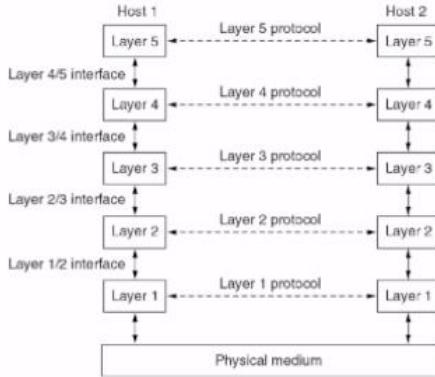


01 – REDES - INTRODUCCION

Jerarquía de los protocolos



Peer, layers, protocols, and interfaces.

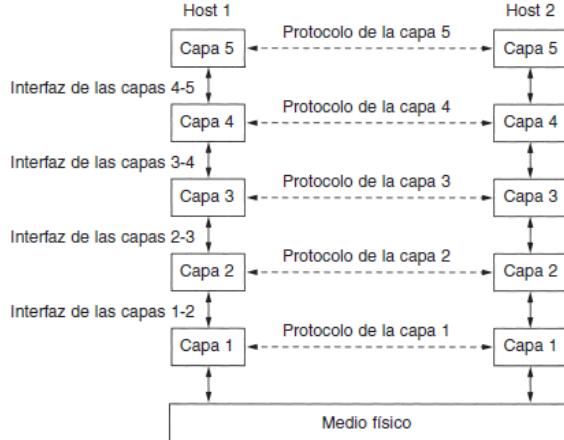


Figura 1-13. Capas, protocolos e interfaces.

La mayoría de las redes están organizadas como una pila de capas o niveles, cada una construida a partir de la que está debajo de ella. El propósito de cada capa es ofrecer ciertos servicios a las capas superiores, a las cuales no se les muestran los detalles reales de implementación de los servicios ofrecidos.

En el grafico la comunicación virtual se muestra con líneas punteadas y con líneas sólidas la comunicación física.

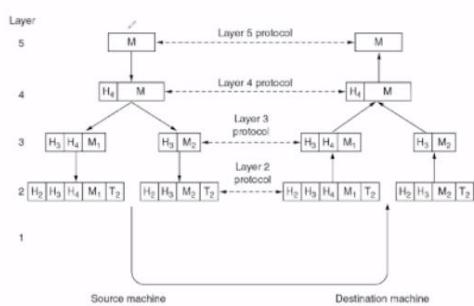
Si tengo el Host 1 que quiere comunicarse con el Host 2 hay que tener muchas consideraciones en cuenta, tipo enlace, tipo de cable, protocolo, si uno es más rápido que el otro, etc.

Si tengo que hacer un programa que controle todo esto sería un gran problema.

En software de redes se plantea separarlo en capas, y cada capa ataca un problema distinto. Cada capa le da un servicio resuelto a la capa superior.

El Host 1 no le puede enviar información al host 2 directamente, tiene que pasar por un proceso de encapsulamiento de cada capa hasta llegar a la capa física, ir por un medio físico y el host destino tiene que hacer el proceso inverso des encapsulando la información.

Ambas capas deben estar de acuerdo usando las mismas convenciones, el mismo protocolo.



Example information flow supporting virtual communication in layer 5.

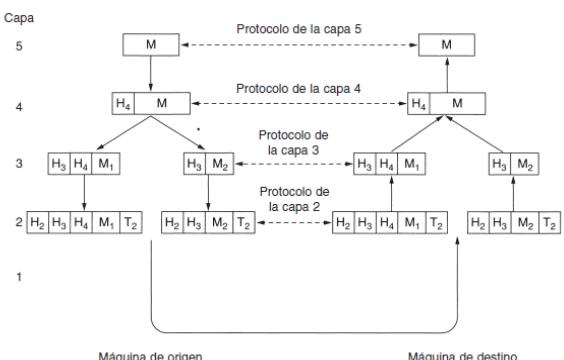


Figura 1-15. Ejemplo de flujo de información que soporta una comunicación virtual en la capa 5.

Este es un ejemplo un poco más real de cómo sería la transmisión de un mensaje.

En la capa 5 tiene un mensaje, se lo envía a la capa inferior la cual le pone un encabezado de su capa H4. Esto lo quiere enviar a una capa 3 pero podría pasar que la capa 3 no acepte un tamaño tan largo de mensaje entonces la capa 4 parte el mensaje en 2. Entonces la primer parte H4 con M1 (que sería una parte del mensaje) se lo envía a la capa 3 la cual le pone un encabezado de capa H3, luego le manda la 2da parte del mensaje y le suma un encabezado H3 también.

Cada uno de estos es enviado a la capa 2, que también le suma otro encabezado de capa 2 H2, algunas veces le suma también un tráiler T2.

Ya todos estos son ceros y unos que serán enviados por el cable en orden, cuando llega a destino se comprueba que este bien el encabezado y la cola y lo va enviando a la capa superior hasta llegar al mensaje.

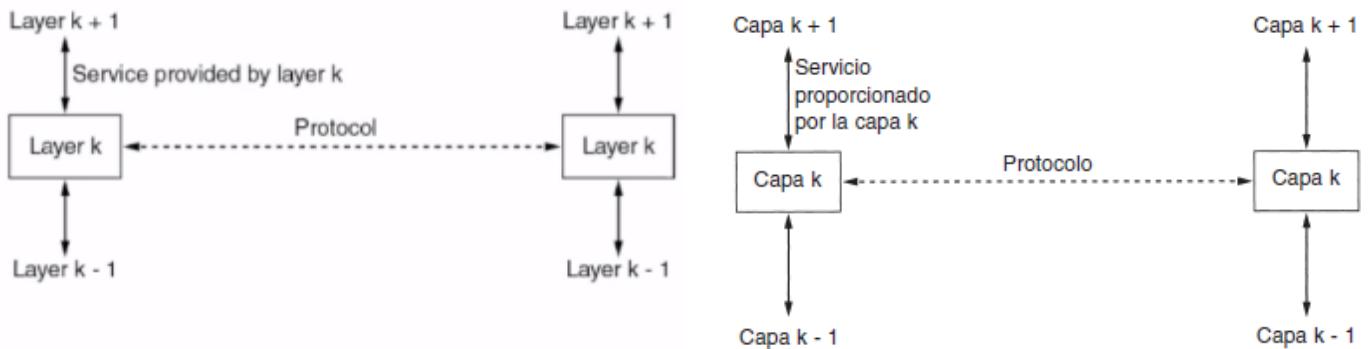
Aspectos de diseños de las capas (página 30)

- | | |
|--|--|
| <ul style="list-style-type: none">• Addressing or naming (in all layers)• Error Control (error detection, error correction)• Flow Control (different speeds sender/receiver)• Media Multiplexing (Statically or Dynamically)• Routing (multiple path)• Quality of service (real time)• Security (Confidentiality, Authentication, Integrity) | <ul style="list-style-type: none">• Direccionamiento• Control de errores• Control de flujo• Multiplexacion• Enrutamiento• Calidad de Servicio• Seguridad |
|--|--|

Problemas que deben solucionar cada capa.

- Como se cuál es la máquina de destino, deberían tener un esquema de nombres o **direcciónamiento**.
- Qué voy a hacer con los **errores**, los voy a corregir, los voy a detectar, los dejo pasar. Esto va a depender del protocolo y lo que quiera hacer.
- Qué pasa si una máquina es más rápida que la otra, hago **control de flujo** o no,
- Qué pasa si es canal es compartido, como lo **multiplexo**.
- Qué pasa si tengo que encaminar el mensaje por distintas redes y tengo más de un camino a la red de destino, **enrutamiento**.
- **Calidad de servicio**, los distintos mensajes deben ser manejados dependiendo del tipo, si tengo video en tiempo real se trata de una manera distinta que otro mensaje.
- Qué pasa si quiero agregar confidencialidad (solo el destinatario pueda ver el mensaje), autenticación es que el que manda el mensaje es quien dice ser, integridad es que nadie en el camino lo modifique.
Seguridad.

Relación de servicios a protocolos (página 36)



The relationship between a service and a protocol.

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

Un **servicio** es un conjunto de primitivas (operaciones) que una capa proporciona a la capa que está sobre ella. El servicio define qué operaciones puede realizar la capa en beneficio de sus usuarios, pero no dice nada de cómo se implementan tales operaciones. Un servicio está relacionado con la interfaz entre dos capas, donde la capa inferior es la que provee el servicio y la superior, quien lo recibe.

Un **protocolo**, en contraste, es un conjunto de reglas que rigen el formato y el significado de los paquetes, o mensajes, que se intercambiaron las entidades iguales en una capa. Las entidades utilizan protocolos para implementar sus definiciones del servicio. Son libres de cambiar sus protocolos cuando lo deseen, siempre y cuando no cambie el servicio visible a sus usuarios. De esta manera, el servicio y el protocolo no dependen uno del otro.

En otras palabras, los servicios se relacionan con las interacciones entre capas. En contraste, los protocolos se relacionan con los paquetes enviados entre entidades iguales de máquinas diferentes.

Relación entre servicio y protocolo.

Si quiero que dos máquinas interactúen entre sí, la capa N, tanto del origen como el destino deberían seguir el mismo protocolo o la misma norma.

Servicio es lo que le ofrece la capa a la de arriba, y le pide a la de abajo.

Modelos de referencia

Reference Models

- The OSI Reference Model
- The TCP/IP Reference Model
- A Comparison of OSI and TCP/IP
- A Critique of the OSI Model and Protocols
- A Critique of the TCP/IP Reference Model

Modelos de referencia usados en las redes en cuanto a capas, hay dos modelos que se usan, Modelo OSI y el otro modelo TCP/IP.

El modelo OSI es un modelo de referencia de ISO que decidió separar el problema en 7 capas, algunos protocolos se animaron a implementarlo, no tuvo mucho éxito comercial.

Por otro lado está el modelo TCP/IP.

Modelo de referencia OSI (página 37)

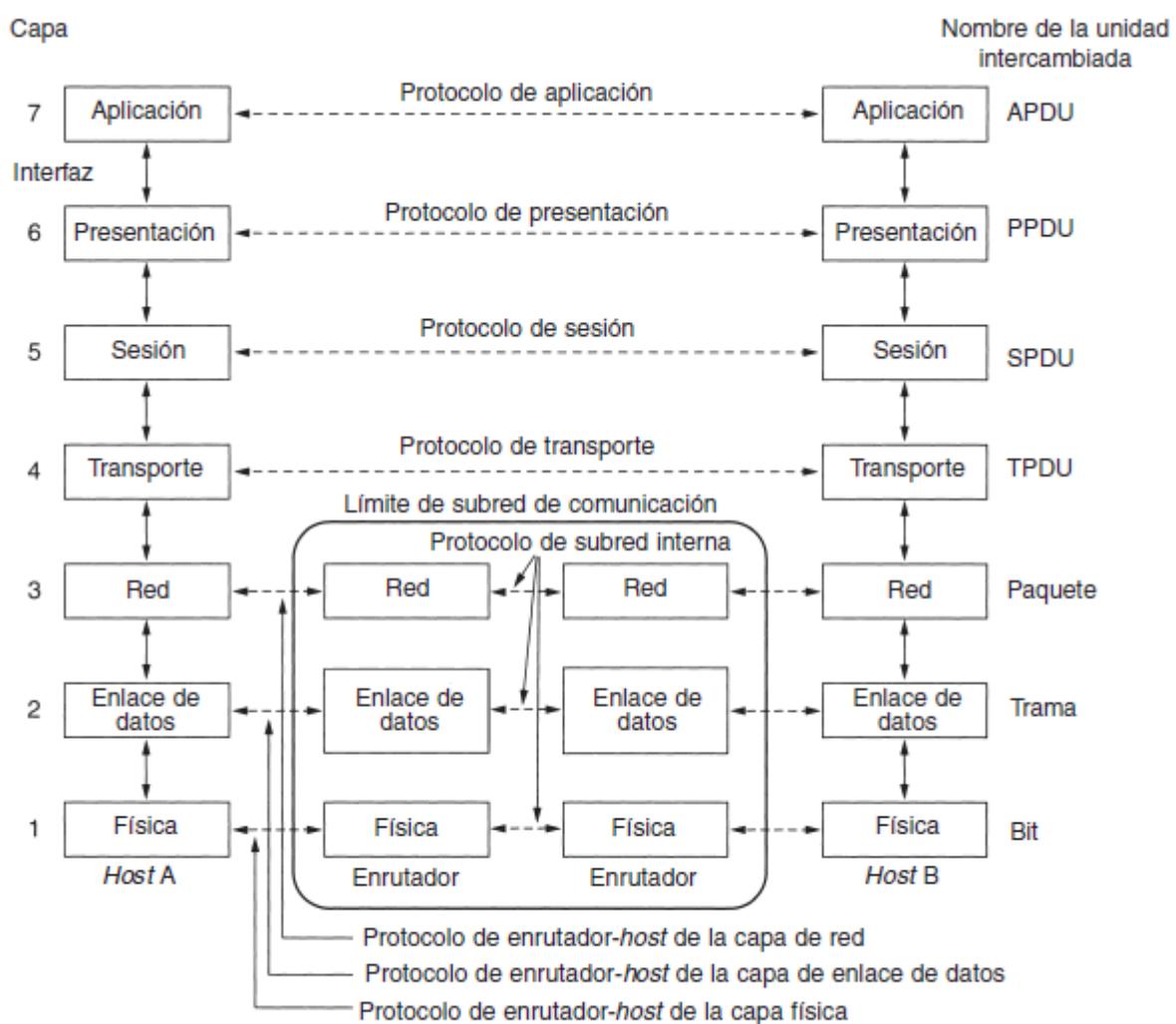
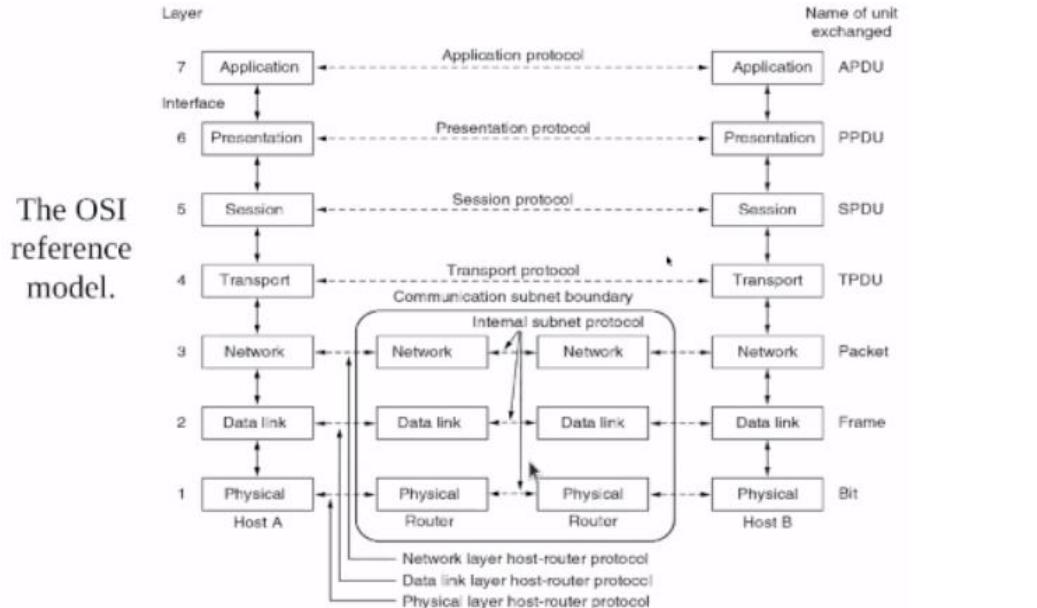


Figura 1-20. El modelo de referencia OSI.

Tiene 7 capas: Física, Enlace, Red, Transporte, sesión, presentación y aplicación.

Capa Física:

En esta capa se lleva a cabo la transmisión de bits puros a través de un canal de comunicación

Se encarga de definir todo lo referido a nivel de tensión, modulación, tipo de conectores, tipo de antena, frecuencia.

Capa de enlace de datos:

Las redes de difusión (broadcast) tienen un solo canal de comunicación, por lo que todas las máquinas de la red lo comparten. Si una máquina envía un mensaje corto —en ciertos contextos conocido como paquete—, todas las demás lo reciben. Un campo de dirección dentro del paquete especifica el destinatario. Cuando una máquina recibe un paquete, verifica el campo de dirección. Si el paquete va destinado a esa máquina, ésta lo procesa; si va destinado a alguna otra, lo ignora.

En contraste, las redes punto a punto constan de muchas conexiones entre pares individuales de máquinas. Para ir del origen al destino, un paquete en este tipo de red podría tener que visitar primero una o más máquinas intermedias. A menudo es posible que haya varias rutas o longitudes diferentes, de manera que encontrar las correctas es importante en redes de punto a punto. Por regla general (aunque hay muchas excepciones), las redes más pequeñas localizadas en una misma área geográfica tienden a utilizar la difusión, mientras que las más grandes suelen ser de punto a punto. La transmisión de punto a punto con un emisor y un receptor se conoce como unidifusión (unicasting).

La tarea principal de esta capa es transformar un medio de transmisión puro en una línea de comunicación que, al llegar a la capa de red, aparezca libre de errores de transmisión. Logra esta tarea haciendo que el emisor fragmente los datos de entrada en tramas de datos (típicamente, de algunos cientos o miles de bytes) y transmitiendo las tramas de manera secuencial. Si el servicio es confiable, el receptor confirma la recepción correcta de cada trama devolviendo una trama de confirmación de recepción.

Las redes de difusión tienen un aspecto adicional en la capa de enlace de datos: cómo controlar el acceso al canal compartido. Una subcapa especial de la capa de enlace de datos, la subcapa de control de acceso al medio, se encarga de este problema.

La capa de enlace toma eso directamente, no se preocupa del medio físico. Se encarga de solucionar las comunicaciones entre computadoras con la misma tecnología de red, sobre todo si son de una red de difusión. Como numerarlas o como direccionarlas. En el caso de que sean de difusión, como toman el canal, como manejan las colisiones en el caso de haberlas, si tiene manejo de error o no.

Capa de Red:

Cuando un paquete tiene que viajar de una red a otra para llegar a su destino, pueden surgir muchos problemas. El direccionamiento utilizado por la segunda red podría ser diferente del de la primera. La segunda podría no aceptar todo el paquete porque es demasiado largo. Los protocolos podrían ser diferentes, etcétera. La capa de red tiene que resolver todos estos problemas para que las redes heterogéneas se interconecten.*

En las redes de difusión, el problema de enrutamiento es simple, por lo que la capa de red a veces es delgada o, en ocasiones, ni siquiera existe.

La capa de red se despreocupa de lo anterior y se encarga de las maquinas que no están en la misma red, como hace para encaminar los datos. En esta capa también deberá generar alguna numeración o alguna identificación para cada uno de los host para poder saber cómo llegar a destino.

Capa de transporte:

La función básica de esta capa es aceptar los datos provenientes de las capas superiores, dividirlos en unidades más pequeñas si es necesario, pasar éstas a la capa de red y asegurarse de que todas las piezas lleguen correctamente al otro extremo.

La capa de transporte también determina qué tipo de servicio proporcionar a la capa de sesión y, finalmente, a los usuarios de la red.

La capa de transporte es una verdadera conexión de extremo a extremo, en toda la ruta desde el origen hasta el destino. En otras palabras, un programa en la máquina de origen lleva a cabo una conversación con un programa similar en la máquina de destino, usando los encabezados de mensaje y los mensajes de control. En las capas inferiores, los protocolos operan entre cada máquina y sus vecinos inmediatos, y no entre las máquinas de los extremos, la de origen y la de destino, las cuales podrían estar separadas por muchos enruteadores. En la figura 1-20 se muestra la diferencia entre las capas 1 a 3, que están encadenadas, y las capas 4 a 7, que operan de extremo a extremo.

La capa de transporte es la primera que va de extremo a extremo. Como esto es un mecanismo de IPC intenta ir desde el proceso origen al proceso destino, a cuál de los 100 procesos que tiene corriendo es al que le tengo que enviar la información.

Capa de sesión:

Esta capa permite que los usuarios de máquinas diferentes establezcan sesiones entre ellos. Control de dialogo, administración de token, sincronización.

La capa sesión en OSI, antes de intercambiar información asegurarme de ser quien digo ser, de alguna manera es iniciar una sesión antes de mandar datos.

Capa de presentación:

A diferencia de las capas inferiores, a las que les corresponde principalmente mover bits, a la capa de presentación le corresponde la sintaxis y la semántica de la información transmitida. A fin de que las computadoras con diferentes representaciones de datos se puedan comunicar.

La capa de presentación intenta resolver como se presenta la información, big-endian o Little-endian, si la codificación es ASCII o algún otro tipo de codificación.

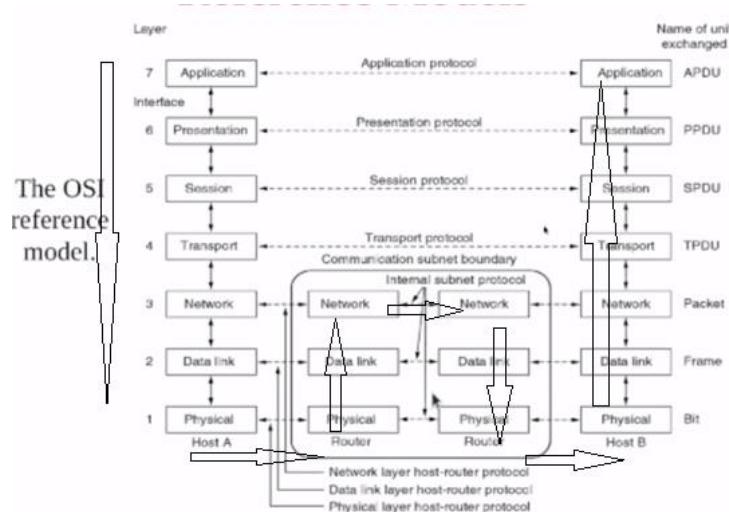
Capa de aplicación:

Esta capa contiene varios protocolos que los usuarios requieren con frecuencia. Un protocolo de aplicación de amplio uso es HTTP (Protocolo de Transferencia de Hipertexto), que es la base de World Wide Web.

La aplicación es el proceso que se está ejecutando de un lado y del otro. Quién es cliente y quien es servidor.

El cuadro del medio:

Suponiendo que el host A está en un tipo de red y el host B está en otro tipo de red, en el medio voy a necesitar tener un **encaminador o un router**. Este router deberá tener una interfaz que tenga la misma tecnología que el host A y otra interfaz que tenga la misma tecnología que el host B. Este router deberá usar el mismo protocolo de capa física para cada uno de los lados, al igual que capa de enlace y capa de red.



Si el **proceso del host A** le quiere enviar un mensaje a un proceso del host B, la información viajaría encapsulando los datos, una vez que llega al medio físico se lo envío pero al router, este router va a ir haciendo la des-

encapsulación, cuando llegue a la capa de red, va a notar que la información no se la mandan a él, sino que es para B, entonces entiende que lo debe encaminar, toma el otro camino, lo encapsula, va por capa física, cuando llega al destino hace el camino inverso y le llega el mensaje a B.

