antes de transmitir otra vez. Porque este tiempo puede ser en general demasiado grande, otro método es controlar la taza de TPDUs mandados por tick para garantizar que no se pueden duplicar números de secuencia. La taza no puede ser demasiado grande (un máximo de uno) ni demasiado pequeña.
- Otro problema es establecer la conexión y intercambiar los valores iniciales de los números de secuencia. Un paquete duplicado atrasado puede causar una falla en este establecimiento. La solución es un *three-way handshake.* Host 1 manda un CONNECT REQUEST con su número de secuencia. Host 2 manda un acuse de recibo con el número de 1 y su número propio. Finalmente host 1 manda un acuse de recibo con la elección de host 2.



Desconexión

- La desconexión asimétrica puede perder datos. La desconexión simétrica permite que cada lado pueda liberar una dirección de la conexión a la vez.
- La desconexión simétrica sufre del problema de los dos ejércitos. Hay dos ejércitos pequeños y separados que quieren atacar un ejército más grande. Si tienen un medio de comunicación mala, ¿cómo pueden sincronizar un ataque?
- En general, no es posible tener un protocolo perfecto en este caso. En nuestro caso tenemos hosts que quieren desconectar en vez de atacar, pero el problema es el mismo.
- Se usa el protocolo siguiente: El primer lado manda un DISCONNECT REQUEST (DR) y inicia un reloj. El recipiente entonces manda un DR y inicia un reloj. Cuando lo recibe, el mandador original libera la conexión y manda un ACK. El recipiente lo recibe y libera la conexión también. Si hay un timeout en el recibidor, libera la conexión. Si hay un timeout en el mandador, manda el DR otra vez (hasta N intentos) y finalmente libera la conexión. Este protocolo puede fallar si ninguna transmisión del mandador llega.



Control de flujo

- Si la subred no es confiable, el mandador tiene que almacenar los TPDUs mandados, como en el nivel de enlace. El recibidor puede decidir cuánto espacio en buffers quiere dedicar a sus conexiones, porque sabe si no acepta un paquete el mandador lo reenviará.
- En el caso de tráfico de bajo ancho de banda y en ráfagas, es mejor que se dedican los buffers dinámicamente. Pero con flujos constantes de alto ancho de banda, es mejor si el recibidor reserva una ventana completa de buffers para permitir la velocidad máxima de datos.
- La asignación dinámica de buffers corresponde al uso de ventanas de tamaños variables. El mandador pide algún número de buffers, y el recibidor contesta con el número disponible. Durante la conexión el recibidor puede cambiar este número. Para evitar bloqueos indefinidos debido a acuses de recibo perdidos, los hosts deben intercambiar a veces paquetes de control con los estados de los buffers.
- El mandador también tiene que controlar el tamaño de la ventana teniendo en cuenta la capacidad de la subred. Si la subred puede manejar c TPDUs/seg y el tiempo para mandar un paquete y recibir su acuse es r, la ventana debiera tener un tamaño de cr. Con este tamaño la conexión normalmente está llena. El mandador puede medir y monitorear estos parámetros.

Multiplexación

- A veces el nivel de transporte tiene que multiplexar las conexiones. Por ejemplo, si la subred cobra el uso de un circuito virtual por tiempo, puede ser demasiado caro para una organización tener muchos circuitos abiertos a la vez. Por lo tanto se usa la *multiplexación hacia arriba*, con más de una conexión de transporte en una sola conexión de red.
- Otra razón para usar la multiplexación es si la subred impone un control de ventana deslizante. Si un usuario necesita más capacidad que es disponible con la ventana, se puede dividir la conexión de transporte entre unas conexiones de red. Esto el la *multiplexación hacia abajo*.

Recuperación de caídas

- Si una parte de la subred se cae durante una conexión, el nivel de transporte puede establecer una conexión nueva y recuperar de la situación. El problema es más difícil si uno de los dos hosts se cae.
- Por ejemplo, considera un host que está mandando un archivo a otro usando un protocolo de parar-y-esperar. El recibidor se cae y se rebootea rápidamente, pero ha perdido su posición. Le puede informar al mandador sobre el problema. El mandador puede estar en dos estados: S1, con un TPDU pendiente sin acuse, o S0, con ningún TPDU pendiente. Tiene que decidir si debiera retransmitir el TPDU más reciente.
- El problema es que el mandador no sabe si el recibidor ha escrito el contenido de la última TPDU. Por ejemplo, si el mandador retransmite solamente cuando está en S1, tenemos el siguiente problema: El recibidor recibió el paquete, generó un acuse, y se cayó sin escribir. El mandador está en S0 y no retransmitirá el paquete.
- Si usamos una estrategia donde el recibidor escribe antes de generar el acuse, tenemos un problema nuevo. En este caso el recibidor escribe pero se cae antes de generar el acuse. El mandador está en S1 y retransmitirá el paquete, que es un error. Protocolos más complejos no ayudan cuando tenemos eventos distintos come estos.
- En general se puede recuperar de las caídas solamente si hay suficiente estado preservado.