Las líneas punteadas entre aplicación, presentación, sesión y transporte son como hablan lógicamente, la aplicación A cree que habla con la aplicación B porque está usando IPC pero en realidad físicamente los datos no van de un extremo al otro si no que siguen todo ese camino marcado.

Modelo de referencia TCP/IP (página 41)



The TCP/IP reference model.

Figura 1-21. El modelo de referencia TCP/IP.

Capa de interred:

La capa de interred define un paquete de formato y protocolo oficial llamado IP (Protocolo de Internet). El trabajo de la capa de interred es entregar paquetes IP al destinatario. Aquí, el enrutamiento de paquetes es claramente el aspecto principal, con el propósito de evitar la congestión. Por estas razones es razonable decir que la capa de interred del modelo TCP/IP es similar en funcionalidad a la capa de red del modelo OSI. La figura 1-21 muestra esta correspondencia.

Capa de transporte:

Está diseñada para permitir que las entidades iguales en los hosts de origen y destino puedan llevar a cabo una conversación, tal como lo hace la capa de transporte OSI. Aquí se han definido dos protocolos de transporte de extremo a extremo. El primero, TCP (Protocolo de Control de Transmisión), es un protocolo confiable, orientado a la conexión, que permite que un flujo de bytes que se origina en una máquina se entregue sin errores en cualquier otra máquina en la interred. Divide el flujo de bytes entrantes en mensajes discretos y pasa cada uno de ellos a la capa de interred. En el destino, el proceso TCP receptor reensambla en el flujo de salida los mensajes recibidos. TCP también maneja el control de flujo para asegurarse de que un emisor rápido no saturé a un receptor lento con más mensajes de los que puede manejar.

El segundo protocolo de esta capa, UDP (Protocolo de Datagrama de Usuario), es un protocolo no confiable y no orientado a la conexión para aplicaciones que no desean la secuenciación o el control de flujo de TCP y que desean proporcionar el suyo. También tiene un amplio uso en consultas únicas de solicitud-respuesta de tipo cliente-servidor en un solo envío, así como aplicaciones en las que la entrega puntual es más importante que la precisa, como en la transmisión de voz o vídeo. La relación de IP, TCP y UDP se muestra en la figura 1-22. Puesto que el modelo se desarrolló, se ha implementado IP en muchas otras redes.

Capa de aplicación:

El modelo TCP/IP no tiene capas de sesión ni de presentación. No se han necesitado, por lo que no se incluyen. Contiene todos los protocolos de nivel más alto. TELNET, FTP, SMTP.

Capa host a red:

Debajo de la capa de interred hay un gran vacío. El modelo de referencia TCP/IP en realidad no dice mucho acerca de lo que pasa aquí, excepto que puntuiza que el host se tiene que conectar a la red mediante el mismo protocolo para que le puedan enviar paquetes IP. Este protocolo no está definido y varía de un host a otro y de una red a otra. Este tema rara vez se trata en libros y artículos sobre TCP/IP.

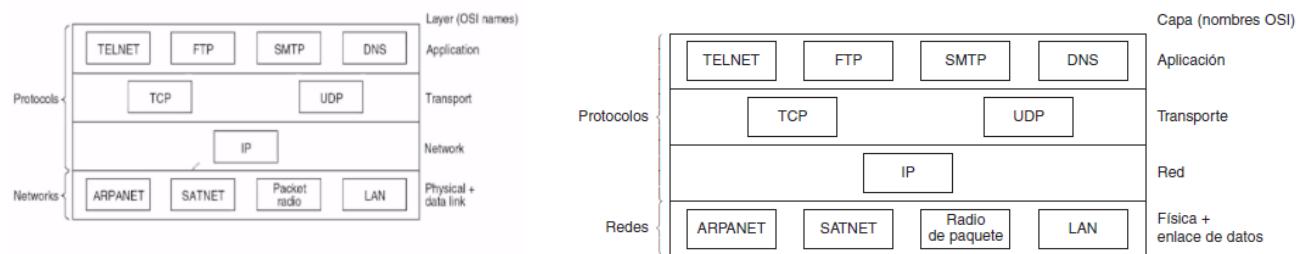
El modelo TCP/IP tiene básicamente 4 capas, la capa 7 se llama aplicación igual, la capa 4 se llama transporte, la 3 se llama internet en vez de red, luego la 2 y la 1 casi que no existen al igual que la 5 y la 6.

Cuando diseñaron TCP/IP no lo pensaron en capas, lo escribieron para que funcione, no les pareció necesario separarlo en tantas partes.

Lo que resolvería la presentación lo resuelve la aplicación, y el problema de sesión lo resuelve la capa de transporte, entonces puedo sacar esas dos capas.

Por otro lado, a partir de la capa 2 hacia abajo el protocolo no especifica nada, significa que podría utilizar cualquier capa física y capa de enlace, lo bueno es que no obliga a usar una tecnología física ni de enlace, no depender de comprar una marca o protocolo específico.

De todas maneras en todas las bibliografías siempre que se hable de capas se refiere a capas del OSI.



Protocols and networks in the TCP/IP model initially.

Figura 1-22. Protocolos y redes en el modelo TCP/IP inicialmente.

Ejemplos de protocolos que usan OSI, X.25 que no se usan más.

DNS en aplicación es conocido, es un protocolo del modelo TCP/IP, al mandar un correo usamos indirectamente el protocolo SMTP, al transferir cosas vamos a usar FTP, para conexión remota usamos TELNET.

En la capa de transporte el más usado se llama TCP (protocolo de control de transmisión), también está la opción de UDP (protocolo de datagramas de usuario).

En la capa de red el más conocido o el más usado es IP (INTERNET PROTOCOL), el que se usa es el 4 pero también podría ser IPv6.

Luego hacia abajo no hay protocolo porque el modelo de referencia TCP/IP no especifica nada, aca podría estar 802.11, Ethernet, cualquiera.

Comparación entre los modelos de referencia OSI y TCP/IP (página 44)

Tres conceptos son básicos para el modelo OSI:

Concepts central to the OSI model

- Services
- Interfaces
- Protocols
- Servicios
- Interfaces
- Protocolos

Una diferencia patente entre los dos modelos es el número de capas: el modelo OSI tiene siete y el TCP/IP sólo cuatro. Los dos tienen capas de (inter)red, transporte y aplicación, pero las otras capas son diferentes.

Otra diferencia está en el área de la comunicación orientada a la conexión comparada con la no orientada a la conexión. El modelo OSI soporta ambas comunicaciones en la capa de red, pero sólo la de comunicación orientada a

la conexión en la capa de transporte, donde es importante (porque el servicio de transporte es transparente para los usuarios). El modelo TCP/IP sólo tiene un modo en la capa de red (no orientado a la conexión) pero soporta ambos modos en la capa de transporte, lo que da a los usuarios la oportunidad de elegir. Esta elección es importante especialmente para protocolos sencillos de solicitud-respuesta.

Cuando diseñaron el modelo de referencia OSI se enfocaron en identificar bien los servicios, especificar bien las interfaces que había entre cada capa, y designar que cosas iban a cumplir cada protocolo. Una vez hecho esto se pusieron a escribirlo.

Criticas al modelo OSI y los protocolos. (página 46)

Why OSI did not take over the world

- Bad timing
- Bad technology
- Bad implementations
- Bad politics

1. Aparición inoportuna.
2. Mala tecnología.
3. Malas implementaciones.
4. Malas políticas.

La crítica del modelo OSI es que no se esparció a lo largo del mundo dado que:

- BAD TIMING Cuando decidieron escribirlo ya era tarde porque todo el mundo usaba TCP/IP
- BAD TECHNOLOGY aunque no eran tan malas.
- BAD IMPLEMENTATIOS dado que eran muchas capas, muy lento, hacían detección y corrección de errores, entonces cuando había que hacer una corrección se hacía en todas las capas y se repetía.
- BAD POLITICS

Principalmente por el TIMING, cuando salió el protocolo que cumplía con este modelo.

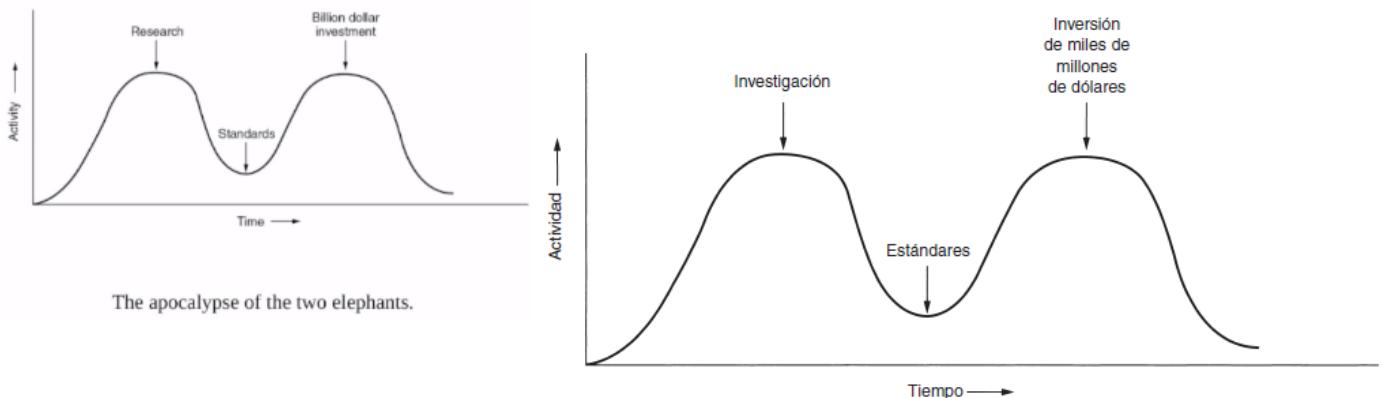


Figura 1-23. El apocalipsis de los dos elefantes.

Cuando hay mucha actividad de investigación no tiene sentido hacer un estándar, una vez que deja de haber innovación en esa tecnología y hay menos actividad es momento de hacer el estándar, luego los fabricantes comienzan a tener actividad e inversión en producción de productos que cumplan con el estándar.

En medio de la investigación saqué el estándar OSI, sin los protocolos, entonces después para implementar todos esos protocolos tomó mucho tiempo, porque recién ahí se dieron cuenta que eran un montón de capas en las que se había separado la problemática.

Mientras tanto MIT fue más pragmático y creó el modelo TCP/IP que no se basó en capas, solo lo hicieron para que funcionara y separó los problemas. Luego vieron que más o menos se adapta a las capas que propuso OSI.

Critica del modelo de referencia TCP/IP (página 48)

Problems:

- Service, interface, and protocol not distinguished
- Not a general model
- Host-to-network “layer” not really a layer
- No mention of physical and data link layers
- Minor protocols deeply entrenched, hard to replace

Criticas al protocolo TCP/IP

- No separa servicio, interface y protocolo, mezcla algunas cosas.
- No es un modelo como el modelo conceptual de OSI.
- Puede verse también como una ventaja que no hable de interfaces, porque no se ata a ninguna tecnología.
- Al no estar separado en capas, no es tan fácil sacar la capa 3 y reemplazarla por algo que funcione mejor.

Hybrid Model



Figura 1-24. Modelo de referencia híbrido que se usará en este libro.

The hybrid reference model to be used in this book.

Se va a utilizar un modelo híbrido

CAPA 1 - CAPA FISICA (página 85)

En este capítulo analizaremos la capa que está en la parte más baja de la jerarquía. Dicha capa define las interfaces mecánica, eléctrica y de temporización de la red. Comenzaremos con un análisis teórico de la transmisión de datos, el cual nos llevará a descubrir que la Madre Naturaleza establece límites en lo que se puede enviar a través de un canal.

Services offered from Physical layer

The major functions and services are:

- Bit-by-bit or symbol-by-symbol delivery
- Providing a standardized interface to physical transmission media

En la capa física lo que viaja son bits, o luz no luz, onda electromagnética, polarizado o no, y la función de que le da a la capa superior de enlace es la transmisión o entrega de estos bits. Provee una interface física.

Services offered from Physical layer (2)

The standardized interface define:

- Data encoding
- Transmission technique
- Physical Medium transmission
- Physical Medium connection

Que codificación, técnica de transmisión, medio. Esto es lo que debería definir la capa física

The Theoretical Basis for Data Communication

- Fourier Analysis
- Bandwidth-Limited Signals
- Maximum Data Rate of a Channel

Hasta qué velocidad puedo transmitir en un canal determinado, hay ecuaciones que se pueden aplicar de comunicaciones para saber cuál sería la tasa máxima de transmisión, también se ve en la capa física.

Guided Transmission Data

- Magnetic Media ?
- Twisted Pair
- Coaxial Cable
- Power lines
- Fiber Optics

Hay distintos medios de transmisión, par trenzado, coaxial, línea de fibra.

Wireless Transmission

- Radio Transmission
- Microwave Transmission
- Infrared and Millimeter Waves
- Lightwave Transmission

Radio, microondas, infra microondas, transmisión de luz.

Communication Satellites

- Geostationary Satellites
- Medium-Earth Orbit Satellites
- Low-Earth Orbit Satellites

También hay otras alternativas que son de mayor alcance, satélites geoestacionarios, órbita mediana o baja, son canales de comunicaciones posibles.

En este capítulo estudiaremos los principios de diseño de la capa 2, la capa de enlace de datos. Este estudio tiene que ver con los algoritmos para lograr una comunicación confiable y eficiente entre dos máquinas adyacentes en la capa de enlace de datos. Por adyacente, queremos decir que las dos máquinas están conectadas por un canal de comunicaciones que actúa de manera conceptual como un alambre (por ejemplo, un cable coaxial, una línea telefónica o un canal inalámbrico de punto a punto). La propiedad esencial de un canal que lo hace asemejarse a un alambre es que los bits se entregan con exactitud en el mismo orden en que fueron enviados.

Data Link Layer Intro

- Transmission between adjacent Machines
 - Same communication channel
 - Same order delivery
- Limitations
 - Errors
 - Different bit rates
 - Propagation delay

Esta capa se encarga de transmitir entre máquinas de la misma red, van a usar el mismo canal de comunicación, sale un byte, llega otro byte, es ordenado.

Si bien la capa física nos resuelve varios factores, la capa de enlace tiene varios problemas, ¿qué pasa si tiene errores?, ¿Qué pasa si el emisor es más rápido que el receptor?, ¿Tengo en cuenta el delay de propagación?.

Lo único que toma de la capa física es la transmisión bit a bit y todo el estándar de modulación, conectores, nivel de tensión.

Cuestiones de diseño de la capa de enlace de datos (página 184)

Data Link Layer Design Issues

- Services Provided to the Network Layer
- Framing
- Error Control
- Flow Control

¿Qué servicios le vamos a proponer a la capa de red?

Le vamos a dar entramado, manejo de error (que podría ser detección y corrección) y manejo de flujo.

Functions of the Data Link Layer

- Provide service interface to the network layer
- Dealing with transmission errors
- Regulating data flow
 - Slow receivers not swamped by fast senders

La capa de enlace de datos tiene que desempeñar varias funciones específicas, entre las que se incluyen:

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

Para cumplir con estas metas, la capa de enlace de datos toma de la capa de red los paquetes y los encapsula en tramas para transmitirlos. Cada trama contiene un encabezado, un campo de carga útil (payload) para almacenar

el paquete y un terminador o final, como se ilustra en la figura 3-1. El manejo de las tramas es la tarea primordial de la capa de enlace de datos. En las siguientes secciones examinaremos en detalle todos los aspectos mencionados.

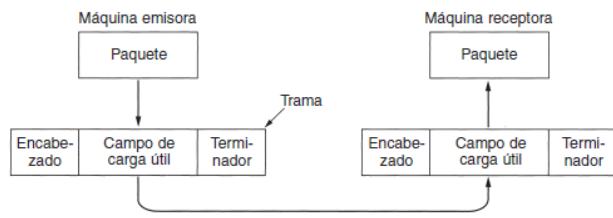


Figura 3-1. Relación entre los paquetes y las tramas.

Lo que le da a la capa de red, es gestionar los errores y manejar el control de flujo. Si tengo un receptor que no puede manejar muy rápido la información debería de alguna manera que el emisor no lo aturda.

Servicios proporcionados a la capa de red (página 184)

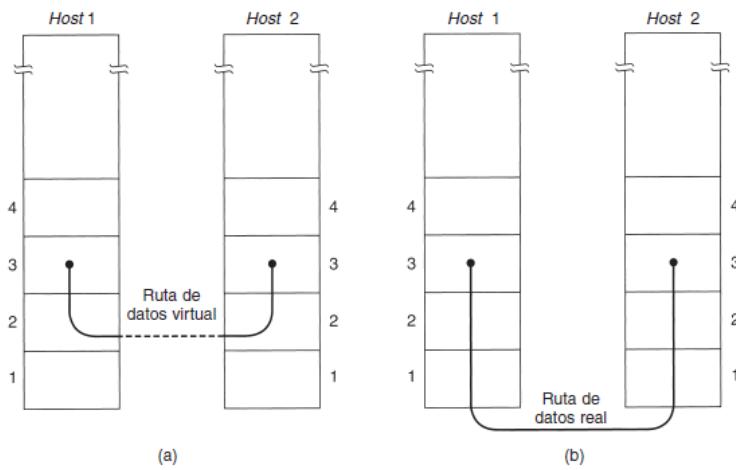


Figura 3-2. (a) Comunicación virtual. (b) Comunicación real.

La física era la 1, ahora estamos parados en la 2 que es la capa de enlace y tenemos que darle servicio a la capa siguiente que es la de red.

Virtualmente la capa 3 entiende que la capa de enlace 2 directamente se comunica con la capa de enlace del otro lado, es decir que la capa 3 no sabe cuántas capas hay abajo, solo sabe que le pide servicio a la anterior.

Functions of the Data Link Layer (2)

- Type of services
 - Unacknowledged connectionless
 - Very low error rates
 - Error -> No recovery
 - 802.3 – Ethernet
 - Acknowledged connectionless
 - Unreliable channels
 - Error -> Retransmission (inefficient. Why?)
 - 802.11 - Wifi
 - Acknowledged connection-oriented
 - Numbered frames
 - 3 Phases
 - Long unreliable channels – Satellite or Telephone

La capa de enlace de datos puede diseñarse para ofrecer varios servicios. Los servicios reales ofrecidos pueden variar de sistema a sistema. Tres posibilidades razonables que normalmente se proporcionan son:

- 1. Servicio no orientado a la conexión sin confirmación de recepción.**
- 2. Servicio no orientado a la conexión con confirmación de recepción.**
- 3. Servicio orientado a la conexión con confirmación de recepción.**

- Con respecto al **tipo de servicios**, la capa de enlace puede ser sin conexión y sin confirmación como se ve en el primer caso.
- Podría ser sin conexión pero con acuse de recibo (confirmación).
- Podría ser una comunicación orientada a conexión con acuse de recibo.

Esto hablando de dos entidades en la misma capa de enlace.

En el **1ro** se aplica a capas que tienen una baja tasa de errores, si ocurre un error no se hace nada. Entonces en este casi se tengo un dato para enviar lo envío y listo, el receptor no avisa que lleno ni nada, tampoco hago una inicialización antes de la conexión. Por ejemplo el Ethernet.

Con conexión: *si yo quiero conectarme con alguien llamó por teléfono fijo, espero que alguien me atienda, pregunto por esa persona y si esta recién ahí comenzaría el mensaje que quiero dar. Esto sería con conexión.*

Sin conexión: *enviar un mensaje de WhatsApp por ejemplo, lo envío pero no sé si él lo recibirá.*

El **2do** caso sin conexión pero con confirmación, envío el mensaje y después me contesta el ok, de alguna manera me aseguro que le llegue el mensaje. Son canales no tan confiables y si hay un error el emisor lo retransmite. Por ejemplo el WIFI.

El **3er** caso, para evitar la ineficiencia del caso anterior, numero cada mensaje que transmito entonces la confirmación la hace sobre ese número, en el caso de retransmitir lo hago con uno solo. Para satélites normalmente en capa de enlace se usa servicios orientados a conexión con confirmación.

Entramado (página 187)

- Break bit stream -> frameswhy?
 - Add redundant info
- Start of frames?
 - Byte count
 - Flag byte with byte stuffing
 - Flag bit with bit stuffing
 - Physical layer violations
 - 1. Conteo de caracteres.
 - 2. Banderas, con relleno de caracteres.
 - 3. Banderas de inicio y fin, con relleno de bits.
 - 4. Violaciones de codificación de la capa física.

El entramado toma el mensaje y lo separa en partes o en tramas. La idea de la capa de enlace es agregar información redundante, de control de flujo o de detección y corrección de errores, entonces si tengo un mensaje que es muy largo donde no sé cuándo comienza ni cuándo va a terminar. De aca nace la necesidad de separarlo en tramas y a estas tramas le voy agregando información.

¿Cuándo comienzo una trama?

- Conteo de bytes.
- Byte bandera.
- Bit bandera.

- Violación a la capa física.

El conteo de bytes es muy malo y no se usa, porque si tengo un error se corre y no se sincroniza más.

Se utiliza más el **byte bandera** o el bit bandera.

Framming the Data Link Layer (2)

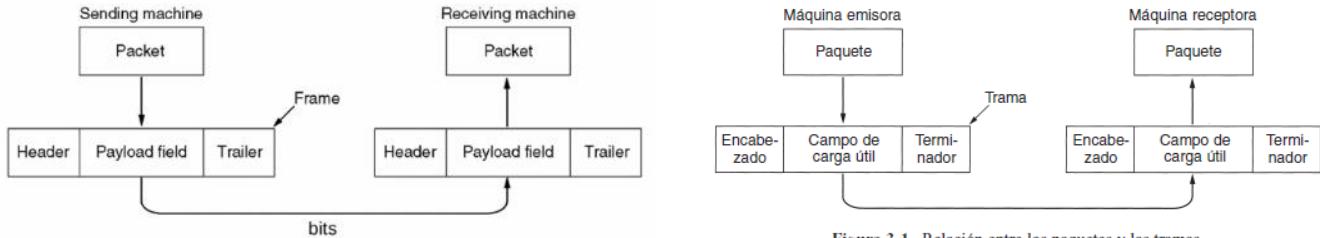


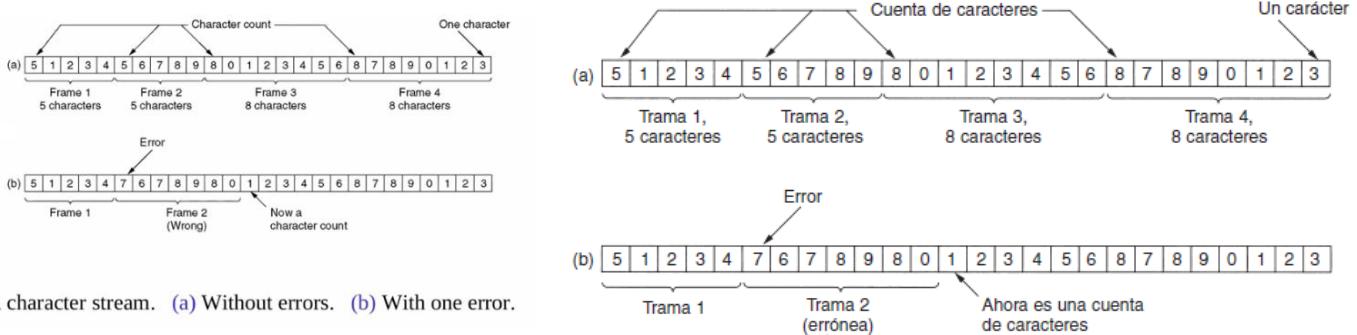
Figura 3-1. Relación entre los paquetes y las tramas.

Relationship between packets, frames and bits

Conteo de caracteres o conteo de Bytes (página 188)

En capa de red el mensaje se llama paquete, en cada capa el mensaje tiene un nombre, cuando hablo de capa de enlace se habla de tramas o frame.

El mensaje se pone en Payload o en la parte de carga de mi trama y se le pone un encabezado y un tráiler.



A character stream. (a) Without errors. (b) With one error.

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

El primer método de entramado se vale de un campo en el encabezado para especificar el número de caracteres en la trama. Cuando la capa de enlace de datos del destino ve la cuenta de caracteres, sabe cuántos caracteres siguen y, por lo tanto, dónde está el fin de la trama.

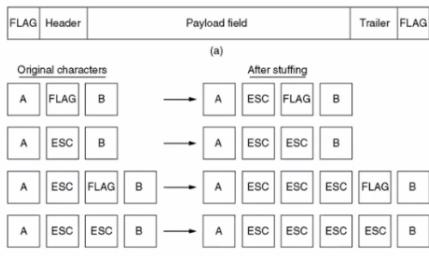
El conteo de bytes es muy sencillo, tengo un encabezado (Header), tiene un byte que me dice que la primera trama tiene 5 caracteres, luego viene el conteo de bytes de la 2da trama que tiene otros 5, la 3ra tiene 8 no siempre tienen el mismo tamaño y la última me dice que también tiene 8. Entonces la capa de datos sabe dónde se encuentran los datos que son luego de este byte.

El problema de este protocolo para el inicio de trama es que si tengo un error pierdo.

El caso b es el caso de una trama con error, un ruido que me cambia un bit por otro. En la 2da trama cambió un 5 por un 7, esto hace que se corra.

Banderas con rellenos de caracteres (página 189)

Framing (2) – Flag Byte



(a) A frame delimited by flag bytes.

(b) Four examples of byte sequences before and after stuffing.
The length may vary depending on the flags in the data area.
PPP uses this approximation

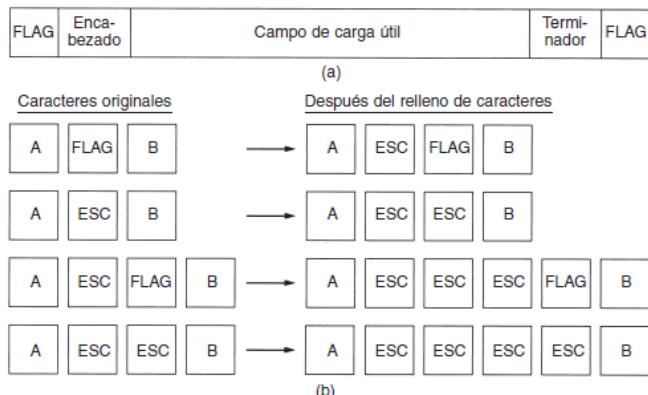


Figura 3-5. (a) Una trama delimitada por banderas. (b) Cuatro ejemplos de secuencias de bytes antes y después del relleno de caracteres.

El segundo método de entramado evita el problema de tener que sincronizar nuevamente después de un error, haciendo que cada trama inicie y termine con bytes especiales. La mayoría de los protocolos han utilizado el mismo byte, llamado bandera (o indicador), como delimitador de inicio y final, que en la figura 3-5(a) se muestra como FLAG.

2do método de entramado llamado byte de bandera.

Cada trama arranca con un byte que tiene un valor predeterminado siendo siempre el mismo y la trama termina con el byte de bandera, podría ser cualquier combinación.

¿Qué pasaría si en el Payload al enviar información hay un byte que tiene el mismo valor?

En la imagen se ve lo que podría estar en el Payload, el carácter A, luego el carácter que coincide con el Flag y el B.

Lo que hace el transmisor antes de transmitirlo es agregarle otro byte de escape para que cuando el transmisor lea, si ve un escape sabe que el byte que sigue no es una bandera, es un byte de dato y eso es lo que se envía.

Entonces esta capa de enlace agrega el byte de bandera, luego revisa todo el mensaje para ver si hay otro byte que coincida con el de bandera, de encontrarlo le agrega el byte de escape.

El receptor cuando ve un escape dentro de los datos lo descarta y sabe que lo que viene es un byte de datos.

Si hay dentro de los datos dos escapes, descarta uno y sabe que el otro es un dato.

Si hay justo un escape y una bandera dentro de los datos lo que hace es agregarle un escape al escape y un escape a la bandera entonces en el receptor se descartan esos dos escapes.

Como crítica para este protocolo es que agrega muchos bytes de control.

Protocolo PPP. Los modem de telefonía lo usaban, también lo usan los modem de ADSL, utilizan una variante que se llama PPPoE, POINT TO POINT PROTOCOL

Banderas con inicio y fin con relleno de bits.

Framing (3) – Flag Bits

(a) 011011111111111111110010
(b) 011011111011111011111010010
Stuffed bits
(c) 011011111111111111110010

Start/End bit pattern 01111110 (0x7E)

(a) The original data.

(b) The data as they appear on the line.

(c) The data as they are stored in receiver's memory after destuffing.
The length may vary depending on the flags in the data area.

HDLC and USB uses this approximation

(a) 011011111111111111110010
(b) 011011111011111011111010010
Bits de relleno
(c) 011011111111111111110010

Figura 3-6. Relleno de bits. (a) Los datos originales. (b) Los datos, según aparecen en la linea. (c) Los datos, como se guardan en la memoria del receptor tras eliminar el relleno.

Bits de bandera:

Cada trama inicia con un byte 7E “01111110”, hasta aca sería lo mismo que el anterior, pero en vez de hacer un relleno de bytes llena con bits.

“a” sería el Payload, el mensaje. Como mi bandera tiene 6 seguidos de 1, para evitar que un byte de datos sea una bandera los va contando y cuando encuentra 5 bits de unos seguidos le agrega un 0.

“b” es lo que transmito por el cable. Entonces solo he agregado bits, en este ejemplo 3 bits. En el receptor hago el camino inverso, cuando encuentro 5 bits en 1 saco el 0 siguiente.

Este es usado por el **protocolo HDLC y USB**.

Violaciones de codificación de la placa física. (página 190)

Framing (4) – Phy. Layer Violations

Some combination are invalid.

Example 4B/5B

Data (4B)	Codeword (5B)	Data (4B)	Codeword (5B)
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Se basa en combinaciones que no pueden existir (es un poco ineficiente).

Lo que hace es lo siguiente, una codificación de 4 bytes la codifica en 5 bytes, entonces una de ellas no va a ser válida y es la que utilizo como inicio o fin.

Con 4 bytes podrías tener 16 combinaciones, con 5 tendrás 32 combinaciones posibles, por ejemplo la combinación 01110 no existe dentro de los verdaderos entonces a esta la puedo usar como inicio y fin de trama.

Control de errores (página 191)

Error Control

- Delivery in order
 - Unacknowledged connectionless
 - No regard at all
 - Acknowledged connectionless
 - A feedback control frame
 - Timer, why?
 - Duplicated frames?
 - Acknowledged connection-oriented
 - Assign sequence numbers to frames

En cuanto a **control de flujo**, la entrega es ordenada porque cada vez que alguien envía una trama llega tal cual al en el mismo orden en el receptor salvo que tenga ruido y no la detecte como trama.

La que es **sin conexión y sin control de flujo** no puede hacer nada, lo ignora al control de flujo, no tiene, porque los datos son enviados y listo. Tampoco puede hacer algo contra los errores

Con **confirmación** si puedo hacer control de flujo, porque hasta que el receptor haga un acuse de recibo al emisor no enviaría otra cosa, si por algún motivo ese mensaje nunca llega no tendría confirmación y es por eso que se le pone un timer. También se podría perder la confirmación y repetiría la trama luego de ese tiempo.

La pregunta sería ¿Cuánto espero antes de reenviar?

En el 3ro de los casos que es con confirmación orientado a conexión de alguna manera negocio y las tramas se numeran. El emisor va a retransmitir la trama que no llegó. Esta es la mejor pero tiene un costo en cuanto a mensajes de control.

Error Detection and Correction

- Error-Correcting Codes (Forward Error Correction)
 - Add redundancy to each frame
 - Noisy channels
- Error-Detecting Codes
 - Add redundancy in retransmissions
 - Highly reliable channels

En relación al control de errores, podría tener detección y corrección de errores.

Con respecto a los **códigos de corrección**, se aplica cuando tengo canales muy ruidosos donde generalmente tengo errores. Adiciono información de control que me permita arreglar estos errores de manera que siempre estoy enviando información de más.

Error-Correcting Codes

- Types of error correcting codes
 - Hamming codes
 - Binary convolutional codes
 - Reed-Solomon codes
 - n degree polynomial -> n+1 points
 - The (255,233) code DSL, DVD, BluRay
 - Low-density parity check codes
 - Each output bit -> fraction of input bits
 - Codeword -> interactive approximation alg.
 - 10 Gbps Ethernet, 802.11 last version

- Código Hamming lo vimos en TD1
- Código binario convolucional.

Los otros están basados más que nada de un modo matemático

- Reed-Solomon que define un polinomio de grado “n” entonces lo que hace es contrastar contra eso. Podría tener un polinomio de grado 1 que representaría una recta entonces con dos puntos podría definir esa recta y sin importar los datos que me den que si no se encuentran en esa recta sé que están mal. Para grados mayores necesitaría “n+1” puntos y con eso los datos que me llegan los contrasto. DSL lo usa dado que va por cobre y es muy ruidoso, DVD y Blu-ray también lo usan.
- Código de chequeo de baja densidad que directamente es para matemáticos. Se arma como una matriz y de todos los bits que entran en esa matriz solo una fracción de ellos salen, esta matriz se llama Codeword y utilizan un algoritmo de aproximación iterativa. Las conexiones de 10Gbps Ethernet lo utilizan porque para estas tasas se utilizan frecuencias más altas y se puede tener ruido.

Error-Correcting Codes

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

Use of a Hamming code to correct burst errors.

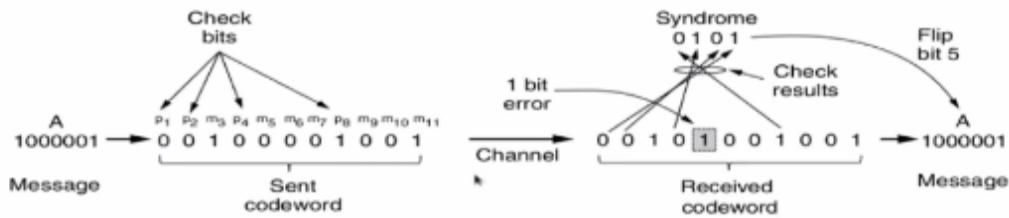
Check bits 1,2,4,8,16

Single bit correcting code (11,7)- Even parity

Hamming, agrega información adicional siempre, si tengo un byte término enviando bastante más que un byte.

Si tengo 7 bits de dato le agrego 4 quedando 11, estos bits que agrego tienen un peso, voy poniendo esta información de acuerdo a operaciones binarias.

Error-Correcting Codes (2)



$$p_1 = m_3 \text{ XOR } m_5 \text{ XOR } m_7 \text{ XOR } m_9 \text{ XOR } m_{11}$$

$$p_2 = m_3 \text{ XOR } m_6 \text{ XOR } m_7 \text{ XOR } m_{10} \text{ XOR } m_{11}$$

$$p_4 = m_5 \text{ XOR } m_6 \text{ XOR } m_7$$

$$p_8 = m_9 \text{ XOR } m_{10} \text{ XOR } m_{11}$$

Entonces se agregan por combinaciones binarias entre los bits de mensaje.

Por ejemplo, el bit de chequeo 4 se hace entre la OR exclusiva mensaje 5, 6 y 7. Como da 0 se pone 0.

Entonces a mi mensaje le agrego esos bits en la ubicación determinada, luego corro ese algoritmo y lo transmito.

En el receptor si hay algún bit que está mal, recupero esos 4 bits p1, p2, p4 y p8 y eso me da el valor del bit que esta cambiado.

Error-Correcting Codes (3)

Sequence of input bits \rightarrow Sequence of output bits

The encoder has memory (depends on the current and previous bits)

Number of previous bits \rightarrow constraint length (k)

Example NASA code , used in Voyager, GSM and 802.11. r=1/2;k=7

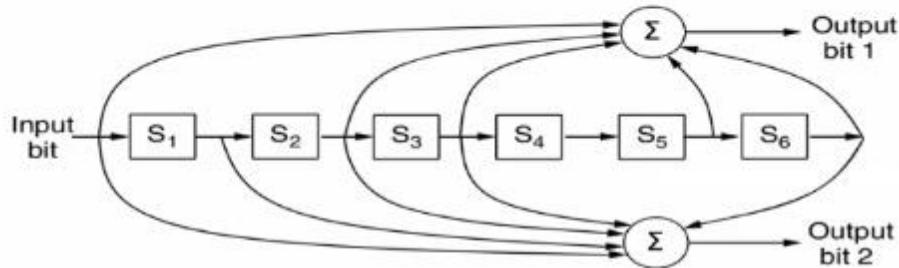


Figure 3-7. The NASA binary convolutional code used in 802.11.

Convolucional binario, a diferencia de otros que trabajan sobre bloques de datos, este trabaja sobre bits individuales.

La salida de un bit depende de la entrada pero también de los estados anteriores.

Se usaba en algunas versiones de la 802.11, no en la última. Se usaba en GSM protocolo de comunicaciones de telefonía. Voyager son dos sondas que están a 14 horas luz.

Error-Detecting Codes

- Error-Detecting codes, block codes
 - Parity
 - Checksum
 - Cyclic Redundancy Checks (CRC)

Código detector de errores, solo detecta, si está mal no hay acción salvo que al usar confirmación no confirme para que se repita el mensaje.

Error-Detecting Codes

- Parity
 - A parity bit en a byte – i.e. RS232
 - Detects a single bit error -> error burst
 - Interleaving → checksum – i.e. IP Proto.

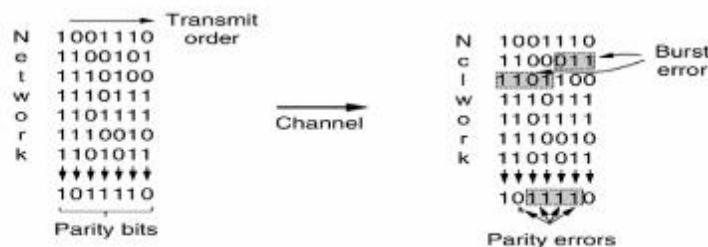


Figure 3-8. Interleaving of parity bits to detect a burst error.

Siempre será por bloques, en el de paridad, a un byte le voy a agregar un bit de paridad y será paridad par o impar dependiendo de la suma siempre me de igual, en el caso de que no dé hay un error, no sabré donde.

Normalmente RS-232 utiliza detección de errores por paridad. Si bien la comunicación serie es bastante ruidosa al ser de baja velocidad no se justifica hacer un algoritmo de corrección de errores. En caso de que detecte un error descarta esa trama o ese byte.

Una manera de mejorar esta detección de errores es haciendo un chequeo de paridad pero en un bloque más grande de bytes de manera que me sea más fácil detectar si ha habido errores. Tener en cuenta que esta detección no funcionaría si hay dos errores, dado que la paridad pasaría a ser la correcta.

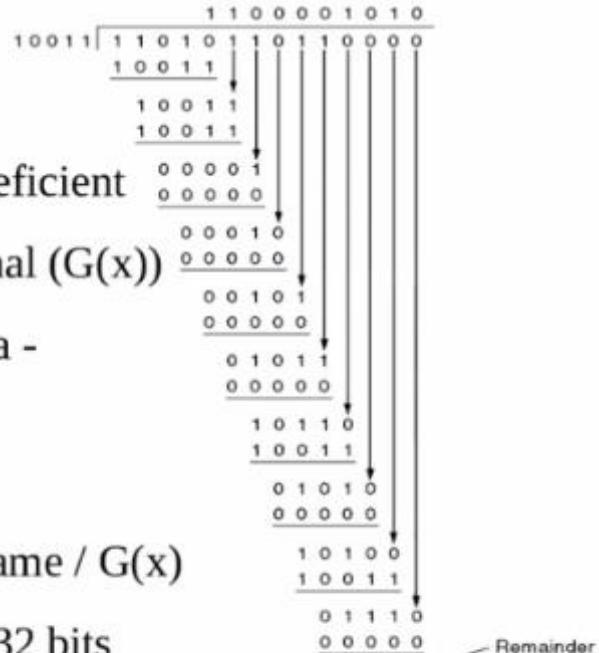
Suponiendo que hay una ráfaga de ruidos y modifíco en la "e", el 101 por 011, y abajo en vez de ser 1110 llegó 1101. En la "e" no habría detectado el error y en la "t" tampoco porque mantienen la paridad.

La idea es tomar un bloque y calcular paridad para ese bloque. En vez de checar el byte entero de la "N" y calcular su paridad, chequea el primer bit y en vez de sumarle al segundo se salta 7. Entonces arma Checksum hacia abajo, va entrelazando paridades siendo más robusto para detectar errores. Checksum es el resultado que acompaña al dato.

En el ejemplo detecta el error, no sabe dónde.

Error-Detecting Codes - CRC

Frame : 1101011011
Generator: 10011
Message after 4 zero bits are appended: 11010110110000



Calculation of the polynomial code

- Data → polynomial with 0/1 coefficient
- Agree in a Generator polynomial ($G(x)$)
- Checksumed frame = $G(x) / \text{data} - \text{reminder}$
- Send Checksumed frame
- Receiver divide Checksumed frame / $G(x)$
- 802.3 (Ethernet) Estandar $G(x)$ 32 bits

Código de redundancia cíclica CRC.

Defino un polinomio que tenga coeficientes ceros y unos de grado N. Tomo el mensaje que quiero mandar y le agrego al final ceros para la cantidad de bits que tiene ese polinomio y hago una operación de división entre el que se generó y mis datos menos el resto, y eso es lo que envío.

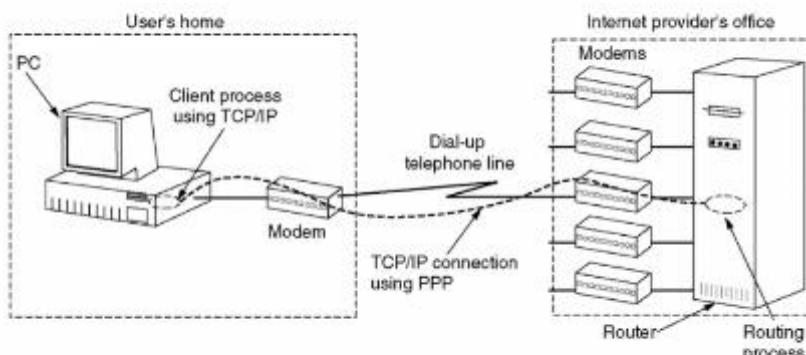
Frame es el mensaje original, el polinomio generador debe conocerlo tanto el emisor como el receptor, hay varios estándares de la IEEE. Al polinomio lo divido en mi mensaje, y el resto es lo que envío.

En el receptor tengo que hacer la operación inversa. El receptor divide la trama en el polinomio y le debería dar cero, si no da cero hay error.

Lo usa 802.3 o Ethernet para hacer trama, y en el polinomio usan uno de 32bits estándar.

El receptor lo va calculando a medida que le van llegando los bits y lo va comparando con lo que le envía el emisor.

The Data Link Layer in the Internet



A home personal computer acting as an internet host.

Protocolo PPP nos conectaba a internet en las viejas épocas, donde teníamos un modem generalmente por puerto serie, tenía un proceso en la PC que se conectaba con otro proceso que se encontraba en el otro lado.

Esto se realizaba usando un protocolo llamado punto a punto. Es como si mi PC estuviese en la misma red que la otra PC.

PPP – Point to Point Protocol

Bytes	1	1	1	1 or 2	Variable	2 or 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Payload []	Checksum	Flag 01111110

The PPP full frame format for unnumbered mode operation.

Utiliza un inicio de trama que es 7E, el inicio y fin de trama son iguales, no está puesto en la imagen pero usa relleno de bits, utiliza solo detección de errores Checksum poniéndose de acuerdo entre emisor y receptor para saber si usaran 2 o 4 bytes.

Otra cosa que la capa de enlace debe solucionar es a quien le envío los datos, en el caso de las redes punto a punto tengo una PC conectada con otra, hay un campo de dirección pero siempre va puesto en unos.

Protocolo me dice a qué protocolo de la capa superior debería mandarle los datos, porque si el emisor lo envía con determinado protocolo el receptor debe usar el mismo para entenderse.

SUBCAPA DE CONTROL DE ACCESO AL MEDIO (página 269)

Chapter 4

The Medium Access Control Sublayer

Las redes pueden dividirse en dos categorías: las que utilizan conexiones punto a punto y las que utilizan canales de difusión. Este capítulo trata las redes de difusión y sus protocolos. En cualquier red de difusión, el asunto clave es la manera de determinar quién puede utilizar el canal cuando hay competencia por él.

Los protocolos usados para determinar quién sigue en un canal multiacceso pertenecen a una subcapa de la capa de enlace de datos llamada subcapa **MAC (Control de Acceso al Medio)**.

Como habíamos visto en la clase pasada, la capa de enlace tenía que ver con entramar los datos, dado que no sabíamos que largo iban a tener, de esta manera poder procesarlos, hacer control de flujo o gestión de errores (corrección y detección).

Para entramarlo había que poner un encabezado, una bandera o lo que fuese necesario.

Vamos a ver una funcionalidad que podrían o no tener una capa de enlace, no siempre nos vamos a encontrar con lo que vamos a ver hoy, y esto es porque la pregunta sería:

¿Por qué una subcapa?

WHY A SUB-LAYER ?

- Point-to-Point Networks
 - No problem accessing the medium
- Broadcast Networks
 - Many nodes
 - Competition for the channel
 - Define protocols to access the channel -> MAC

Cuando tengo redes punto a punto, estos temas que vamos a ver no son de interés, tengo un receptor de un lado y cuando el emisor transmite le llega el mensaje y fin del problema.

El problema que vamos a ver hoy es cuando tengo redes de difusión o de broadcast, un canal compartido, un cable, una frecuencia inalámbrica. Y esto es porque no tengo dos nodos, tengo varios, y se va a tener que negociar para que todos puedan interactuar sin conflicto.