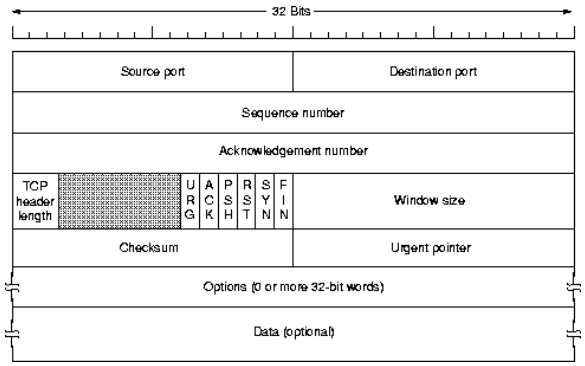
El protocolo de TCP

- El fin de TCP es proveer un flujo de bytes confiable de extremo a extremo sobre una internet no confiable. TCP puede adaptarse dinámicamente a las propiedades de la internet y manejar fallas de muchas clases.
- La entidad de transporte de TCP puede estar en un proceso de usuario o en el kernel. Parte un flujo de bytes en trozos y los mande como datagramas de IP.
- Para obtener servicio de TCP, el mandador y el recibidor tienen que crear los puntos terminales de la conexión (los *sockets*).
    - La dirección de un socket es la dirección de IP del host y un número de 16 bits que es local al host (la *puerta*).
    - Se identifica una conexión con las direcciones de socket de cada extremo; se puede usar un socket para conexiones múltiples a la vez.
    - Los números de puerta bajo 256 son *puertas bien conocidas* para servicios comunes (como FTP).
- Las conexiones de TCP son punto-a-punto y full dúplex. No preservan los límites de mensajes.
- Cuando una aplicación manda datos a TCP, TCP puede mandarlos inmediatamente o almacenarlos (para acumular más). Una aplicación puede solicitar que TCP manda los datos inmediatamente a través del flag de PUSH (empujar).
- TCP también apoya los *datos urgentes*. TCP manda datos con el flag URGENT inmediatamente. En el destino TCP interrumpe la aplicación (la manda una señal), que permite que la aplicación pueda encontrar los datos urgentes.

Implementación del protocolo

- Cada byte en una conexión de TCP tiene su propio número de secuencia de 32 bits.
- En TCP se mandan datos en *segmentos*. Un segmento tiene un encabezamiento de 20 bytes (más opciones) y datos de cero o más bytes. Hay dos límites en el tamaño de segmentos: el tamaño máximo de un paquete de IP (64K bytes) y la unidad de transferencia máxima (MTU, maximum transfer unit) de cada red.
- Si un segmento es demasiado grande para una red, los ruteadores lo puede dividir en segmentos nuevos (cada segmento nuevo añade un overhead de 40 bytes).
- TCP usa un protocolo de ventana deslizante. Cuando un mandador manda un segmento, inicia un reloj. El destino responde con un segmento que contiene un acuse de recibo con un número de acuse igual al próximo número de secuencia que espera. Si el reloj termina antes de que llegue el acuse, el mandador manda el segmento de nuevo. Problemas:
    - Se puede generar un acuse para un fragmento de un segmento, pero se puede perder el resto.
    - Los segmentos pueden llegar fuera de orden.
    - Con fragmentación y retransmisión, es posible que fragmentos de una transmisión y una retransmisión lleguen a la vez.
    - La red puede tener congestión.