Protocolos MAC, control de acceso al medio.

The Channel Allocation Problem

- Static Channel Allocation in LANs and MANs
 - FDM and TDM
 - Constant number of nodes
 - Heavy load traffic
 - Queue theory ($T = 1 /(\mu C - \lambda)$)
 - Dynamic Channel Allocation in LANs and MANs
 - Some considerations

Una forma fácil sería hacer una asignación estática FDM (Multiplexión por división de frecuencia) y TDM (Multiplexión por división de tiempo), esto tiene varias limitaciones, es raro ver esto en redes, no se usa. Porque voy a necesitar un número fijo de nodos, porque si voy a hacer ranuras de tiempo o canales de distintas frecuencias no pueden ser infinitos y debería tener un número constante porque son los que tengo para asignar. Por otro lado deberían transmitir los nodos todo el tiempo para poder aprovecharlo, si no sería bastante ineficiente.

Muestra una formula del delay que tengo en un canal y está demostrado que si disminuyo el ancho de banda del canal y tengo varios canales aumenta exponencialmente el delay de todo el sistema. Es por esto que normalmente se hace de modo dinámico.

Asignación dinámica de canales en LANs y MANs (página 249)

Dynamic Channel Allocation in LANs and MANs

- Station Model.
 - N independent stations.
 - Single Channel Assumption.
 - Single Channel for all the communications.
 - All stations equally capable.
 - Observable Collision.
 - Two simultaneous transmission. Retransmission later.
 - All stations can detect it.
 - Continuous or Slotted Time.
 - Transmission can begin at any instant
 - Transmission can begin at start of a slot
 - Carrier NO Carrier Sense.
 - Station can tell if channel is in use before transmitting
 - Station just transmit. Later determine if was successful
- Modelo de estación
 - Supuesto de canal único
 - Supuesto de colisión
 - Tiempo continuo
 - Tiempo ranurado
 - Detección de portadora
 - Sin detección de portadora

Todas las estaciones que van a transmitir y recibir son independientes, dado que las redes están compuestas por nodos y computadoras independientes.

Hay un solo canal que va a estar compartido por todos los nodos y todos los nodos que van a transmitir tienen la misma capacidad de transmisión.

Si tengo un **medio compartido**, las ondas electromagnéticas se van a superponer, si son niveles de tensión también, entonces termina destruyéndose la información, de alguna manera se deben controlar esas colisiones y de haberlas hay que retransmitir.

Un **tiempo continuo ranurado** (un reloj que va marcando un inicio de ranura), si es continuo significa que los datos pueden enviarse cuando estén mientras que si es ranurado va a iniciar al inicio de esa ranura.

Censado de portadora o no, si censan portadora los nodos podrían saber si hay actividad en el canal y si no hay actividad en el canal pueden transmitir. Podría pasar que de acuerdo al tipo de canal no se pueda censor portadora y es otro de los temas a tener en cuenta a la hora de analizar esto.

Protocolos de acceso múltiple (página 251)

Multiple Access Protocols

- ALOHA
 - Carrier Sense Multiple Access Protocols
 - Collision-Free Protocols
 - Limited-Contention Protocols
 - Wireless LAN Protocols
- ALOHA
 - Protocolos de acceso múltiple con detección de portadora
 - Protocolo libre de colisiones
 - Protocolos de contención limitada
 - Protocolos LANs inalámbricas

Se conocen muchos algoritmos para asignar un canal de acceso múltiple. En las siguientes secciones estudiaremos una muestra representativa de los más interesantes

Vamos a ver el primero de los **protocolos que es inalámbrico y se llama ALOHA**.

ALOHA PURO (página 251)

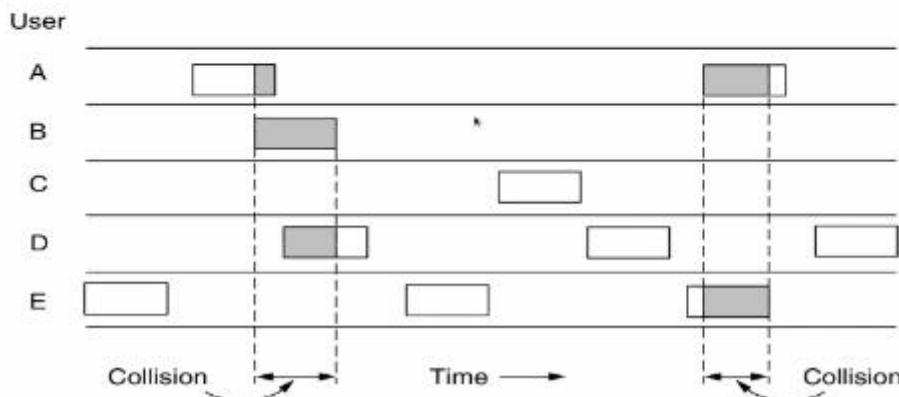
Pure ALOHA

Contention systems (a common channel that can lead to conflicts)

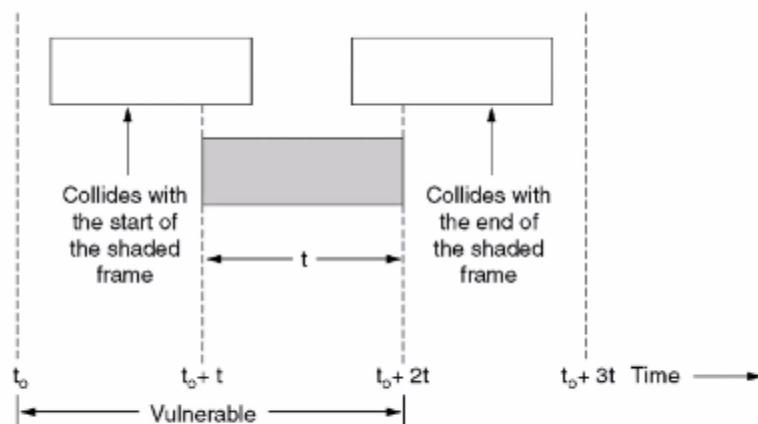
In pure ALOHA, frames are transmitted at completely arbitrary times.

Once the sender transmit, waits for the Central to retransmit the frame.

In case of lost, the sender waits a random time and retransmit



Pure ALOHA (2)



Vulnerable period for the shaded frame.

$0 < N$ frames per time of a frame < 1

Es un protocolo creado en los 70, para dar datos en los archipiélagos, no había manera de enviar datos por cables submarinos entonces decidieron transmitir los datos por equipos de radio, todos a la misma frecuencia, y hay un nodo central que cada vez que recibe un dato lo retransmite, como condición todos los nodos deben tener visibilidad con el nodo central.

El tema es como saber si hay colisión. Uno transmite y el nodo lo recibe, este luego lo retransmite. Entonces si un nodo transmite y luego se queda escuchando y llega retransmitido su mensaje es porque llegó correctamente, caso contrario vuelve a transmitir los datos, esto lo hace en un tiempo aleatorio para tener menos posibilidades de volver a colisionar.

No tiene mucha eficacia con el uso del canal.

Slotted ALOHA

Divide time into discrete intervals - SLOTS

Sincronize all the stations

A station must wait the start of a slot to send data

La siguiente mejora se llama ALOHA RANURADA.

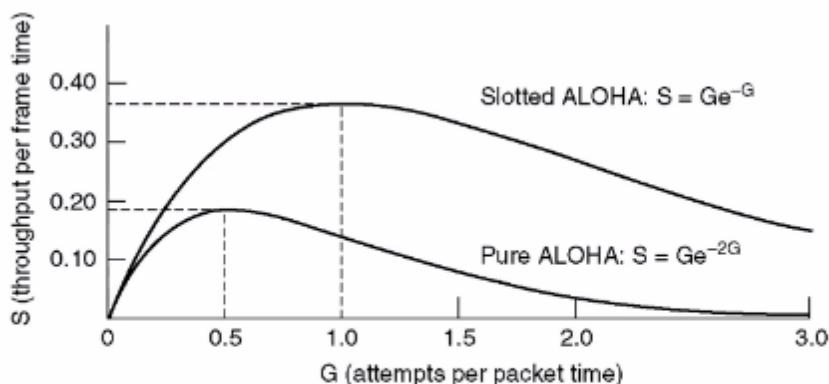
Pone ranuras de tiempo, los nodos una vez que tengan datos tienen que esperar que inicie una ranura de tiempo.

El nodo periódicamente envía una trama muy pequeña que se llama beacon (faro). Si tengo datos para transmitir no comienzo a hacerlo hasta el momento de recibir la trama. En este caso si hay dos nodos que estén esperando la trama para comenzar a transmitir habría colisión.

Pero en este caso la trama a ser enviada solo es vulnerable al comienzo, no como en el caso anterior donde todo el tiempo el dato a enviar era vulnerable si otro comenzaba a transmitir. Si comienza bien a transmitir ya no habrá colisiones dado que nadie puede comenzar a transmitir sin tener la trama de habilitación.

Le agrego más delay pero estoy evitando más colisiones.

Pure ALOHA (3)



Throughput versus offered traffic for ALOHA systems.

$$G = N + Ret$$

$$S = P_0 G$$

El libro muestra algunas graficas con una distribución probabilística de datos a transmitir. En el eje vertical pone la tasa de salida, la cual se puede ver que es bastante mala, y en el horizontal la cantidad de tramas que tiene que transmitir más la cantidad de tramas que tiene que retransmitir por colisiones. G es la cantidad de tramas totales a transmitir.

A medida que aumenta la cantidad de tramas para transmitir bajara la eficiencia porque abra más colisiones.

Carrier Sense Multiple Access

- Station listen the carrier and act accordingly
- 1-persistent CSMA
 - Listen, if idle -> transmit
 - Busy -> wait until idle, then transmit (with 1 probability)
 - Collision -> wait a random time
 - Impatient stations
 - Propagation delay (bandwidth-delay product)
- Nopersistent CSMA
 - Listen, if idle -> transmit
 - Busy -> wait random time
 - Better utilization but longer delay
- P-persistent CSMA
 - Slotted channel
 - Busy -> wait until idle, then transmit with p probability. Else wait next slot

CSMA, estos protocolos son todos cableados (coaxil, par trenzado).

El censado de portadora se refiere a que antes de transmitir me fijo si hay alguien transmitiendo y si puedo transmitir.

Esto comenzó con un cable coaxil y todos los ordenadores se conectaban a este mismo cable (MULTIPLE ACCESS).

El primero se llama **persistente CSMA**.

Si el nodo A tiene datos para enviar primero censa si hay portadora, actividad en el cable, si no lo hace transmite y si hay actividad se queda censando hasta que dejen de transmitir.

Pueden producirse colisiones en el caso de que dos equipos quieran transmitir y hay uno transmitiendo, ambos estarán censando y cuando el 3ro deje de transmitir se producirá colisión entre los otros dos. Son nodos impacientes, quieren transmitir ni bien puedan.

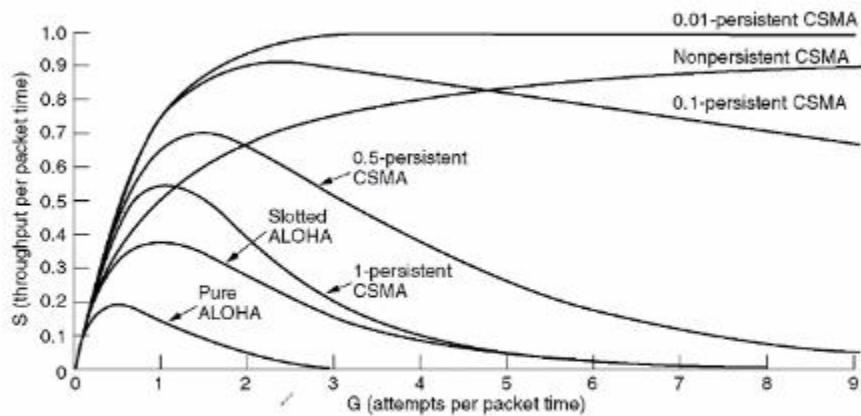
El 2do es el **no persistente CSMA**.

Es un poco más sencillo, si tengo que transmitir escucho el canal, si no hay ruido transmito, si hay alguien no espero, directamente espero un tiempo aleatorio y luego vuelvo a censar si hay portadora. Con esto evito las colisiones pero tengo más delay porque si hay mucho tráfico intentará muchas veces hasta poder transmitir los datos.

El 3ro es un Mix entre medio de los dos anteriores, **persistente P CSMA**.

Esto se aplica para canales **ranurados** donde hay un inicio de trama, entonces los nodos si tienen datos esperan esa trama, si van a transmitir va a depender de una probabilidad P, si decide no transmitir espera el inicio de la siguiente trama. Probabilidad de 0.5.

Persistent and Nonpersistent CSMA



Comparison of the channel utilization versus load for various random access protocols.

Grafico comparativo como el anterior con los mismos ejes.

Se puede ver que ambos ALOHA tienen una tasa bastante mala, el persistente 1 mejora bastante pero al tener muchos paquetes para enviar tendrá muchas colisiones y empeora.

El persistente 0.5 aumenta un poco pero tengo más delay.

CSMA con detección de colisiones – CSMA/CD (página 256)

Los protocolos CSMA persistentes y no persistentes ciertamente son una mejora respecto a ALOHA porque aseguran que ninguna estación comienza a transmitir cuando detecta que el canal está ocupado. Otra mejora es que las estaciones aborten sus transmisiones tan pronto como detecten una colisión. En otras palabras, si dos estaciones detectan que el canal está inactivo y comienzan a transmitir en forma simultánea, ambas detectarán la colisión casi de inmediato. En lugar de terminar de transmitir sus tramas, que de todos modos están irremediablemente alteradas, deben detener de manera abrupta la transmisión tan pronto como detectan la colisión. La terminación pronta de tramas dañadas ahorra tiempo y ancho de banda.

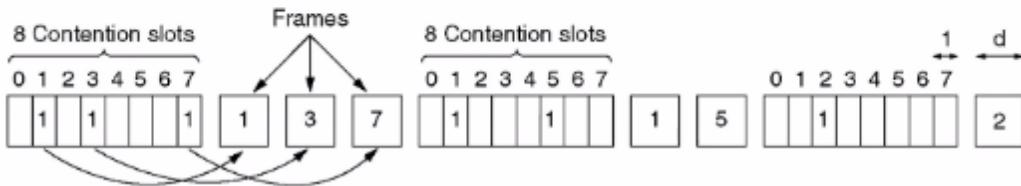
Protocolos libres de colisión.

En la gráfica no está pero el eje horizontal debería tener el tiempo.

En los casos anteriores podía haber colisiones pero en estos protocolos directamente no habrá colisiones.

Collision-Free Protocols

Bit-map protocol



El primero se llama **protocolo de mapa de bit**.

Suponiendo que tengo 8 estaciones que quieren utilizar el mismo medio (cableado o inalámbrico), tendré un inicio de trama o inicio de ranura, en este grafico lo llama slot de contención, donde cada nodo si tiene datos a transmitir se pone en 1 a la ranura que le toca esa trama. En el caso de la imagen quieren transmitir el nodo 1,3 y 7.

Entonces habrá una ranura de tiempo para 1, para el 3 y otra para el 7. Cuando se terminan todos los datos a transmitir, vuelve a hacer una ranura de contención donde estará en 1 los nodos que quieran transmitir, como en la imagen el 1 y el 5.

Si todos quieren transmitir estarían todos en 1. De esta manera nunca habría colisiones. Las ranuras de tiempo no son fijas, se las voy dando a quienes necesiten.

Si tengo muchos nodos la trama de contención va a ser más grande y si transmiso pocos datos pueden ser más grande las tramas de contención que los datos y terminaría usando más el canal para enviar tramas de control.

TOKEN PASSING – No está en el libro

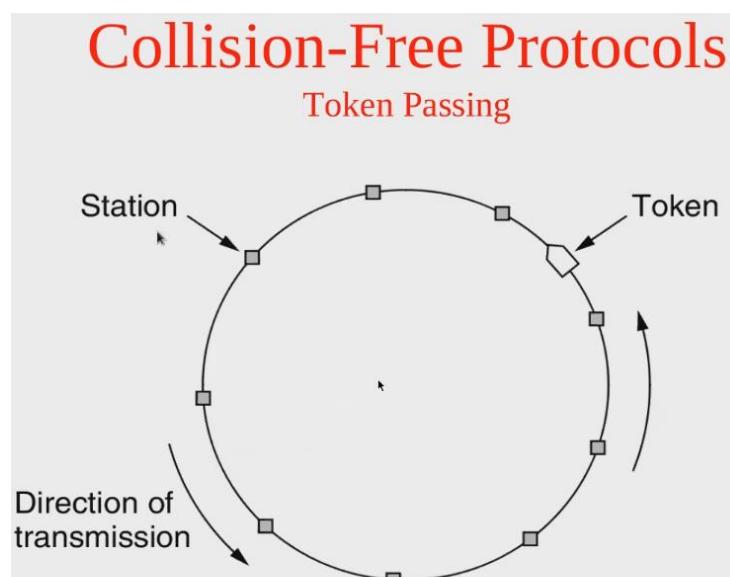


Figure 4-7. Token ring.

The token represents a permission to transmit
Token ring defines the order the token pass

Este es otro mecanismo libre de colisión llamado Token Passing o Token Ring.

Los nodos están interconectados, este Token es una secuencia prestablecida, una trama prestablecida pequeña con un valor determinado. Entonces el nodo A se lo envía al nodo B, y este al C, es una red cerrada.

Si A quiere mandar datos, no puede hacerlo hasta que no le llegue el Token, cuando este le llega, no lo reenvía, lo saca de la red e inyecta sus datos, estos dan toda la vuelta en la red, cuando llegan nuevamente a "A" los saca y vuelve a inyectar el Token, el cual seguirá dando la vuelta hasta que otro lo necesite.

Protocolo de conteo descendente binario (página 260)

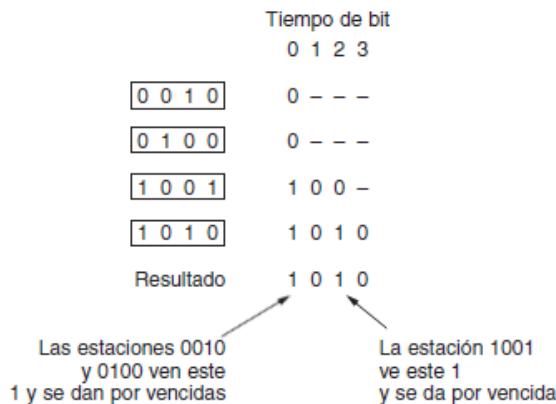


Figura 4-7. Protocolo de conteo descendente binario. Los guiones indican silencios.

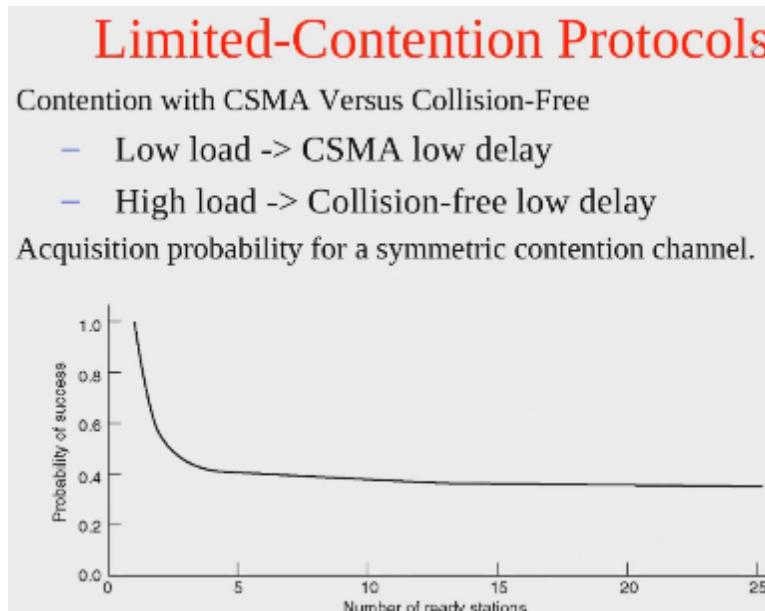
Otro protocolo anticolisión se llama conteo descendente binario.

Se asigna una dirección. En este ejemplo se tienen hasta 16 posibles y hay 4 que quieren escribir en un momento determinado, esto es ranurado entonces al inicio de la ranura el nodo que quiere escribir pone su dirección, entonces el canal va a hacer una OR entre ellos, entonces suponiendo que inicia la ranura con 4 nodos que quieren transmitir:

En el primer instante de tiempo el primer nodo pone un 0, el 2do un 0, 3ro un 1 y 4to un 1. Los dos primeros nodos no van a seguir porque ellos tenían 0 y en el canal aparece un 1. En el 2do instante siguen los dos últimos, si ambos ponen 0 seguirán en la contención, en el siguiente instante el ultimo pone un 1 entonces el 3ro deja de intentar. Al final el último pone el último 0 y esa es la dirección del nodo 1010 que va a transmitir en ese momento.

Hay que tener en cuenta que el nodo que tenga más unos tendrá más prioridad para transmitir.

Protocolos de contención limitada

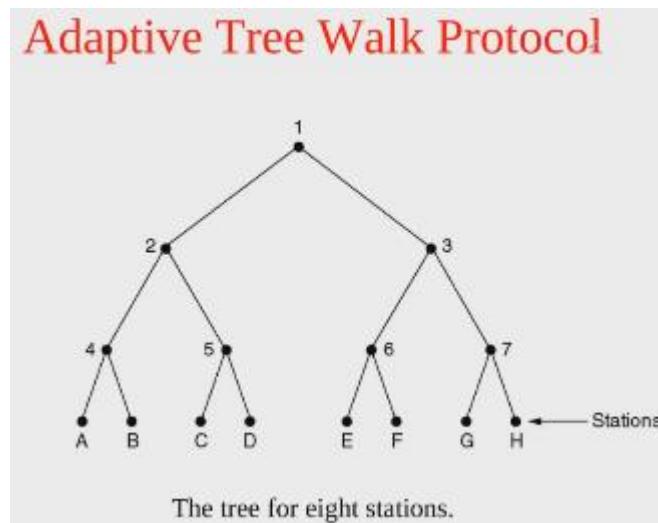


Si comparamos los protocolos libres de colisión con los CSMA

CSMA son mejores cuando hay poca carga, tienen menos delay, tiene más performance o más tasa de transferencia.

Cuando hay mucha carga los protocolos CSMA tienen muchas colisiones, donde los protocolos libres de colisión funcionan mejor.

Protocolo de recorrido de árbol adaptable (página 263)



Entonces muchos se preguntarán porque no usamos un sistema adaptativo, cuando hay mucha carga usar libre de colisión y cuando haya poca carga sea CSMA. Y esto es lo que se plantea en este protocolo de camino adaptativo, o de paso adaptativo.

Si tengo 8 estaciones tengo un árbol invertido. Si más de un canal quiere transmitir habrá competencia, de haber 1 hará uso del canal directamente pero si hay varios lo que hace es que compitan solo los que está debajo de 2, una vez que comunique debajo de 4 pasa a 5 para que comunique el que necesite debajo de 5 y hace lo mismo luego debajo de 3.

En definitiva si hay mucho para transmitir primero el A, luego el B y así hasta llegar a H.

Cuando hay muy pocos directamente los deja que transmitan.

Protocolos de LANs inalámbricas (página 267)

Wireless LAN Protocols

Not applicable CSMA -> activity near the sender, not receiver

- Hidden terminal problem
- Exposed terminals problem

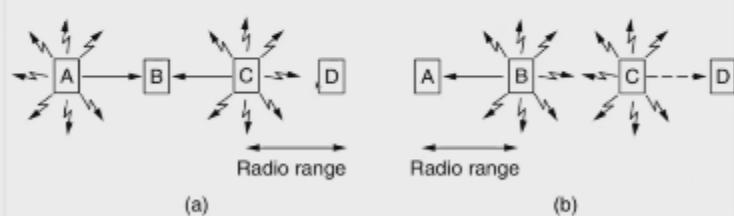


Figure 4-11. A wireless LAN. (a) A and C are hidden terminals when transmitting to B. (b) B and C are exposed terminals when transmitting to A and D.

Vamos a ver los principios básicos de protocolos inalámbricos, no lo veremos en detalle porque deberíamos ver el protocolo 802.11 WIFI pero es bastante largo.

Las redes o protocolos inalámbricos no se puede aplicar censado de portadora y esto se debe principalmente a que puede haber uno transmitiendo cuando yo quiera hacerlo y como no está a mi alcance no lo escucho.

Cuando ceno, ceno la actividad o ceno la portadora por la que estoy por transmitir dado que el que transmite censa antes. Resulta que habría que censarlo en el receptor porque puede ser que otro este transmitiendo pero no me llegue a mí. A esto se le llama **terminal oculto**.

En la imagen "a" se ven 4 nodos, el rango de radio llega solo a un nodo, es decir que A puede transmitirle a B pero no alcanza a C, B alcanza a C pero no a D.

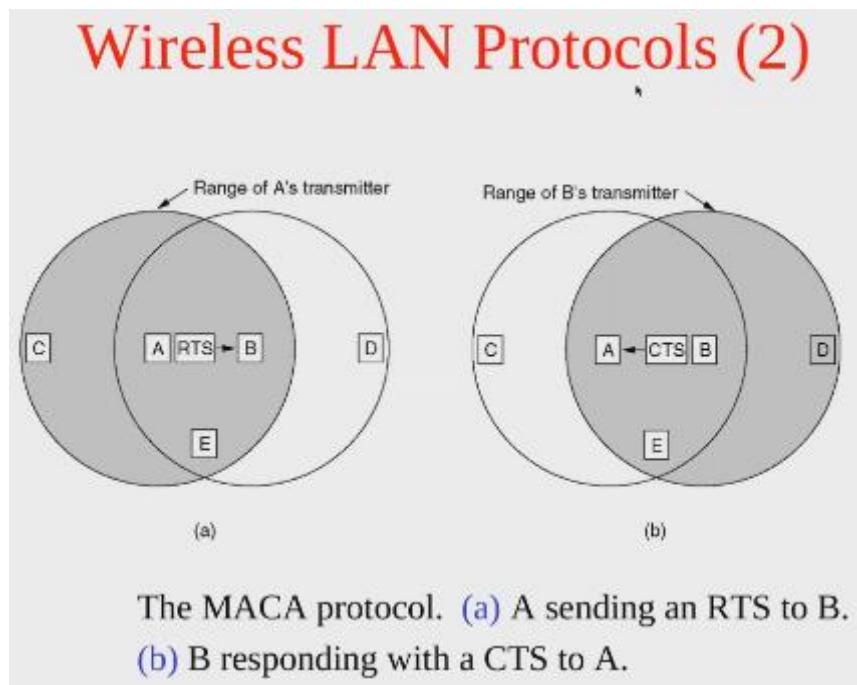
Si A quiere transmitirle a B, censa y no ve nadie transmitiendo y le transmite, pero C también ceno y nadie está transmitiendo en su contorno y le transmite a B, y el problema es que B es receptor en ambos casos y está recibiendo por ambos lados y se producirá una colisión, donde A ni C se enteran. Este es el problema de terminal oculta.

En "b" es el problema de **terminal expuesta**, es al revés, no podría transmitir porque no puedo por colisión.

B le quiere transmitir a A, entonces B censa en su periferia y se encuentra que C le está transmitiendo a D, entonces B no le transmite a A y no es lo lógico porque no habría colisión.

Como conclusión no puedo censar canales inalámbricos.

MACA y MACAW (página 269)



Para solucionar esto se usa un protocolo MACA pero es similar al que utiliza 802.11.

Si "A" tiene que mandarle un mensaje a B, antes de enviar un mensaje le envía una trama muy pequeña con un patrón especial (RTS: Solicitud de envío) que indica que quiere enviar información a B.

Entonces A se lo envía a B pero también lo recibe C y E, por lo tanto estos ya saben que A va a enviar datos, de esta manera ni C, B y E van a enviar datos.

Si B está listo para recibir le responde con un CTS (Libre para envío), listo para recibir datos. En este caso este CTS si lo recibe también D, entonces tampoco va a transmitir porque sabe que B está esperando que A le envíe datos.

A pesar de estas precauciones, aún pueden ocurrir colisiones.

ETHERNET (página 271)

Ethernet

- Ethernet Cabling
- The Ethernet MAC Sublayer Protocol
- Ethernet Performance
- Switched Ethernet
- Fast Ethernet
- Gigabit Ethernet
- IEEE 802.2: Logical Link Control

Vamos a ver como es la trama de Ethernet, ver un caso de uso de la subcapa MAC.

Utilizaremos los términos "Ethernet" e "IEEE 802.3" de manera indistinta.

Cableado Ethernet (página 271)

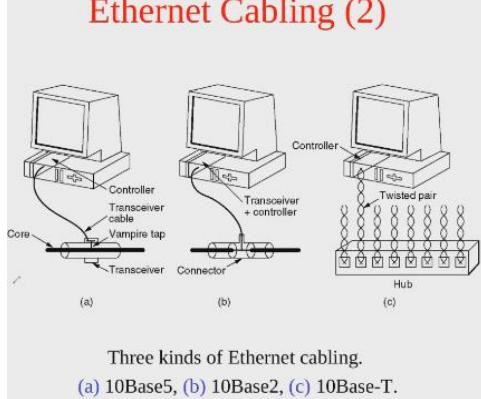
Ethernet Cabling

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

The most common kinds of Ethernet cabling.

Acá habla un poco de las normas pero es viejo esto.

Ethernet Cabling (2)



Three kinds of Ethernet cabling.

(a) 10Base5, (b) 10Base2, (c) 10Base-T.

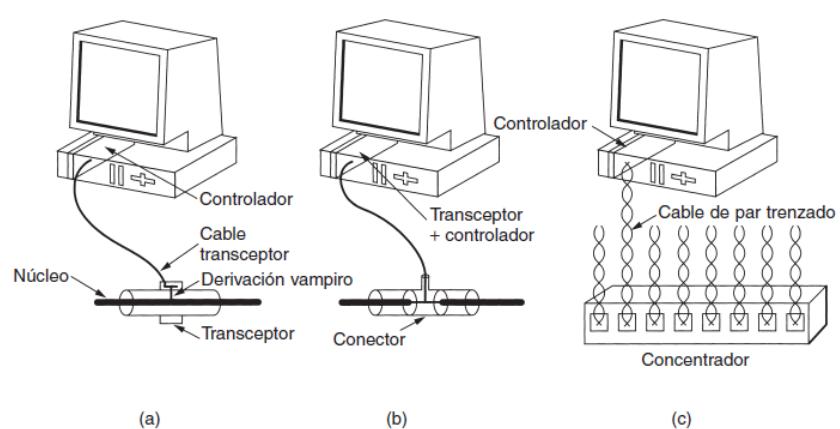


Figura 4-14. Tres tipos de cableado Ethernet. (a) 10Base5. (b) 10Base2. (c) 10Base-T.

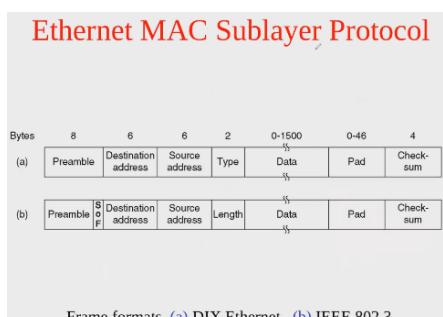
Originalmente era un cable coaxil bastante grueso y duro, complicada la instalación, y todos los nodos se conectaban ahí con un conector tipo vampiro que se clavaba al vivo “a”.

Como era muy complicada la instalación pasaron a un cable tipo coaxil más fino y flexible y con conectores en vez de vampiros, esto facilitaba la instalación, lo malo es que los conectores eran bastante débiles en cuanto a su conexión entonces si se abría el circuito ya no tenía una guía de onda y desconectaba las máquinas de la red “b”.

Pasaron a utilizar un **concentrador HUB**, y cada máquina se conecta en **estrella a ese concentrador**.

En las dos primeras imágenes se entiende que era un BUS compartido, en el 3ro sigue siendo un medio compartido, solo que en vez de usar un coaxil utilizan dos pares trenzados. En cada una de las maquinas sube por el par trenzado y baja llegando a la segunda maquina donde sube por un par trenzado y baja. En el caso de que una de las maquinas no este internamente se puentea. Físicamente es una estrella pero lógicamente sigue siendo un BUS.

El protocolo de subcapa MAC de Ethernet (página 275)



Bytes	8	6	6	2	0-1500	0-46	4
(a) Preámbulo		Dirección de destino	Dirección de origen	Tipo	Datos	Relleno	Suma de verificación
(b) Preámbulo	SOF	Dirección de destino	Dirección de origen	Largo	Datos	Relleno	Suma de verificación

Figura 4-17. Formatos de trama. (a) Ethernet DIX. (b) IEEE 802.3.

Este es el formato de la trama, cada vez que algo envía información en un cable coaxil o en un **cable UTP** manda una trama de este tipo.

Lo que se ve esta en función del tiempo.

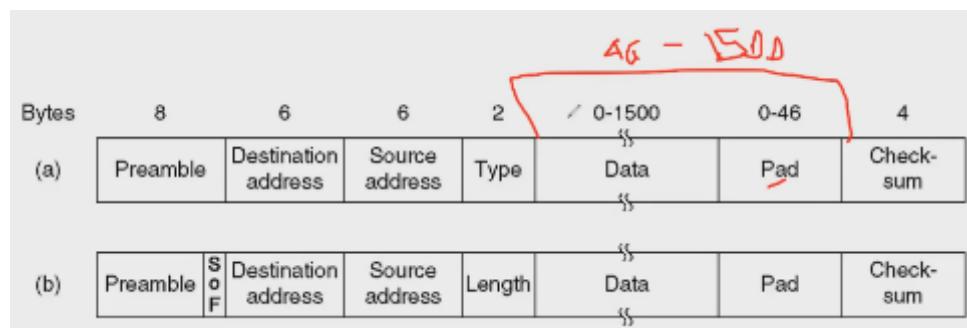
- **Primero envío 8 Bytes con un preámbulo** que son una combinación de ceros y unos 10101010, es una onda cuadrada como portadora y sincroniza todas las máquinas.
- Luego **envío la dirección destino 6 Bytes**. Las direcciones IP tienen 4 Bytes, son 6 porque aca estamos en capa de enlace, y la capa de enlace no es red, la IP es en una capa superior a esta, aca hablamos de direcciones y de capa de enlace, son las MAC y tienen 6 Bytes, tienen 3 Bytes de mayor peso para el fabricante y luego tienen 2 a la 24 combinaciones para ponerles a sus placas.
- **La dirección de origen también 6 Bytes.**
- **2 Bytes para el tipo.**
- **Datos**, donde podemos ver que es variable, podría tener hasta 1500 bytes.

- Al final un **Checksum 4 Bytes**, este tomaba un polinomio, lo dividía, tomaba el resto y lo enviaba, en el receptor se divide por este y debería dar 0.

La dirección destino es necesaria para que a las máquinas que no esté destinado el dato puedan descartarlo directamente. Imaginemos que tenemos un montón de máquinas todas transmitiendo, cada vez que me llegue una trama independientemente de quien y para quien, la placa de red debería procesarla toda, verificar el Checksum, si esto está bien luego ver el campo Type que me dice a qué capa de red le mando los datos, una vez que la capa de red revisa los datos se da cuenta que no es para mí, entonces todo este procesamiento no tiene mucho sentido. Es por esto que la dirección de destino esta al comienzo de la trama, si no es para mí ni siquiera ejecuta una interrupción del sistema operativo, es directamente ignorada.

La dirección de origen es para tener la confirmación de recepción.

El **Type** significa que los datos que recibo se los envío a la capa superior, que podría ser IP o cualquier otro protocolo distinto. En esos dos Bytes específico eso.



En ese sector tengo que tener un mínimo de 46 Bytes a 1500 Bytes como máximo. Si no tengo datos pone un relleno de 46 Bytes, si envío 3 Bytes pone un relleno de 43 Bytes. Si envío 100 Bytes no hace falta llenar.

Determino una trama que tenga un valor mínimo. ¿Para qué?

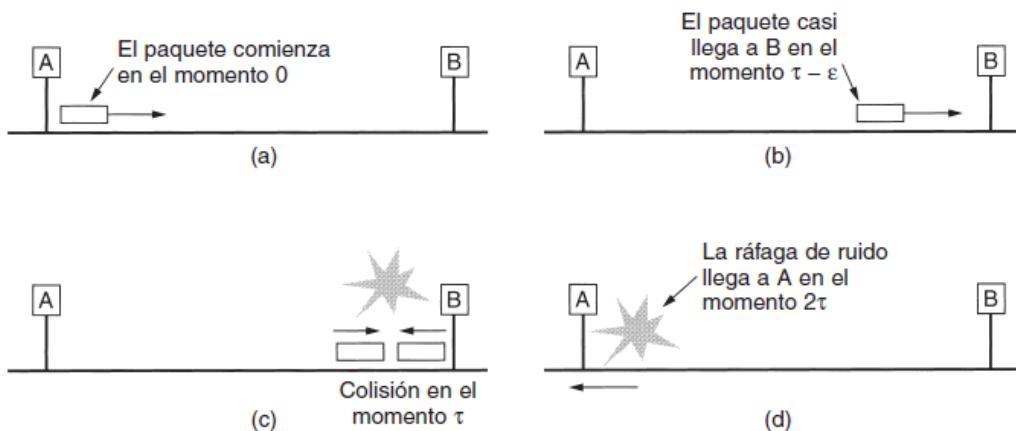


Figura 4-18. La detección de una colisión puede tardar hasta 2τ .

Esto es más que nada para **detectar colisiones**, para detectarla debería estar transmitiendo, porque la colisión es un procedimiento analógico, comparo el nivel de señal que hay en el canal con los datos que estoy transmitiendo, entonces no puedo saber que esta colisionando mi trama si ya termine de transmitir.

Vamos a suponer el caso más extremo, por norma sé que cada red tiene un largo determinado de longitud. Entonces en el instante inicial cero "a" hay un host conectado en una punta "A" y en el otro extremo final "B" (la peor de las condiciones). En el instante inicial 0 transmite una trama y va a llevar un tiempo de propagación τ , tiempo que demora en llegar de "A" a "B", si $\tau - \epsilon$ (ϵ infinitesimal) está casi llegando, "B" censa y aun no hay portadora porque no ha llegado el frente de onda entonces "B" comienza a transmitir y hay colisión los ceros y unos se van a propagar y los tendré de vuelta en "A" cuando hayan pasado 2τ .

Debería de alguna forma asegurarme que en 2τ aun este transmitiendo, para comprar lo que transmití con lo que estoy censando en el canal sea distinto.

Entonces **con un largo determinado, una velocidad de propagación determinada me es fácil calcular el tamaño mínimo de trama para que en el peor de los casos cuando haya colisión en la otra punta aun este transmitiendo datos y pueda detectar la colisión.**

Ethernet MAC Sublayer Protocol

Bytes	8	6	6	2	0-1500	0-46	4	
(a)	Preamble	Destination address	Source address	Type	Data ..	Pad	Check-sum	
(b)	Preamble	S o F	Destination address	Source address	Length ..	Data ..	Pad	Check-sum

Frame formats. (a) DIX Ethernet, (b) IEEE 802.3.

Esto que hemos visto es para el caso “a” el Ethernet.

Luego lo normalizaron y crearon el **protocolo 802.3 que es para redes cableadas de área local.**

- Tiene 7 Bytes que son de preámbulo ceros y unos, mas 1 Byte (SOF, delimitador de inicio de trama) que también son ceros y unos.
- Tiene 6 Bytes de dirección de destino.
- 6 Bytes de dirección de origen.
- Y luego cambiaron los 2 Bytes que ahora son el largo en vez del Type.
- Lo mismo para los datos y el relleno.
- Usaron el mismo polinomio de Checksum.

Puedo tener tramas del tipo “a” y del tipo “b” conviviendo en la misma red física sin problemas.

La gente de IEEE 802.3 sacó el Type, pero es un dato muy importante, porque si fuese IPv6 no funcionaría, entonces dentro de los datos, en los primeros Bytes generaron una **subcapa de la subcapa llamada LLC** y ahí van dos Bytes que dice que tipo es. Y para el caso “a” no se encuentra el largo porque esta es una trama de capa de enlace, esto arranca con un cero y uno que da una portadora, entonces cuando se acaba la portadora se acaba la trama y es por esto que no hace falta saber el largo.

Todos los códigos de capa de red tendrán un número mayor a 1500, como la longitud no puede superar esto, de esta manera ambos pueden convivir dentro de la misma red física.

Si Type es más de 1500 es una trama Ethernet, si es menor a 1500 estamos hablando de 802.3.

Switched Ethernet

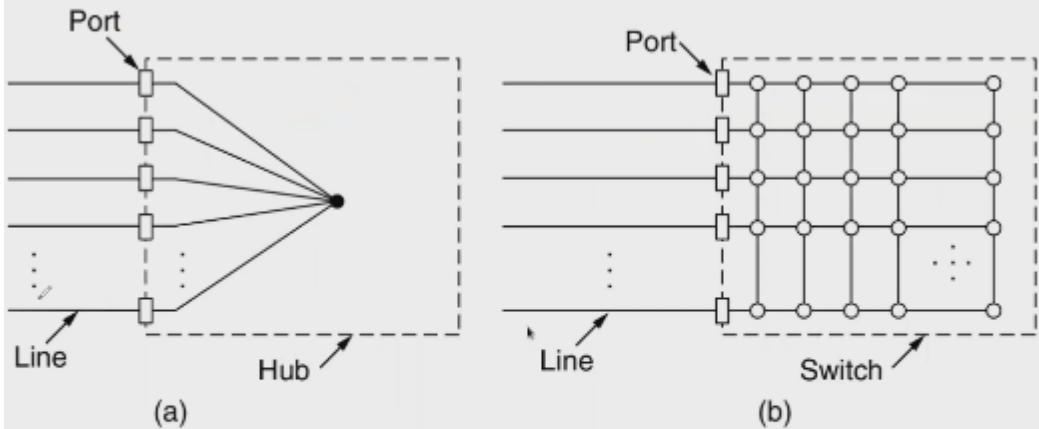


Figure 4-17. (a) Hub. (b) Switch.

Different Collision Domains

No need of CSMA

No more promiscuous mode

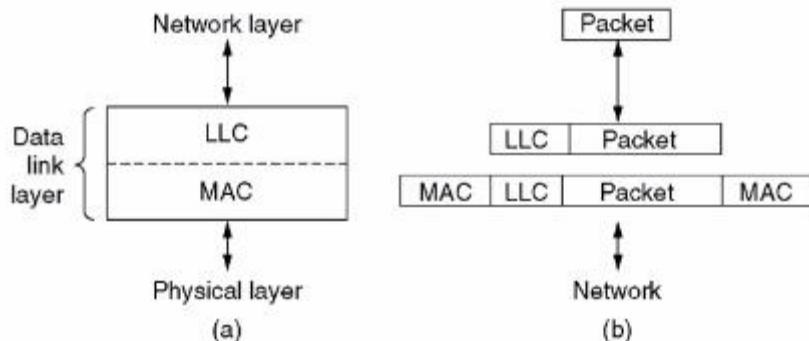
Como habíamos visto en el ejemplo del **HUB**, no importa porque puerto o boca transmita algo, siempre se va a retransmitir por todas las otras. A esto se le dice que tiene un dominio de colisión, cualquiera que escriba por cualquier lado, si alguien está escribiendo en ese mismo momento van a colisionar, porque todos se juntan en el mismo punto.

Con el **Switch** el mensaje solo llega al destinatario, además podemos tener comunicaciones simultáneas entre distintos nodos.

Los **Switch** tienen una tabla interna dinámica en RAM donde tienen que puerto o línea tiene tal MAC, cada vez que entra un paquete por una de las bocas se registra en la tabla el puerto y la MAC.

Estándar IEEE 802.2: control lógico del enlace

IEEE 802.2: Logical Link Control



(a) Position of LLC. (b) Protocol formats.

El uso típico del LLC es el siguiente. La capa de red de la máquina emisora pasa un paquete al LLC usando las primitivas de acceso del LLC. A continuación, la subcapa LLC agrega un encabezado LLC que contiene los números de

secuencia y confirmación de recepción. La estructura resultante se introduce entonces en el campo de carga útil de una trama 802 y se transmite. En el receptor ocurre el proceso inverso.

Detalla como hace con el campo que antes se llamaba Type y ahora puso el largo. Le agregan una subcapa a la subcapa MAC que se llama LLC y ahí va el Type, entonces entre la MAC y el LLC formarían todo lo que es la capa de enlace teniendo por debajo la capa física y arriba la de red.

Si vemos la trama primero iría todo el encabezado de la trama del 802.11, luego el encabezado con los datos del Type, luego el dato que era hasta 1500, y al final la última parte de la MAC CRC.

CAPA 3 CAPA DE RED (página 343)

La capa de red se encarga de llevar los paquetes desde el origen hasta el destino. Llegar al destino puede requerir muchos saltos por enrutadores intermedios. Esta función ciertamente contrasta con la de la capa de enlace de datos, que sólo tiene la meta modesta de mover tramas de un extremo del cable al otro. Por lo tanto, la capa de red es la capa más baja que maneja la transmisión de extremo a extremo.

Para lograr su cometido, la capa de red debe conocer la topología de la subred de comunicación (es decir, el grupo de enrutadores) y elegir las rutas adecuadas a través de ella; también debe tener cuidado al escoger las rutas para no sobrecargar algunas de las líneas de comunicación y los enrutadores y dejar inactivos a otros. Por último, cuando el origen y el destino están en redes diferentes, ocurren nuevos problemas. La capa de red es la encargada de solucionarlos. En este capítulo estudiaremos todos estos temas y los ilustraremos principalmente valiéndonos de Internet y de su protocolo de capa de red, IP, aunque también veremos las redes inalámbricas.