El encabezamiento de TCP



- La puerta de la fuente y del destino identifican la conexión.
- El número de secuencia y el número de acuse de recibo son normales. El último especifica el próximo byte esperado.
- La longitud (4 bits) indica el número de palabras de 32 bits en el encabezamiento, ya que el campo de opciones tiene una longitud variable.
- Los flags:
    - **URG.** Indica que el segmento contiene datos urgentes. El *puntero urgente* punta al desplazamiento del número de secuencia corriente donde están los datos urgentes.
    - **ACK.** Indica que hay un número de acuse en el campo de acuse.
    - **PSH (Push).** El recibidor no debiera almacenar los datos antes de entregarlos.
    - **RST (Reset).** Hay un problema en la conexión.
    - **SYN.** Se usa para establecer las conexiones. Una solicitud de conexión tiene SYN = 1 y ACK = 0, mientras que la aceptación de una conexión tiene SYN = 1 y ACK = 1.
    - **FIN.** Indica que el mandador no tiene más datos a mandar. La desconexión es simétrica.
- TCP usa una ventana de tamaño variable. Este campo indica cuantos bytes se pueden mandar después del byte de acuse. El valor cero es legal; uno puede dar la permisión transmitir de nuevo con un segmento con el mismo número de acuse y una ventana más de cero.
- El checksum provee más confiabilidad.
- Las opciones permiten que los hosts puedan especificar el segmento máximo que están listos para aceptar (tienen que poder recibir segmentos de 556 bytes), usar una ventana mayor que 64K bytes, y usar repetir selectivamente en vez de repetir n.

Administración de conexiones

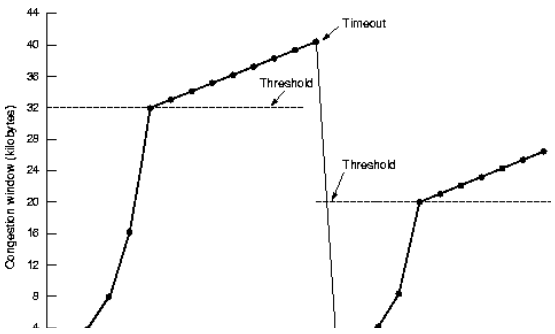
- TCP usa el three-way handshake para establecer una conexión. Si un segmento de solicitud de conexión llega y no hay un proceso esperando, TCP contesta con un paquete con el bit de RST establecido.
- Dos hosts pueden tratar de establecer una conexión entre los mismos sockets simultáneamente. Resulta en una sola conexión.
- Se usa un reloj para asignar los números iniciales de secuencia. Después de una caída un host tiene que esperar el tiempo de vida máximo de un paquete (120 segs).
- Para desconectar un sentido de la conexión, TCP manda un paquete de FIN y entonces espera un acuse. Para evitar el problema de los dos ejércitos el mandador desconecta después de dos tiempos máximos de vida de un paquete.

Política de transmisión

- Cuando la ventana tiene un tamaño de cero, el mandador no puede mandar segmentos, con dos excepciones. Se pueden mandar los datos urgentes y se pueden mandar un segmento de 1 byte para causar que el recibidor genere un acuse nuevo con la ventana. Este último es para evitar el bloqueo indefinido.
- Los mandadores no tienen que mandar datos inmediatamente y los recibidores no tienen que mandar acuses inmediatamente. Se puede usar esta flexibilidad para mejorar el rendimiento.
- Por ejemplo, considera una conexión de TELNET. Cada carácter tipiado genera un paquete de IP de 41 bytes, seguido por un acuse de 40 bytes, seguido por una actualización de la ventana de 40 bytes cuando la aplicación lee el carácter, seguido por 41 bytes cuando la aplicación hace un eco del carácter, y finalmente 40 y 40 más.
- Una optimización es demorar los acuses y las actualizaciones de ventana por 500 msegs para usar piggybacking.
- Una segunda optimización para minimizar los segmentos con solamente un byte de información es el *algoritmo de Nagle*. En esto se almacena los datos hasta que llegue el último acuse o haya suficiente datos para llenar la mitad de la ventana o un segmento máximo.
- Otro problema es el *síndrome tonto de ventana*. En esto la ventana del recibidor está llena. La aplicación en el recibidor lee un byte, que causa que el recibidor manda una actualización del tamaño de ventana al mandador. El mandador manda un byte y la ventana está llena otra vez. La solución de Clarke es no mandar una actualización de ventana hasta que el recibidor pueda manejar el segmento máximo de la conexión o el buffer esté medio vació, cualquier es menor. También el mandador no manda segmentos pequeños (si posible). Debiera esperar hasta que haya suficiente espacio en la ventana para un segmento completo o el buffer del recibidor esté medio vacío (según su estimación de su tamaño).