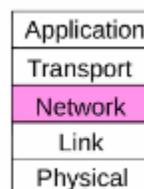
Network Layer

Chapter 5

- Design Issues
- Routing Algorithms
- Internetworking
- Network Layer of the Internet

The Network Layer

Responsible for delivering packets between endpoints over multiple links



Es la capa 3, se encarga de entregar los paquetes entre distintas redes si tenemos más de una red interconectada y esta capa es la que se las arregla para pasar datos de una a otra.

ASPECTOS DE DISEÑO DE LA CAPA DE RED (PÁGINA 343)

Design Issues

- Store-and-forward packet switching »
- Connectionless service – datagrams »
- Connection-oriented service – virtual circuits »
- Comparison of virtual-circuits and datagrams »

Vamos a ver algo de diseño llamado **Store-and-forward** que es para el ruteo.

Vamos a ver **servicios con y sin conexión** y una comparación entre ellos.

Comutación de paquetes de almacenamiento y reenvío (página 344)

Store-and-Forward Packet Switching

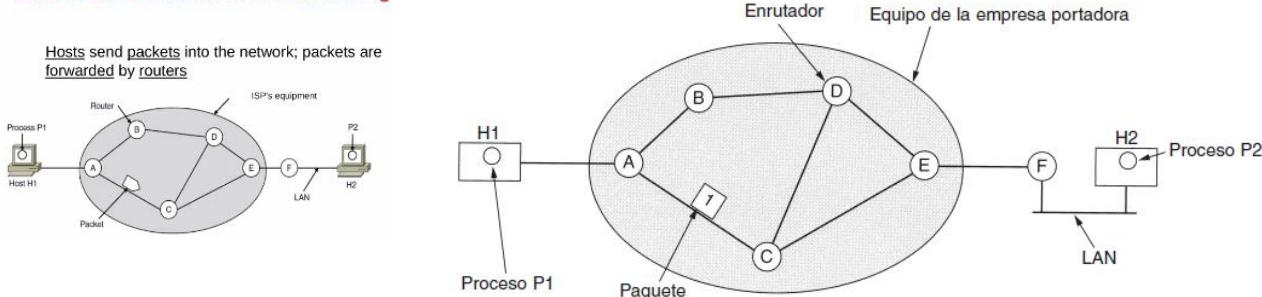


Figura 5-1. El entorno de los protocolos de la capa de red.

Aca podemos ver que hay más de una red. Podríamos considerar que el Host 1 está en una red, luego tenemos la red del proveedor y el Host 2 en otra red.

Entonces esta capa de red se tiene que encargar de que llegue el mensaje del “proceso del Host 1” al “proceso del Host 2”, pero en realidad la capa de red no se encarga de enviar cosas a los procesos, solo se encarga de que lleguen a la máquina.

Las redes están conectadas por dispositivos que entienden las capas de enlace de cada una de las redes a las que se conectan. Si tengo una red que es PPP y otra que es Ethernet de alguna manera el equipo que las conecta como por ejemplo A deberá tener una interfaz que soporte esa red PPP con la capa de enlace correspondiente y una interfaz Ethernet que soporte esa capa de enlace, y de esta manera tiene que ser cada router. Estos equipos que **comparten más de una red se llaman router o Gateway, encaminador.**

Cada vez que llega una trama (capa de enlace) a estos equipos, un byte detrás de otro o con niveles de señal, el router para procesarlo necesita tener toda esa trama de capa de enlace, ahí lo procesa y lo reenvía, entonces estos router hacen algo que se llama **Store Forwarded** (almacenamiento y reenvío), almacenan hasta que reciben toda la información, se fijan donde tienen que reenviar, este es el trabajo que hacen los router.

Servicios proporcionados a la capa de transporte

Services considerations

- Services must be INDEPENDENT of routers technol.
- Transport layer AGNOSTIC topology, number, type of routers.
- Network layer UNIQUE and UNIFORM identification
- Type of service?
 - Internet Community VERSUS Telephone Comp.

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

Algunas consideraciones:

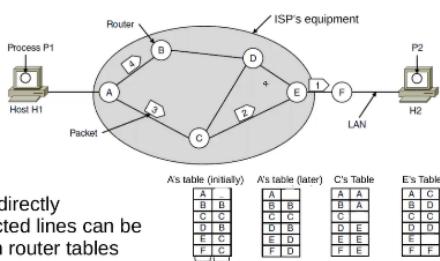
- El servicio que tiene que dar esta capa es independiente de que capa de enlace tengo abajo, no importa si en el camino tengo que atravesar una capa de enlace PPP, una 802.11, 802.3 o cualquier otra tecnología.
- La capa superior, la de transporte tampoco le debería interesar la cantidad de router o redes que atravieso. Ese es el servicio que le voy a dar a la capa superior, es decir que la capa superior se va a olvidar de todos los caminos por los que tiene que ir.
- Esta capa debería tener una identificación única para todas las maquinas, por más que tenga muchas redes interconectadas, cada máquina debería tener una identificación uniforme para empezar y única. Como habíamos visto tenemos una numeración para Ethernet, la dirección MAC de 48 bits pero esto me sirve para identificar una máquina en una capa de enlace Ethernet, si mi información atraviesa otra capa de enlace, necesito una capa superior por encima de esto que tenga un plan de numeración distinto que sea único y uniforme, como por ejemplo IP, dado que se interconectan muchas redes y se le da una numeración distinta a la capa de enlace a cada PC, única y uniforme.
- Esta capa de red podría ser orientada a conexión o no conexión. Esto es parecido al funcionamiento de internet, es un servicio sin conexión y la línea telefónica da servicio con conexión.

Implementación del servicio no orientado a la conexión (página 345)

Connectionless Service – Datagrams

Packet is forwarded using destination address inside it

- Different packets may take different paths



CN5E by Tanenbaum & Wetherall, © Pearson Education-Prentice Hall and D. Wetherall, 20

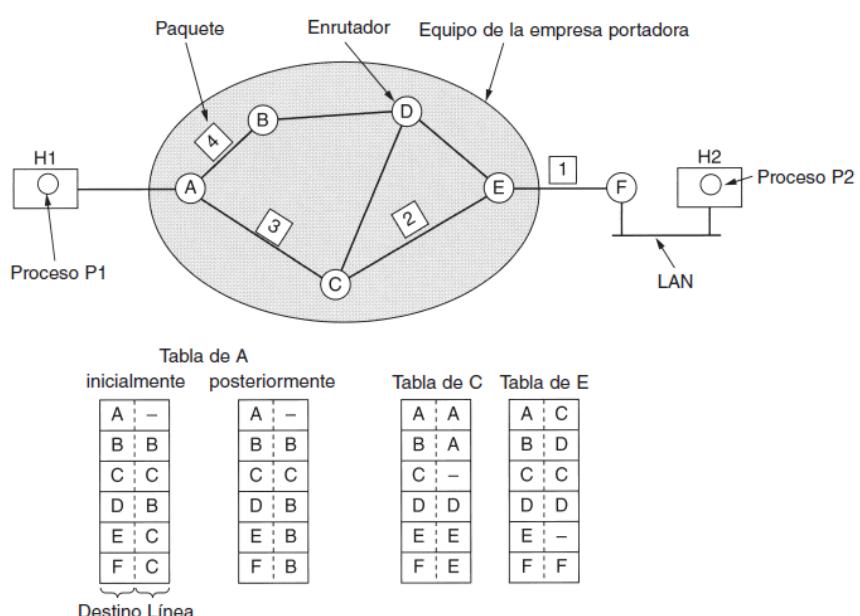


Figura 5-2. Enrutamiento dentro de una subred de datagramas.

Servicio sin conexión:

Las unidades de transmisión en capa de red se llaman datagramas o paquetes, sin conexión manda los datos sin hacer nada antes, manda la información y con suerte el host 2 lo ve.

Entonces cada vez que llega un paquete a un router debe almacenarlo y luego en función de la información que tenga lo va a reencaminar. Entonces cada paquete debe tener información de hacia dónde va. Por ejemplo, el Host 1 le quiere enviar información al Host 2, llega al router "A" como va llegando de a bits espera a que llegue todo, pero luego puede enviarlo por "B" o "C" **[pero en función de que lo envía por uno o por otro?, esto lo hace en función de a dónde quiere que vaya]**. Evidentemente tendrá que meterle más información de control a cada uno de los paquetitos diciendo al menos la dirección del host de destino. Además de esta información adicional los router deberían ser conscientes de cómo están armadas todas las redes, entonces los router tienen una tabla de rutas con esa información.

Entonces por ejemplo llega un paquete del Host 1 para el Host 2, cada uno de los router tienen por donde encaminar dependiendo el destino buscando en esas tablas. Entonces de "A" tiene que llegar a "F" entonces va hacia "C", en "C" como tiene que llegar a "F" sigue por "D", en "D" como tiene que llegar a "F" va a "E", y en "E" como tiene que llegar a "F" va a "F" para llegar a la máquina.

Esta tabla la deben tener todos previamente cargados, por cada paquete que recibe un enrutador lo debe almacenar, luego revisa la tabla con la dirección de destino y lo reencamina.

Podría pasar que se rompa un enlace, entonces algunas entradas en la tabla pueden ser inválidas porque no tengo conectividad, entonces manualmente o automáticamente va a cambiar la tabla que está contenida en el router.

Implementación del servicio orientado a la conexión (página 347)

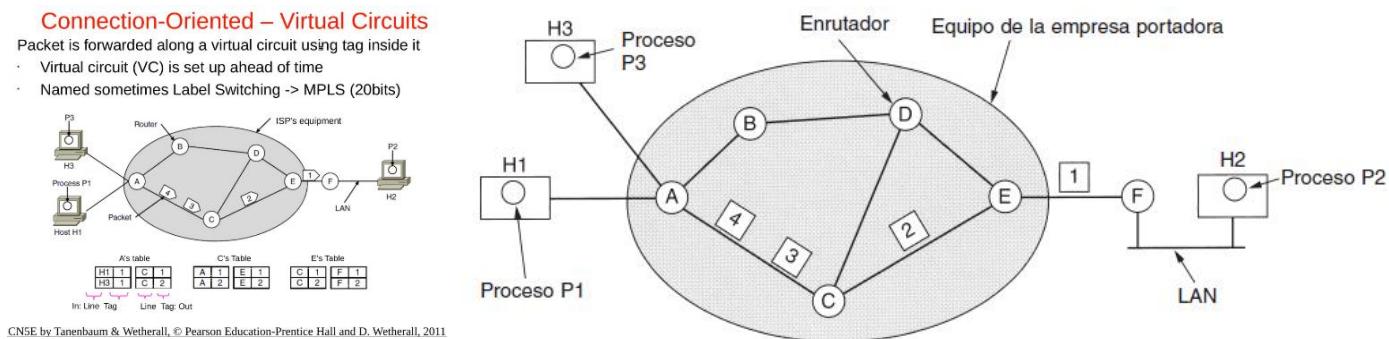


Tabla de A		Tabla de C		Tabla de E	
H1 : 1	C : 1	A : 1	E : 1	C : 1	F : 1
H3 : 1	C : 2	A : 2	E : 2	C : 2	F : 2

Dentro Fuera

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

Están los **circuitos virtuales**, lo primero que hace antes de enviar un solo paquete es crear un circuito virtual, a la información que voy a enviar se le pone una etiqueta del número de circuito que es. Entonces cada router como ya tienen el camino que tiene que hacer van siguiendo esa etiqueta.

Por ejemplo, el Host 1 tiene que enviar al Host 2, previo tengo que armar un circuito, Host 1 establece un circuito virtual con A, entonces al Host 1 le asigna el circuito virtual 1, sale por el router "C" y le llama también circuito virtual 1, se hace lo mismo con E que le llama circuito virtual 1 y lo mismo desde "E" a "F".

Una vez que se arman estas entradas en el router cuando envío paquetes le pongo la etiqueta 1 y cuando lo recibe "A" sabe que sale de ese host y todo sigue siempre el mismo camino.

En el 2do caso cuando llega a C le da la etiqueta 2 y en el resto pasa lo mismo porque ya tendrán dos circuitos.

Con esto los router no tienen que correr un algoritmo de ruteo, solo mira la etiqueta.

Si se rompe uno de los enlaces, como nadie rutea nada, solo se envía de un circuito virtual a otro, **si se corta un enlace no tengo nada que hacer hasta que lo arreglen o alguien borre ese circuito virtual.**

Comparando ambos el **primer modo es más seguro para los datos**, y este **último es más rápido dado que los router demoran mucho menos**, además como se la cantidad de circuitos se de antemano que ancho de banda tendrá cada uno de los circuitos, si tengo 10MB y creo 10 circuitos sabría que puedo asegurarle que cada uno tenga 1MB, esto me permite calidad de servicio. Son enlaces dedicados que los ofrecen telefónica por ejemplo, que pueden asegurar un ancho de banda fijo siempre.

Comparación entre las subredes de circuitos virtuales y las de datagramas (página 348)

Comparison of Virtual-Circuits & Datagrams

Issue	Datagram network	Virtual-circuit network
Circuit setup	Not required	Required
Addressing	Each packet contains the full address of the destination host	Each packet contains a short virtual circuit identifier
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is independently routed through the network	Set up all packets follow the same path
Effect of router failure	Router failure can drop packets during the transit	All packets follow the same path
Quality of service	Difficult	Easy if enough resources can be reserved in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Computational Resources in each router, size of addresses, size in routing tables, etc.

CNEC 2e Tannenbaum & Wetherall © Pearson Education, Prentice Hall and D. Wetherall, 2011

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

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

Esta es una tabla comparativa, donde en redes de datagramas (sin conexión) no necesito establecer un circuito, si tengo datos los mando, mientras que en circuitos virtuales sí.

Con respecto al **direccionamiento** cada paquete debe tener una dirección destino, origen puede ser para responder, mientras que en circuitos virtuales es solo un número de circuito.

Como es sin conexión la red de datagramas, los router no tienen información de estado mientras que en circuitos virtuales sí.

El **ruteo** se tiene que hacer por cada paquete que pasa en redes de datagrama, mientras que en el otro se hace solo al principio del circuito virtual

Si se **rompe un router** se enteran los router del medio y cambia la ruta, en ese momento se pueden perder algunos paquetes pero luego sigue funcionando, en circuitos virtuales tengo que hacer todo nuevamente.

Si necesito **calidad de servicio** y quiero establecer un ancho de banda es difícil en datagramas, en circuitos virtuales es fácil porque se todos los circuitos que van a pasar por cada uno de los enlaces.

Control de congestión también es difícil en datagramas y es más fácil en circuitos virtuales. Congestión: si llegan más paquetes de los que mi router pueden manejar tiene un buffer de entrada que se va a llenar y cuando esto pasa se tardan.

ALGORITMOS DE ENRUTAMIENTO

Routing Algorithms (1)

- Optimality principle »
- Shortest path algorithm »
- Flooding »
- Distance vector routing »
- Link state routing »

Acá vamos a hacer un salto bastante grande y no vamos a ver algoritmos de rango dinámico, esto habla del intercambio de información entre los router y cuando ven que una ruta o un enlace andan mal va actualizando la ruta de todos los router vecinos.

Vamos a ver como encaminar los datos entre distintas redes, ruteo no vamos a ver, solo en la práctica vamos a ver ruteo estático que es poner a mano la tabla de ruta de la que hablamos.

Internetworking

Internetworking joins multiple, different networks into a single larger network

- How networks differ »
- How networks can be connected »
- Tunneling »
- Internetwork routing »
- Packet fragmentation »

Lo que vamos a ver ahora es como interconectar distintas redes de capa de enlace como si fuera una red más grande. Hay que tener en cuenta que no todas las redes son iguales. Vamos a ver como interconectamos que se denomina tunneling, ruteo internet y fragmentación.

Cómo difieren las redes (página 419)

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

Figura 5-43. Algunas de las muchas maneras en que pueden diferir las redes.

Con respecto a cómo difieren las redes, podrían ser bastante distintas, un dato, un datagrama o un paquete en la capa de red deberían ir entre un origen y un destino y van a restar en redes distintas, pero puede pasar por varios caminos, entonces esas distintas redes intermedias pueden tener características distintas.

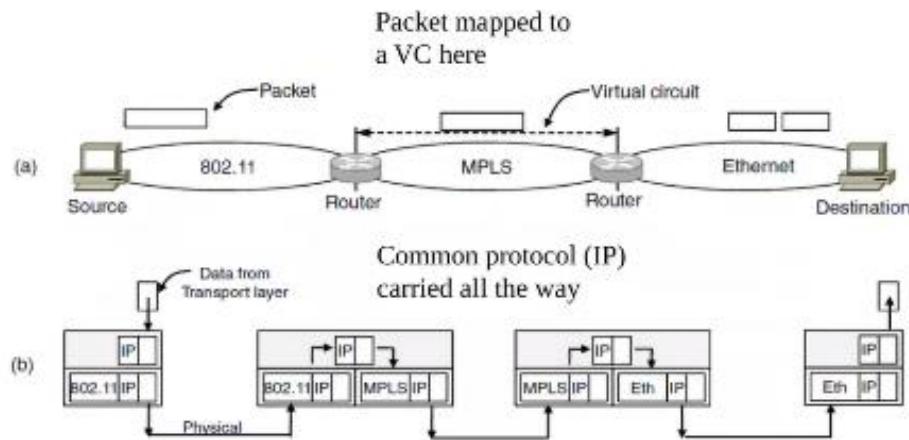
- Podría ser una red con conexión y otra sin conexión.
- Las distintas redes en capa de enlace tienen distintos tamaños, formatos, jerarquía de direcciones. La única manera es crear un nuevo esquema de direccionamiento global.
- Hay redes que son de difusión y otras que son punto a punto.
- Hay redes que soportan paquetes de 1500 Bytes y hay otras que soportan de menos, 800 Bytes.
- Una capa de enlace podría tener calidad de servicio y otras no, entonces sí quiero que un paquete viaje de un extremo a otro y pasa por varias redes como hago para asegurar el tema de calidad de servicio.
- Lo mismo pasa con confiabilidad.
- Lo mismo pasa con seguridad.

La capa de red **se debe abstraer de esto para que un paquete llegue de un extremo a otro**

Conexión de redes (página 420)

How Networks Can Be Connected

Internetworking based on a common network layer – IP



Este es un ejemplo donde vemos 3 redes (capas de enlace), una a la izquierda que es 802.11 inalámbrica, luego una capa de enlace de circuito virtual MPLS (red privada virtual o VPN) y por último tiene una capa de enlace Ethernet. Mi paquete debería atravesar todas estas redes, cada router debe entender de las tecnologías que tenga conectadas.

En "b" vemos un esquema, los datos que vienen de capa de transporte se ponen en el payload (parte de datos) de mi paquete sin conexión IP, luego el paquete IP más el payload se ponen en la parte de carga de una red wifi y viaja hasta el siguiente router, donde va a hacer el camino inverso, saca el Payload y lo mete en la capa superior, ese router debería darse cuenta que ese paquete no es para él, porque si fuera para él lo pasaría a una capa superior y ahí termina todo el proceso, pero como no es el destino final no sigue enviando los datos hacia arriba, llega a la capa de red, almacena, procesa y reenvía, como sabe que por la red que tiene que reenviarlo es MPLS, al paquete IP con los datos lo mete dentro de una trama con conexión MPLS (probablemente antes ha tenido que hacer un circuito virtual porque es con conexión), cuando llega al router del otro extremo hace el mismo procedimiento, saca el paquete IP con su Payload, no pasa el Payload a la capa superior porque sabe que no es el destino final, entonces lo mete en una trama Ethernet, viaja por la red Ethernet y finalmente llega a la máquina destino, en la capa de enlace al Payload lo envía a la capa IP, en esta capa verifica que sea para él y finalmente los datos los envía a la capa de transporte.

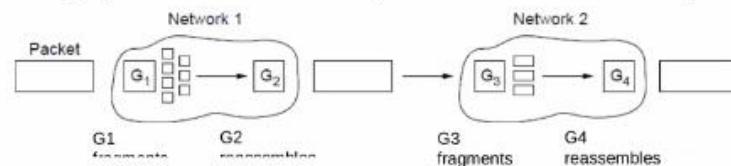
Es un ejemplo sin conexión, lo único que hay con conexión pero en capa de enlace, se les ocurrió poner en el ejemplo el MPLS que lo está considerando como capa de enlace porque está hablando con 802.11 y con Ethernet, pero MPLS en diapositivas anteriores es capa de red. Lo que hace es usarlo como capa de enlace para IP.

Fragmentación (página 427)

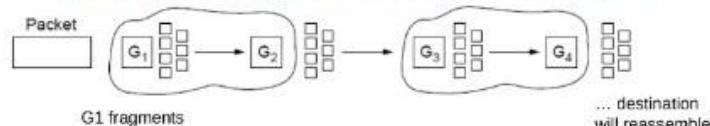
Packet Fragmentation (1)

Networks have different packet size limits for many reasons

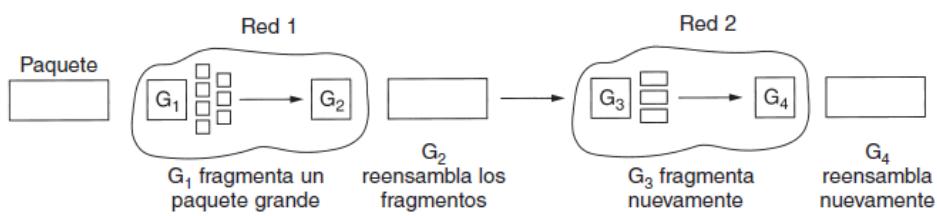
- Large packets sent with fragmentation & reassembly



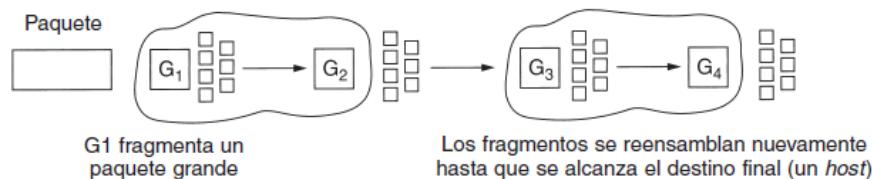
Transparent – packets fragmented / reassembled in each network



Non-transparent – fragments are reassembled at destination



(a)



(b)

Figura 5-50. (a) Fragmentación transparente. (b) Fragmentación no transparente.

Como va pasando por distintas capas de enlace podría pasar por alguna que tengan distinto MTU (unidad máxima de transferencia) o distinto tamaño máximo de carga las capas de enlace, entonces lo que envío en vez de ser una carta sea una encomienda que es grande entonces cuando toca que la lleve un cartero no le va a entrar en el morral.

Comprar un morral nuevo que sería cambiar el protocolo de capa de enlace pero sería imposible a medida que viaja el paquete, o puede partirse en partes. De esto se encarga la capa de red.

En el ejemplo el paquete llega a un router G1, viene de una red y va a pasar por otra red, pero esta red Network 1 tiene una tamaño máximo de información que puede transportar, si hablamos de capa de enlace sería cada trama, más pequeñas, entonces el paquete no entra en una trama, entonces el router G1 una vez que almaceno el mensaje como sabe por qué red enviarlo y sabe que la capa de enlace no soporta ese tamaño lo parte en 7 partes distintas y se lo manda al router destino G2.

Este G2 tiene dos alternativas, a esto se le llama fragmentación transparente o no transparente, re ensamblar todos los paquetes y enviarlo, es transparente porque nadie se entera que fue partido o fragmentado. Si tenemos otra red Network 2 que tiene otra MTU, el paquete original G3 lo fragmenta en 3 partes y el router G4 lo re ensambla y sale uno solo como el original.

A esto se le llama fragmentación transparente porque llega a destino y ni se entera que paso.

La otra alternativa **no transparente** es donde una vez que un router fragmenta como G1, sigue todo el camino fragmentado y en la máquina destino ensambla los paquetes.

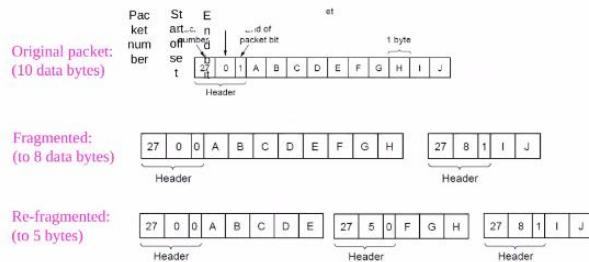
Internet usa la 2da opción, una vez que se fragmenta le deja al destino el trabajo de ensamblar nuevamente, No transparente. Y esto se da porque trabajando en transparente lo router necesitan más RAM para ensamblar el paquete nuevamente.

¿Cómo sabe cuántos paquetes son? IPv4 lo mete en el encabezado IP, IPv6 agrega un encabezado.

Debería traer información de que está fragmentado, el tamaño, el número.

Packet Fragmentation (2)

Example of IP-style fragmentation:



Número del primer fragmento elemental de este paquete

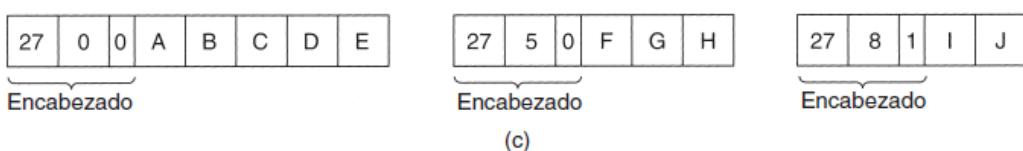
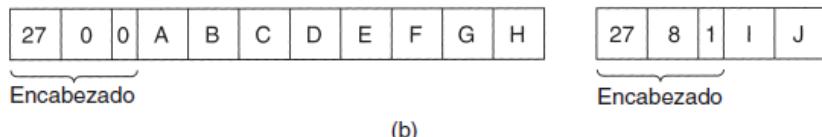
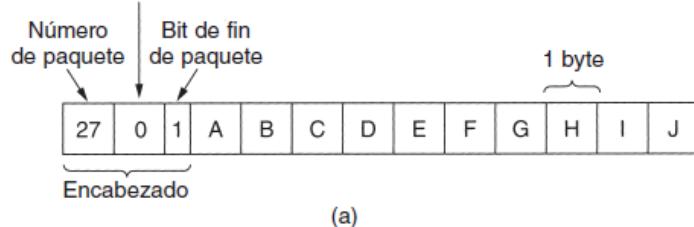


Figura 5-51. Fragmentación cuando el tamaño de datos elemental es de 1 byte. (a) Paquete original que contiene 10 bytes de datos. (b) Fragmentos tras pasar a través de una red con un tamaño máximo de paquete de 8 bytes de carga útil más encabezado. (c) Fragmentos tras pasar a través de una puerta de enlace de tamaño 5.

Yo tenía un maqueta original de 10 Bytes. El Payload son 10 Bytes, y en el encabezado tengo un par de datos, numero de paquete que sería el 27, un offset 0, y un bit que me indica que se termina el paquete.

A cada paquete tengo que ponerle un número, le pondré un offset si es una parte del paquete original (el paquete original no lleva offset), y un bit que me dice que es el último. Este sería el encabezado que le agrego al IP.

Suponiendo que pasa por una capa de enlace donde el máximo es de 8 bytes. Cuando envíe por esa capa de enlace cuyo MTU es de 8 bytes deberá partirlo en 2 donde primero entraran 8 y luego 2, con esos campos que le agrega al encabezado el paquete original es 27 ambos fragmentos con el número 27, el offset es 0 porque es justo el inicio del paquete mientras que el offset en el 2do es 8, luego el bit que me indica que hay más paquetes poniendo un 0, en 1 sería el último.

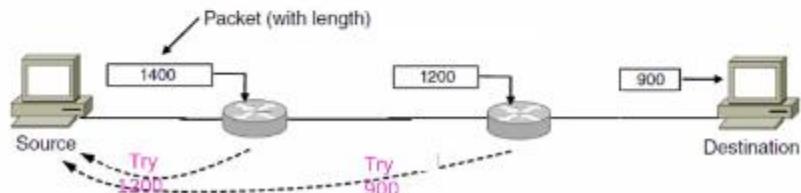
Si esto que está fragmentado pasa por otra capa de enlace que soporta 5 Bytes, serían fragmentos de fragmentos, la 2da parte pasa directo, pero la 1ra tengo que partirlo en dos, siempre usando el mismo número de paquete.

El destino tiene toda la información para ensamblarlo, numero de paquete y offset.

Packet Fragmentation (3)

Path MTU Discovery avoids network fragmentation

- Routers return MTU (Max. Transmission Unit) to source and discard large packets



Path MTU Discovery, no se utiliza mucho, antes de enviar los datos trata de adivinar cuanto es el MTU de origen a destino, el máximo Payload que se puede poner al paquete entonces se intenta armar paquetes con ese Payload o menos.

Como contras hay que recordar que en IP trabajamos sin conexión, entonces antes de enviar los datos tengo que enviar alguna información a los router para ir sabiendo cuando es el MTU de la capa hasta llegar al destino, y 2do problema es que estoy considerando que siempre voy a hacer ese camino y si algún enlace falla el camino podría darse por otro lado entonces lo ya calculado de MTU Discovery no será correcto. Es por esto que no se utiliza.

LA CAPA DE RED DE INTERNET (página 431)

Network Layer in the Internet (1)

- IP Version 4 »
- IP Addresses »
- IP Version 6 »
- Internet Control Protocols »
- Label Switching and MPLS »

Vamos a ver IPv4 y luego vamos a hablar un poco de IPv6.

Network Layer in the Internet (2)

IP has been shaped by guiding principles:

- Make sure it works
- Keep it simple
- Make clear choices
- Exploit modularity
- Expect heterogeneity
- Avoid static options and parameters
- Look for good design (not perfect)
- Strict sending, tolerant receiving
- Think about scalability
- Consider performance and cost

- Fue diseñado para que el protocolo sea sencillo.
- Asegurando que trabaje.
- Hacer que las opciones sean claras.
- Que sea modular.
- Que sea heterogéneo.
- Evitar parámetros estáticos y opciones estáticas.
- Un buen diseño.
- Estricto enviando datos y tolerante al recibirlos.
- Que tenga en cuenta la escalabilidad
- Performance y costo de comunicación.

Network Layer in the Internet (3)

Internet is an interconnected collection of many networks that is held together by the IP protocol

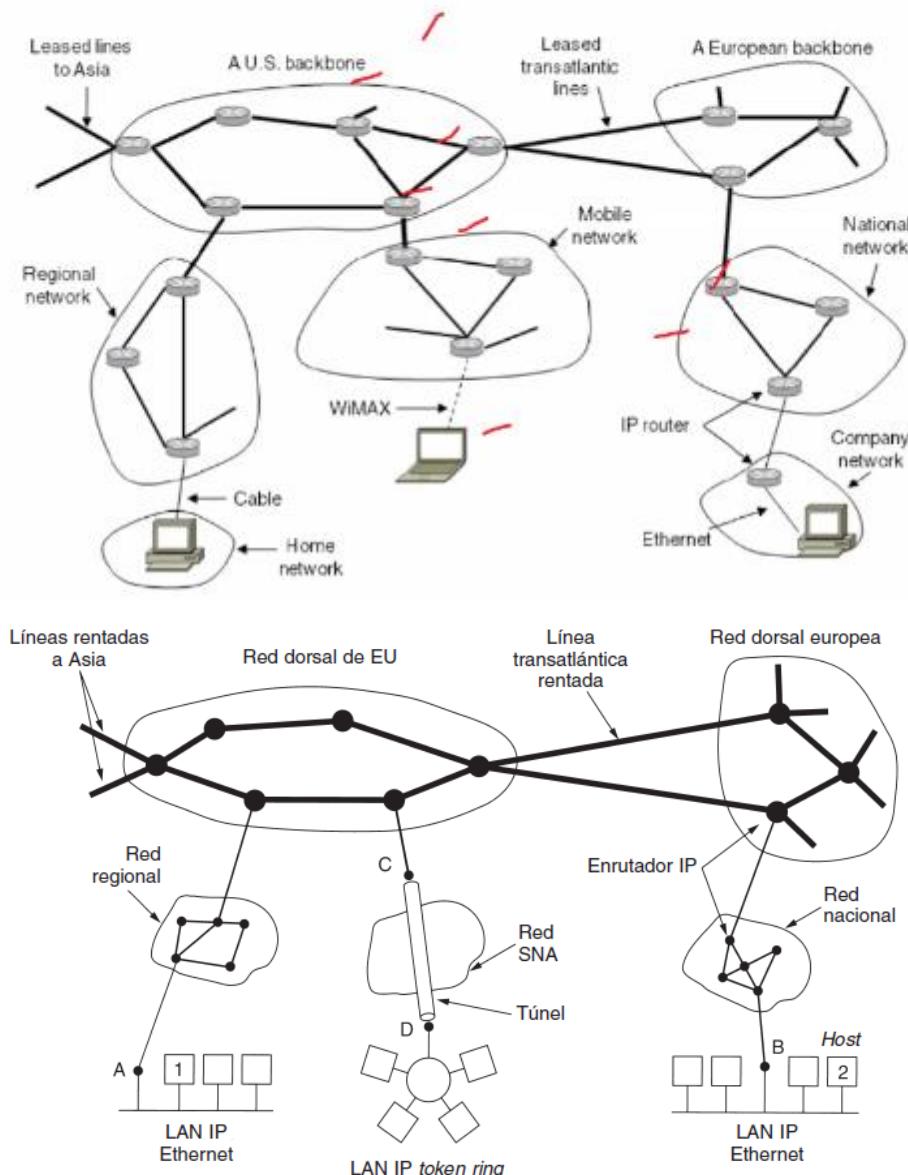


Figura 5-52. Internet es un conjunto interconectado de muchas redes.

Internet es una interconexión de distintas redes.

Hay una red principal de alta velocidad interconectada entre América del norte, Europa y Asia, luego redes regionales y finalmente los proveedores. Todas utilizando el mismo protocolo, todo necesitan tener un plan de

numeración, IP tiene un tamaño de 32 bits por lo tanto deberían tener un numero de IP único, me lo asigna quien me da el servicio de internet.

El protocolo IP (página 433) IPv4

IPv4 (Internet Protocol) header is carried on all packets and has fields for the key parts of the protocol:

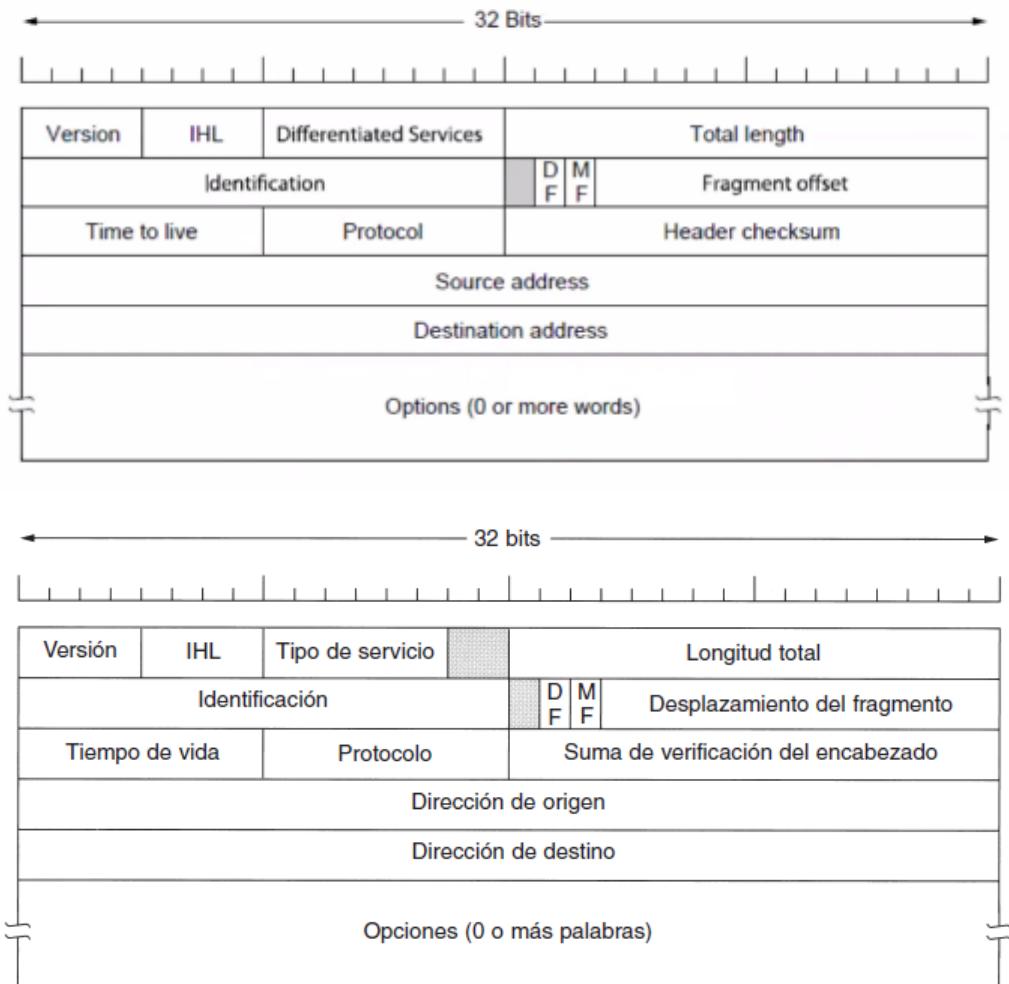


Figura 5-53. El encabezado de IPv4 (Protocolo Internet).

El campo de **Versión** lleva el registro de la versión

Dado que la longitud del encabezado no es constante, se incluye un campo en el encabezado, **IHL**, para indicar la longitud en palabras de 32 bits. El valor mínimo es de 5, cifra que se aplica cuando no hay opciones. El valor máximo de este campo de 4 bits es 15, lo que limita el encabezado a 60 Bytes y, por lo tanto, el campo de **Opciones** a 40 bytes. Para algunas opciones, por ejemplo para una que registre la ruta que ha seguido un paquete, 40 bytes es muy poco, lo que hace inútil esta opción.

El campo de **Tipo de servicio**, su propósito aún es distinguir entre las diferentes clases de servicios. Son posibles varias combinaciones de confiabilidad y velocidad. Para voz digitalizada, la entrega rápida le gana a la entrega precisa. Para la transferencia de archivos, es más importante la transmisión libre de errores que la rápida. En la práctica, los enruteadores actuales ignoran por completo el campo de Tipo de servicio, a menos que se les indique lo contrario.

La **Longitud total** incluye todo el datagrama: tanto el encabezado como los datos. La longitud máxima es de 65,535 bytes.

El campo de **Identificación** es necesario para que el host de destino determine a qué datagrama pertenece un fragmento recién llegado. Todos los fragmentos de un datagrama contienen el mismo valor de Identificación.

DF significa no fragmentar (*Don't Fragment*); es una orden para los enrutadores de que no fragmenten el datagrama, porque el destino es incapaz de juntar las piezas de nuevo.

MF significa más fragmentos. Todos los fragmentos excepto el último tienen establecido este bit, que es necesario para saber cuándo han llegado todos los fragmentos de un datagrama.

El **Desplazamiento del fragmento** indica en qué parte del datagrama actual va este fragmento. Todos los fragmentos excepto el último del datagrama deben tener un múltiplo de 8 bytes, que es la unidad de fragmentos elemental. Dado que se proporcionan 13 bits, puede haber un máximo de 8192 fragmentos por datagrama, dando una longitud máxima de datagrama de 65,536 bytes, uno más que el campo de Longitud total.

El campo de **Tiempo de vida** es un contador que sirve para limitar la vida de un paquete. Se supone que este contador cuenta el tiempo en segundos, permitiendo una vida máxima de 255 seg; debe disminuirse en cada salto y se supone que disminuye muchas veces al encolarse durante un tiempo grande en un enrutador. En la práctica, simplemente cuenta los saltos. Cuando el contador llega a cero, el paquete se descarta y se envía de regreso un paquete de aviso al host de origen. Esta característica evita que los datagramas vaguen eternamente, algo que de otra manera podría ocurrir si se llegan a corromper las tablas de enrutamiento.

Una vez que la capa de red ha ensamblado un datagrama completo, necesita saber qué hacer con él. El campo de **Protocolo** indica el protocolo de las capas superiores al que debe entregarse el paquete. TCP es una posibilidad, pero también está UDP y algunos más. La numeración de los protocolos es global en toda Internet, y se define en el RFC 1700.

La **Suma de verificación del encabezado** verifica solamente el encabezado. Tal suma de verificación es útil para la detección de errores generados por palabras de memoria erróneas en un enrutador. El algoritmo es sumar todas las medias palabras de 16 bits a medida que llegan, usando aritmética de complemento a uno, y luego obtener el complemento a uno del resultado. Para los fines de este algoritmo, se supone que la suma de verificación del encabezado es cero cuando llega el paquete al destino. Este algoritmo es más robusto que una suma normal. Observe que la suma de verificación del encabezado debe recalcularse en cada salto, pues cuando menos uno de los campos siempre cambia (el campo de Tiempo de vida), pero pueden usarse trucos para acelerar el cálculo.

La **Dirección de origen** y la **Dirección de destino** indican el número de red y el número de host.

Este es el **formato del protocolo IP** que venimos hablando. Como son muchos campos y ocupan mucho lugar está dispuesto en forma de pila.

Llega el paquete chequea:

- Versión.
- Tiempo de vida.
- Luego el tamaño del Header IHL.
- Luego Checksum.
- Genera la comparación con el tamaño del Header.
- Dirección destino para saber si es para él o no.

Si no es para él, debe reenviarlo, el protocolo siempre será IP, puede cambiar la capa de enlace. Lo que tengo que cambiar al reenviar es el tiempo de vida, lo tengo que decrementar en 1, y al cambiar esto que está en el encabezado debo cambiar el Checksum y poner el nuevo.

En esta instancia podría enviarlo o no, conozco el tamaño total y puedo tener la necesidad de tener que fragmentarlo, y en ese caso deberé ir poniendo los valores en identificación y en offset para cada uno de los fragmentos, y para cada uno de los fragmentos deberé poner el tiempo de vida y calcular el Checksum.

IP Addresses (1) – Prefixes

Adjacent addresses are allocated in blocks called prefixes

- Prefix is determined by the network portion
- Has 2^L addresses aligned on 2^L boundary
- Written address/length, e.g., 18.0.31.0/24

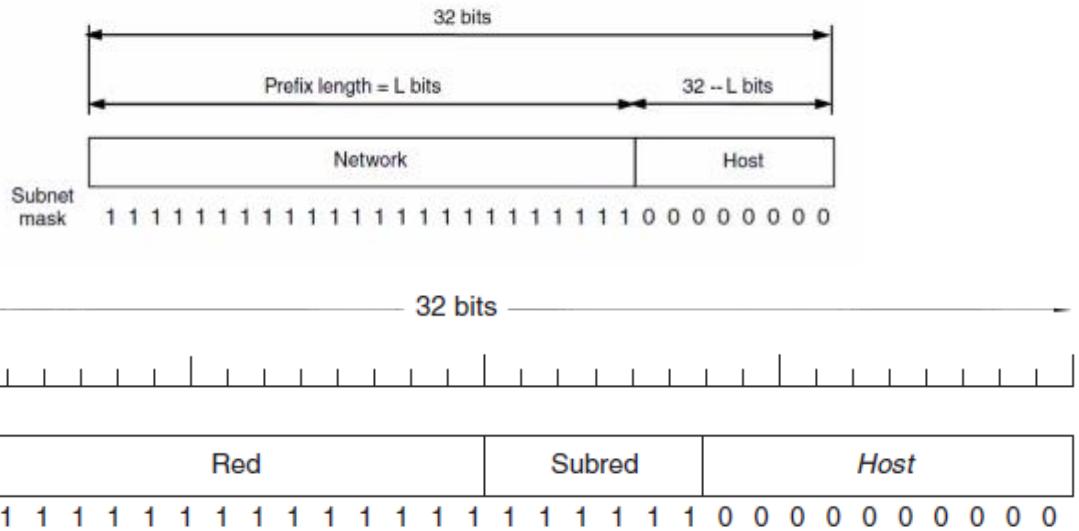


Figura 5-58. Una red de clase B dividida en 64 subredes.

Las **direcciones son de 32 bits**, pero se les ocurrió definir redes y no direcciones de 32 bits. Redes con Host dentro de las redes para cada una de las direcciones como se ve en la imagen.

Y esto es porque se dieron cuenta que es más fácil poner en las tablas de rutas direcciones de redes y no todas las direcciones posibles de Host.

Es mucho más fácil que los router tengan direcciones de redes y no una por una las direcciones de host porque si no serían tablas interminables de 2^{32} . Entonces todos los Host que estén en la misma red física le podemos dar el mismo nombre de red, entonces a los router intermedios le agrego una sola entrada en la tabla de rutas que es a esa red. Porque independientemente del host o la dirección a la que quiera ir siempre iré por el mismo camino.

De esta manera hago una sola entrada a esa red, sin importar que host de esa red sea el paquete debe ir por los mismos router y llegar a destino.

Direcciones IP

Por varias décadas, las direcciones IP se dividieron en cinco categorías, las cuales se listan en la figura 5-55. Esta asignación se ha llamado direccionamiento con clase (classful addressing). Ya no se utiliza, pero en la literatura aún es común encontrar referencias. Más adelante analizaremos el reemplazo del direccionamiento con clase.