Control de congestión

- TCP trata de manejar la congestión usando la conservación del número de paquetes. La idea es no inyectar un paquete nuevo en la red hasta un paquete antiguo es entregado. TCP trata de hacer esto manipulando dinámicamente el tamaño de la ventana.
- El primer paso es detectar la congestión. Antes era difícil ya que el timeout de un paquete perdido podía haber causado por ruido en la línea o congestión en un ruteador. Ahora la mayoría de los timeouts son debidos a la congestión.
- Hay dos problemas distintos que tenemos que solucionar: la capacidad de la red y la capacidad del recipiente. Para manejar cada uno, el mandador mantiene dos ventanas. Una ventana se negocia con el recipiente, y su tamaño es basado en el tamaño del buffer del recipiente. La otra ventana es la *ventana de congestión*. El mandador solamente puede mandar el mínimo de las dos ventanas.
- Cuando se establece una conexión, el mandador inicializa la ventana de congestión al tamaño del segmento máximo de la conexión. Si puede mandar un segmento máximo sin un timeout, se dobla el tamaño de la ventana y se mandan dos segmentos máximos. Este aumento continúa hasta que hay un timeout. Se llama la *salida lenta*, pero no es lenta, sino exponencial.
- Para controlar la congestión hay un tercer parámetro, el *umbral*, que al principio tiene un valor de 64K. Cuando hay un timeout, se establece el valor del umbral a la mitad de la ventana corriente de congestión, y se establece la ventana de congestión a un segmento máximo. Otra vez se usa la salida lenta, pero después de llegar al umbral la ventana crece solamente por un segmento máximo a la vez (es decir, linealmente) hasta la ventana del recibidor.



Administración de relojes

- TCP usa un *reloj de retransmisión* para determinar si debiera retransmitir un segmento. ¿Por cuánto tiempo debiera durar el intervalo de timeout?
- El problema en establecer un valor para este timeout es que la varianza en tiempos de ida y vuelta es mucho mayor que en el nivel de enlace. El promedio y varianza pueden cambiar rápidamente mientras que la congestión crece y desparece. Si el timeout es demasiado largo el rendimiento sufre, pero si es demasiado corto hay retransmisiones innecesarias.
- Se usa un algoritmo debido a Jacobson. TCP mantiene un variable *RTT* para cada conexión con una estimación del tiempo de ida y vuelta. Si el próximo segmento toma M segundos, se actualiza *RTT* según RTT = alpha RTT + (1-alpha)M. Un valor típico para alpha es 7/8.
- Dado *RTT* hay que elegir todavía un timeout de retransmisión. TCP normalmente usa beta RTT, donde beta debiera ser proporcional (más o menos) a la desviación estándar. Se usa la *desviación promedia* D, donde D = alpha D + (1-alpha)|RTT - M|. alpha puede ser distinta a la previa.
- Finalmente, el timeout normalmente es *RTT*+4D.
- Un problema en la estimación de *RTT* es lo que pasa cuando se retransmite un segmento. Cuando el acuse llega, no es claro si refiere al primer segmento mandado o a la retransmisión. El algoritmo de Karn es no actualizar *RTT* con los acuses de segmentos retransmitidos. En vez de eso, se dobla el timeout con cada timeout hasta que los segmentos llegan sin timeout.
- TCP mantiene también un *reloj de persistencia*. El mandador lo usa para prevenir el bloqueo indefinido en el caso donde el mandador cree que la ventana del recibidor es cero y la actualización del recibidor fue perdido.
- Algunas implementaciones usan un *reloj de keepalive* (mantener vida) para determinar si el otro lado todavía está allí, y si no, desconectar. Una desventaja es que puede matar una conexión debido a una falla temporal de la red.

Rendimiento

- En general entender el rendimiento de redes es más arte que ciencia. La teoría no ayuda mucho.
- Fuentes de problemas de rendimiento:
  - **Congestión.**
  - **Desequilibrios entre recursos.** Por ejemplo, una línea gigabit conectada a un PC.
  - **Sobrecarga síncrona.** Un TPDU malo mandado en un broadcast puede producir miles de mensajes de error (una *tormenta de broadcast*. UDP sufría de este problema. Otro ejemplo: Después de una falla de electricidad todas las máquinas que están rebooteando pueden sobrecargar el servidor de RARP.
  - **Ajustamiento malo.** Si suficiente memoria no es dedicada a buffers, o el algoritmo de planificación no da una prioridad suficiente alta al procesamiento de TPDUs, o los timeouts son demasiado cortos o largos, se pueden perder paquetes.
  - **Redes gigabit.** En estos el *producto de ancho de banda por retraso*, que es la capacidad del canal entre mandador y recibidor, es muy grande (por ejemplo, 5 megabytes en una conexión transcontinental). Se necesitan ventanas grandes en los recibidores.
  - **Jitter.** Las aplicaciones de audio y video demandan una varianza muy pequeña en la desviación estándar del tiempo de transmisión.
- Medición de rendimiento de redes:
  - Asegurar que la muestra es suficientemente grande para pruebas estadísticas.

- Usar muestras representativas, con horas y días distintos.
- Tener cuidado en el uso de un reloj de grano grueso. Con repeticiones de mediciones se puede usar tal reloj para lograr una precisión más alta.
- Evitar condiciones extrañas (por ejemplo, respaldos durante pruebas). Es mejor usar una red desocupada y generar la carga tú mismo.
- El uso de cachés de archivos o buffers en el nivel de transporte pueden cambiar las mediciones.
- Entender exactamente lo que estás midiendo. Por ejemplo, las diferencias en los drivers de red entre dos máquinas distintas pueden ser la fuente de rendimientos distintos.
- Tener cuidado en extrapolar los resultados; no son necesariamente lineales.

Diseño para rendimiento mejor

- **La velocidad de CPU es más importante que la de la red.** En casi todas las redes el overhead del sistema operativo y los protocolos es mayor que el tiempo en el alambre. Ejemplos: En un Ethernet el tiempo mínimo para la comunicación de un RPC (llamada a procedimiento remoto) es 102 microsegundos, pero la realidad es más cerca a 1500 microsegundos. En una red gigabit, el problema es la transferencia de datos del buffer al proceso de cliente.
- **Reducir el número de paquetes para reducir el overhead de software.** Cada TPDU implica algún procesamiento, más el costo de una interrupción. Con paquetes mayores hay una reducción en el número de paquetes.
- **Minimizar los cambios de contexto.** Como las interrupciones pueden disminuir el rendimiento drásticamente. Podemos tener, por ejemplo, 4 cambios de contexto entre procesos de usuario, el kernel, y el administrador de red para entregar datos recibidos.
- **Minimizar la duplicación.** Durante el procesamiento de un paquete en los niveles distintos frecuentemente se duplica algunas veces. Por ejemplo, si una máquina de 50 MIPS hace tres copias de cada palabra y necesita 5 instrucciones por copia, vamos a necesitar 75 nanosegundos por byte. Entonces la máquina no puede manejar más de 107 Mbps, y probablemente no esto, porque hay otro overhead del protocolo, las instrucciones que refieren a la memoria son mucho más lentas, etc.
- **Puedes comprar más ancho de banda pero no retraso menor.** La velocidad de luz es un límite inevitable.
- **Evitar la congestión es mejor que recuperar de la congestión.** Cuando hay congestión en la red se pierden paquetes, se malgasta ancho de banda, se introducen retrasos, etc.
- **Evitar los timeouts.** Los timeouts implican operaciones duplicadas. Es mejor que los relojes son conservativos.

Procesamiento rápido de TPDUs

- La clave en el procesamiento rápido de TPDUs es identificar y acelerar el caso normal (transferencia de datos en un sentido).
- En el caso normal del mandador, los encabezamientos de TPDUs seguidos son casi los mismos. La entidad de transporte mantiene un encabezamiento de prototipo que copia a un buffer. Entonces sobreescribe los campos que cambian.
- En el recibidor el primer paso es identificar la conexión a que pertenece el TPDU (por ejemplo, usando una tabla de hash y las direcciones de socket). Una optimización aquí es mantener un puntero al último registro usado y probarlo primer; da una taza de ciertos de 90%.
- Si el TPDU es uno normal, se llama un procedimiento de camino rápido. Copia los datos a usuario y calcula el checksum a la vez (eliminando un pase adicional). En general el esquema de chequear si el encabezamiento es lo que es esperado y tener un procedimiento para manejarlo se llama la *predicción de*

encabezamiento. Con estas y otras optimizaciones es posible que TCP corra con una velocidad de 90% de una copia local de memoria a memoria.

- Otro área para optimización es la administración de relojes. La mayoría de los relojes nunca expiran. Una posibilidad es manejar los relojes como una lista; cada entrada contiene un contador que indica cuantos ticks después su predecesor expira. Entonces se decrementa la entrada en la cabeza a cero.
- Un esquema más eficiente (sin el costo alto de insertar y eliminar) es usar una *rueda de reloj*. Hay que saber el intervalo máximo de un timeout. Se usa un array circular donde cada tick corresponde a una entrada. Las entradas son punteros a los relojes por el tiempo indicado.

El nivel de aplicación

DNS--Domain Name System

- En vez de direcciones binarias de red, los programas normalmente usan strings de ASCII, tal como *uas.uasnet.mx.* Pero la red entiende solamente las direcciones binarias, así que se necesita una manera para traducir entre las dos.
- Originalmente en la ARPANET se usaba un archivo *hosts.txt* con todos los hosts y sus direcciones IP. Cada noche todos los hosts lo bajaban.
- Claramente este enfoque no puede escalar. Se necesita un sistema que evite conflictos pero no requiera la administración central. Ahora se usa DNS (sistema de nombres de dominios) para manejar los nombres. Usa una base de datos distribuida y una esquema jerárquico de administración de nombres.
- En principio, DNS funciona en la manera siguiente: Para traducir un nombre a una dirección un programa llama a un procedimiento de resolución. Esto manda un paquete de UDP al servidor local de DNS, que busca la dirección de IP usando el nombre. La vuelve al procedimiento de resolución, que la vuelva al programa.

Espacio de nombres de DNS

- Como en el correo, se parte la Internet en algunos cientos de dominios de primer nivel (*edu, com, cl, de, be,* etc.). Se parte cada dominio en subdominios, que se parten, etc. El sistema es un árbol.
- Hay dos tipos de dominios de primer nivel: genérico y países. Los dominios genéricos son *com* (comercial), *edu* (educación), *gov* (gobierno de EE.UU.), *mil* (fuerzas armadas de EE.UU.), *net* (proveedores de red), y *org* (organizaciones sin fines comerciales).
- Se dividen las partes de un nombre por puntos: *uas.uasnet.mx.* Los nombres pueden ser absolutos o relativos (por ejemplo, *ing*). Los últimos se tienen que interpretar en algún contexto.
- Los nombres son insensibles a caja. Los componentes pueden tener hasta 63 caracteres, y los nombres completos no pueden exceder 255 caracteres.
- Cada dominio controla la asignación de los dominios baja él. Un dominio puede crear nuevos subdominios sin la autorización de dominios arriba.
- Los nombres son basados en las organizaciones, no en las redes físicas.

Registros de recurso

- Cada dominio puede tener un conjunto asociado de *registros de recurso.* El procedimiento de resolución realmente recibe registros de recurso del servidor de DNS.
- Un registro de recurso tiene cinco partes: nombre de dominio, tiempo de vida, tipo, clase, y valor.
- El nombre de dominio es la clave del registro. Normalmente hay muchos registros por dominio y la base de datos guarda información sobre dominios múltiples. La orden de registros no importa.
- El tiempo de vida (en segundos) indica la estabilidad de la información en el registro. Se usa para controlar la expiración de copias de la información.

- Tipos y valores:
    - **SOA (Start of Authority).** Los parámetros para esta zona.
    - **A (Address).** La dirección de IP por el host.
    - **MX (Mail Exchange).** La prioridad y el nombre del dominio que puede aceptar correo electrónico dirigido a esto.
    - **NS (Name Server).** El nombre de un servidor de DNS para este dominio.
    - **CNAME (Canonical Name).** Un alias para un nombre (por ejemplo, *www.uasnet.mx*).
    - **PTR (Pointer).** Otro tipo de alias.
    - **HINFO (Host Info).** Una descripción de la máquina y su sistema operativo.
    - **TXT (Text).** Otra información opcional.
- La clase es siempre IN para información de Internet.
- Ejemplo: En *nslookup* prueba "*ls -d uas.uasnet.mx*".

Servidores de nombres

- Se divide el espacio de nombres de DNS en *zonas*. Cada zona tiene alguna parte del árbol y contiene servidores de nombre con la información autoritativa de la zona. Normalmente hay un servidor principal con su información en un archivo y otros secundarios que reciben su información del primario. Se pueden ubicar algunos servidores para una zona fuera de la zona para mejorar la confiabilidad.
- Cuando una pregunta llega en un servidor y el dominio está en la zona del servidor, vuelva el registro autoritativo. Si el dominio es remoto y no hay información disponible en el servidor, le pregunta al servidor del primer nivel del dominio. Esto le pregunta a uno de sus hijos, y la cadena sigue.
- Por fin cuando los resultados llegan se guardan en una caché.
- En vez de una solicitud recursiva, es posible recibir el nombre del servidor de probar próximo. En esta manera el cliente tiene más control sobre la búsqueda.