IP Addresses (1.5) – Prefixes

- Old addresses came in blocks of fixed size (A, B, C)

Clase	Range of host addresses		
A	0	Network	Host
B	10	Network	Host
C	110	Network	Host
D	1110	Multicast address	
E	1111	Reserved for future use	

Clase	Gama de direcciones de host		
A	0	Red	Host
B	10	Red	Host
C	110	Red	Host
D	1110	Dirección multidifusión	
E	1111	Reservado para uso futuro	

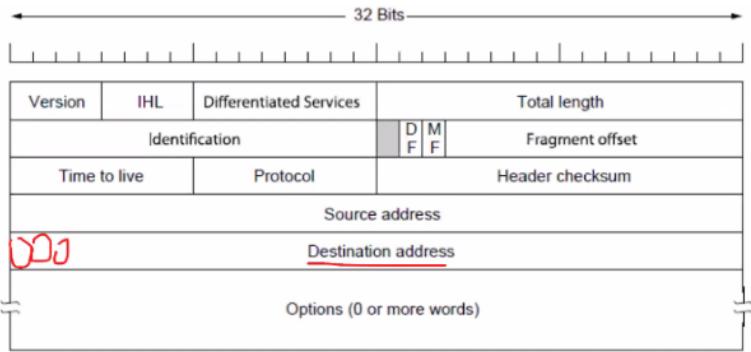
Figura 5-55. Formatos de dirección IP.

Cuando lo hicieron no pensaron en el alcance que esto iba a tener. Entonces pensaron en lo siguiente.

Si alguna dirección comienza con 0 la red va a tener los primeros 8 bits, los restantes 24 serán para los host de esa red. 2^{24} Host.

Si la dirección comienza con 10 los primeros 16 bits me van a delimitar el nombre de la red y los restantes 16 bits la cantidad de host que puedo tener en esa red.

Todos los que comienzan con 110 tendrán 24 bits de red y solo 8 bits de host. En teoría podría tener 255 host.



Como se puede ver aca no tengo ninguna indicación de red, solo tengo dirección de origen y dirección de destino, entonces los router intermedios cuando ven la dirección de destino se fijan en los primeros bits para ver qué tipo de red es, entonces en función de eso sabían con qué red interactuar en la tabla de rutas.

El problema es que a medida que paso el tiempo se quedaron sin direcciones, la granularidad en la elección de la red y la cantidad de host no era muy buena entonces terminaron por desperdiciar muchas direcciones por lo cual luego de eso se les ocurrió que además de esos 32 bits tener otros 32 bits que me digan la máscara de red.

Subredes (página 438)

La **máscara de red**, es un numero de 32 bits que tiene en unos **1** la parte de esos 32 bits que van a ser red y en **0** la parte que será host.

Entonces una red clase C tendrá 24 bits en 1 y 8 en ceros.

IP Addresses (2) – Hierarchy

Advantages:

- Routers can forward packets based only in the network portion of the IP address
- The routing tables are much smaller

Disadvantages:

- The IP address of a host depends on where it is located
- Wastefull of addresses
 - Subnets

La **ventaja** de esto es que los router reencaminan los paquetes en función de la dirección de red y no de la IP, y gracias a esto las tablas son mucho más chicas.

Como **desventaja** es que todas las redes que tengan la misma parte de red deberían estar en el mismo lugar físicamente, por lo tanto seguramente voy a desperdiciar direcciones.

IP Addresses (3) – Subnets

Subnetting splits up IP prefix to help with management

- Looks like a single prefix outside the network

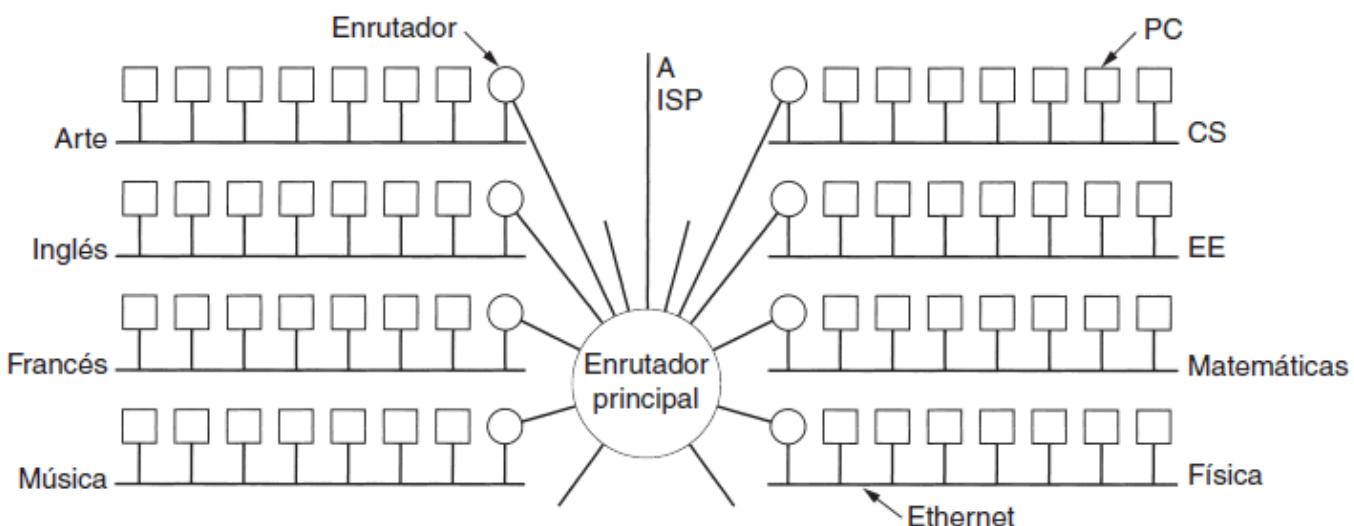
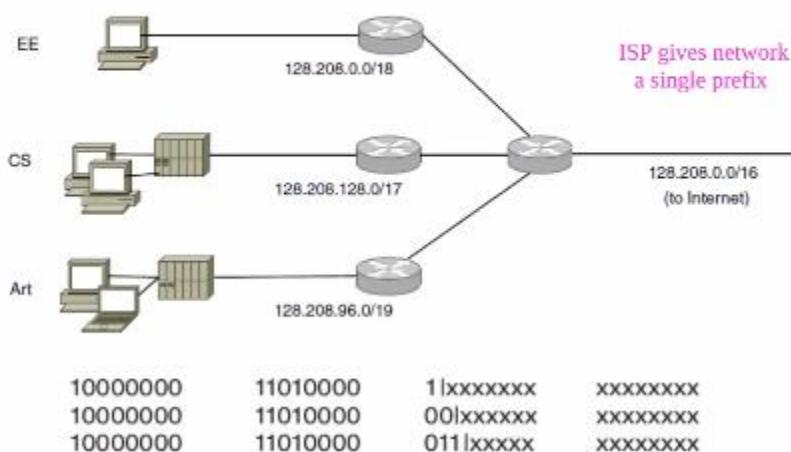


Figura 5-57. Una red de un campus que consiste de LANs para varios departamentos.

Una alternativa para no desperdiciar direcciones es utilizar lo que se llaman subredes.

Por ejemplo, yo le pido al asignador de direcciones IP, una /16, donde tendré hasta 2^{16} host, es decir que voy a tener unos 65000 posibles host, y en la oficina no los tendré.

Lo que se hace es partir esa red en subredes, obviamente todas tendrán el mismo inicio dado que hacia la derecha esta internet, y si alguien quiere contactarse con unas de las maquinas sea en la subred que sea debería poder.

El que tiene conocimiento que tengo subredes es el primer router, y dará una red para cada una de las subredes.

Entonces voy a armar una subred que en vez de tener /16 tendrá /18, que serán 2^{14} host (EE), 2^{15} host (CS) y 2^{13} host (Art).

Cualquier paquete que llegue de internet lo hará por medio de la dirección de red que son los primeros 2 bytes, 128.208 y no la subred, llegarán a ese primer router que si sabe que hay subredes, entonces dependiendo de los bits que sigan a continuación, "1", "00" o "011" sabe a qué subred va. Todas comienzan con el mismo sufijo y como condición los sufijos no se repiten.

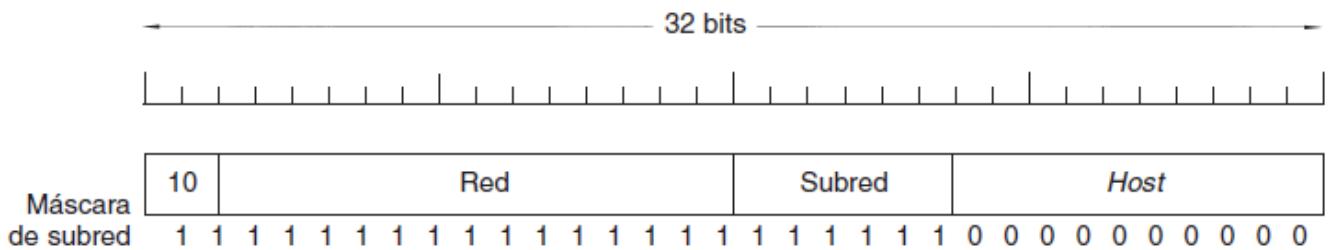


Figura 5-58. Una red de clase B dividida en 64 subredes.

Fuera de la red, la subred no es visible, por lo que la asignación de una subred nueva no requiere comunicación con el ICANN ni la modificación de bases de datos externas. En este ejemplo, la primera subred podría usar direcciones IP a partir de 130.50.4.1, la segunda podría empezar en 130.50.8.1, la tercera podría empezar en 130.50.12.1, etcétera.

Para ver por qué las subredes se cuentan en grupos de cuatro, observe que las direcciones binarias correspondientes son como se muestra a continuación:

Subred 1: 10000010 00110010 000001|00 00000001

Subred 2: 10000010 00110010 000010|00 00000001

Subred 3: 10000010 00110010 000011|00 00000001

La barra vertical (|) muestra el límite entre el número de subred y el número de host. A su izquierda se encuentra el número de subred de 6 bits; a su derecha, el número de host de 10 bits.

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

Cuando llega un paquete IP, se busca su dirección de destino en la tabla de enrutamiento. Si el paquete es para una red distante, se reenvía al siguiente enrutador de la interfaz dada en la tabla; si es para un host local (por ejemplo, en la LAN del enrutador), se envía directamente al destino. Si la red no está en la tabla, el paquete se reenvía a un enrutador predeterminado con tablas más extensas. Este algoritmo significa que cada enrutador sólo tiene que llevar el registro de otras redes y hosts locales (no de pares red-host), reduciendo en gran medida el tamaño de la tabla de enrutamiento.

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

Por ejemplo, a un paquete dirigido a 130.50.15.6 que llega a un enrutador de la subred 5 se le aplica un AND con la máscara de subred 255.255.252.0/22 para dar la dirección 130.50.12.0. Esta dirección se busca en las tablas de enrutamiento para averiguar la manera de llegar a los hosts de la subred 3. Por lo tanto, la división de redes reduce espacio en la tabla de enrutamiento creando una jerarquía de tres niveles, que consiste en red, subred y host.

CIDR—Enrutamiento interdominios sin clases (classless)

El IP ya ha estado en uso intensivo por más de una década; ha funcionado extremadamente bien, como lo demuestra el crecimiento exponencial de la Internet. Desgraciadamente, el IP se está convirtiendo con rapidez en

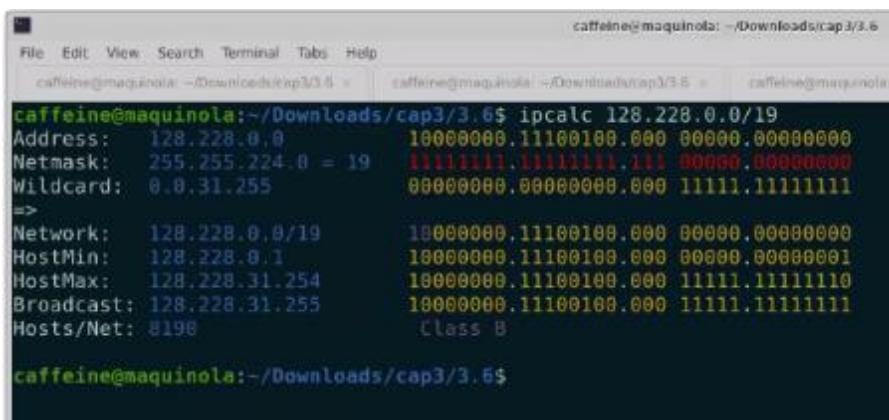
víctima de su propia popularidad: se le están acabando las direcciones. Este desastre inminente ha propiciado una gran cantidad de controversias y debates en la comunidad de Internet sobre lo que debe hacerse al respecto.

En resumen, la mayoría de las soluciones resuelven un problema pero crean uno nuevo. La solución que se implementó y que dio a Internet un respiro es el CIDR (Enrutamiento Interdominios sin Clases). El concepto básico del CIDR, que se describe en el RFC 1519, es asignar las direcciones IP restantes en bloques de tamaño variable, independientemente de las clases. Si un sitio necesita, digamos, 2000 direcciones, se le da un bloque de 2048 direcciones con un límite de 2048 bytes.

Eliminar las clases hace más complicado el reenvío. En el sistema antiguo con clases el reenvío se hacía de la siguiente manera: cuando un paquete llegaba a un enrutador, una copia de la dirección IP se desplazaba 28 bits a la derecha para obtener un número de clase de 4 bits. Entonces una rama de 16 vías ordenaba los paquetes en A, B, C y D (si lo soportaba), con ocho de los casos para la clase A, cuatro de los casos para la clase B, dos de los casos para la clase C, uno para D y otro para E. A continuación, el código para cada clase enmascaraba los números de red de 8, 16 o 24 bits y los alineaba a la derecha en una palabra de 32 bits. Entonces se buscaba el número de la red en la tabla A, B o C, por lo común mediante indexación en las redes A y B y aplicando hash en las redes C. Una vez que se encontrara la entrada, se podría buscar la línea de salida y remitir el paquete.

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

Una herramienta que tienen prácticamente todos los sistemas operativos es ipcalc y al agregar el nombre de red y la máscara nos muestra cual es el nombre de la red, cual es el primer host y el último host, cantidad de host que tiene la red.



```
caffeine@maquinola:~/Downloads/cap3/3.6$ ipcalc 128.228.0.0/19
Address: 128.228.0.0      10000000.11100100.0000.00000000
Netmask: 255.255.224.0 = 19 1111111.11111111.1111.00000000
Wildcard: 0.0.31.255       00000000.00000000.0000.11111111
=>
Network: 128.228.0.0/19   10000000.11100100.0000.00000000
HostMin: 128.228.0.1      10000001.11100100.0000.00000001
HostMax: 128.228.31.254   10000000.11100100.0000.11111110
Broadcast: 128.228.31.255 10000000.11100100.0000.11111111
Hosts/Net: 8190            Class B
```

Podemos ver que nos da los datos de la red, me dice que el host mínimo es 128.228.0.1 y el último en 128.228.31.254, ¿este último está en la misma red? Está en la misma red, porque la máscara llega hasta el 3er bit del 3er byte, y se puede ver que los bits que están a la izquierda son iguales al del host inicial y al de red.

```

caffeine@maquinola:~/Downloads/cap3/3.6$ ipcalc 10.228.0.0/23
Address: 10.228.0.0      00001010.11100100.00000000 0.00000000
Netmask: 255.255.254.0 = 23 1111111.1111111.1111111 0.00000000
Wildcard: 0.0.1.255       00000000.00000000.00000000 1.1111111
=>
Network: 10.228.0.0/23    00001010.11100100.00000000 0.00000000
HostMin: 10.228.0.1       00001010.11100100.00000000 0.00000001
HostMax: 10.228.1.254     00001010.11100100.00000000 1.1111110
Broadcast: 10.228.1.255   00001010.11100100.00000000 1.1111111
Hosts/Net: 510            Class A, Private Internet

```

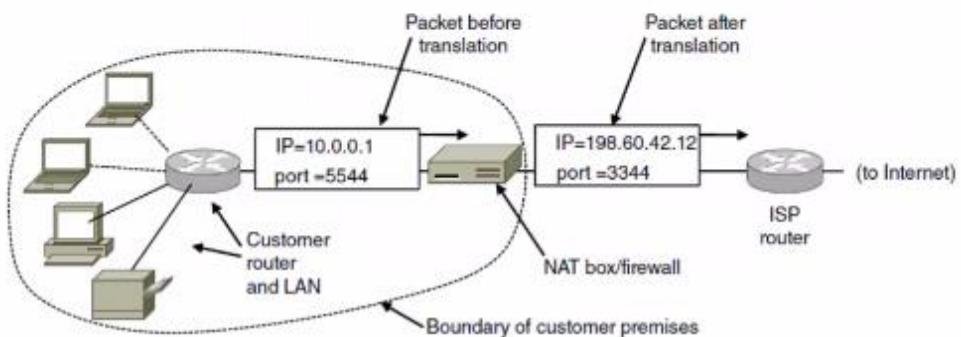
Aca podemos ver otro ejemplo y un detalle, dice clase A, internet privado. Hay algunos rangos de red que son privados, ningun router de internet tienen ruta a esos host, si alguien quiere enviar un paquete a ese host no van a llegar. 10.228.0.0 es una, 172.16.0.0 tambien, y 192.168.1.0 tambien es privada.

NAT—Traducción de Dirección de Red (página 444)

IP Addresses (6) – NAT

NAT (Network Address Translation) box maps one external IP address to many internal IP addresses

- Uses TCP/UDP port to tell connections apart
- Violates layering; very common in homes, etc.



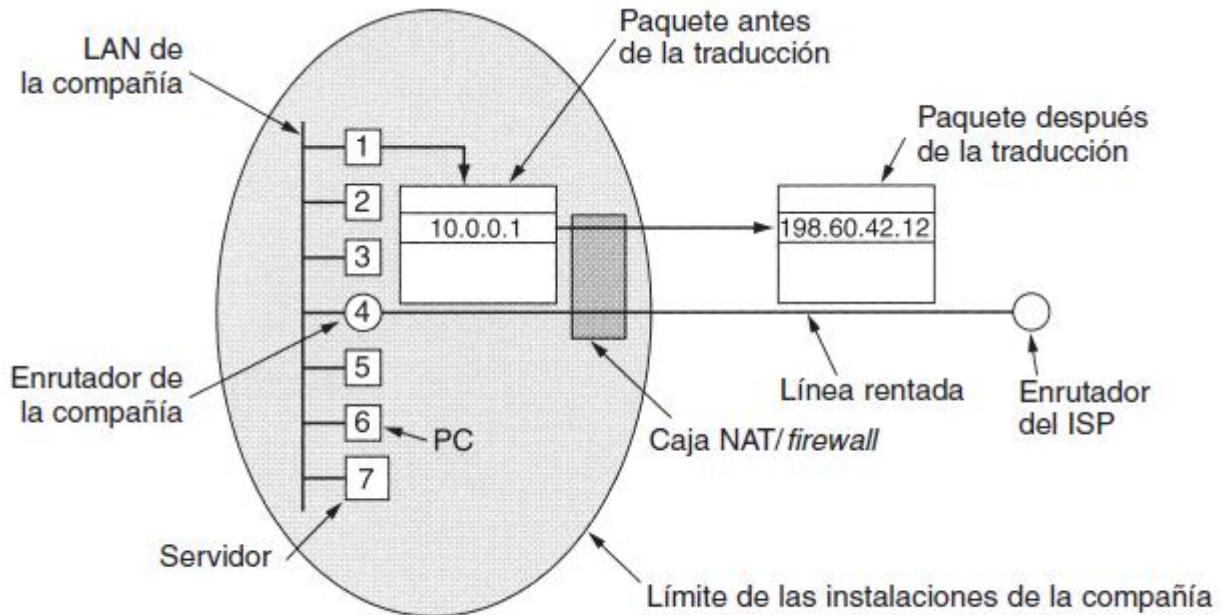


Figura 5-60. Colocación y funcionamiento de una caja NAT.

Hasta ahora habíamos visto que el paquete IP no se modificaba al pasar de una capa de red a otra, solo se modificaba el encabezado, tiempo de vida, Checksum.

En el escenario que vemos en la imagen llamado **NAT, si se modifica**, y esto porque ya no hay más direcciones IP disponibles, IPv4 no hay más. Entonces **cuando nosotros queremos conectar la red a internet no nos van a dar dirección pública.**

Tenemos un par de alternativas, en nuestra red **poner un rango de direcciones IP privadas que no se rutean a internet**, podría ser la **192.168.0.0/16**. Y el **router** que lo conecta a internet tiene una interfaz hacia internet y una interfaz hacia su red **LAN**, y este **router** además de **encaminar**, cada vez que un paquete va a internet **modifica la dirección origen y le pone la dirección origen de la IP pública**, de manera que **al llegar el paquete a destino, luego la respuesta vuelve a la IP pública del router origen, y el router sabe cuál es la IP original, las intercambia nuevamente a la privada y le responde al host**. Esto viola el funcionamiento original de IP pero esto permite que una red generalmente con direcciones privadas pueda conectarse a través de Gateway o un router con NAT a internet. Esto es lo que hacen todos los router hogareños.

El encabezado principal del IPv6 (página 466)

IP Version 6 (1)

Major upgrade in the 1990s due to impending address exhaustion, with various other goals:

- Support billions of hosts
- Reduce routing table size
- Simplify protocol
- Better security
- Attention to type of service
- Aid multicasting
- Roaming host without changing address
- Allow future protocol evolution
- Permit coexistence of old, new protocols, ...

Deployment has been slow & painful, but may pick up pace now that addresses are all but exhausted

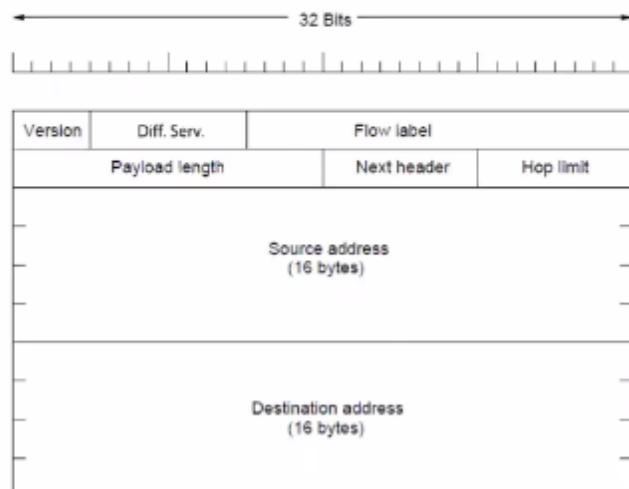
Otra alternativa mejor es utilizar IPv6, es una versión más nueva del protocolo IP de red que venimos viendo, en 1992 salieron los primeros estándares de este protocolo.

Agranda un poco el campo de direcciones, un nuevo protocolo que no es compatible con IPv4. Tomaron la 2da aproximación en protocolo deductivo, la única manera de que convivan es que la máquina que los usa tenga los dos protocolos implementados en la pila entonces si le llega algo en IPv4 lo manda a la pila IPv4 y si no al IPv6, no hay otra manera de que convivan.

Lo hicieron más sencillo para que los router intermedios tengan que trabajar tanto.

IP Version 6 (2)

IPv6 protocol header has much longer addresses (128 vs. 32 bits) and is simpler (by using extension headers)



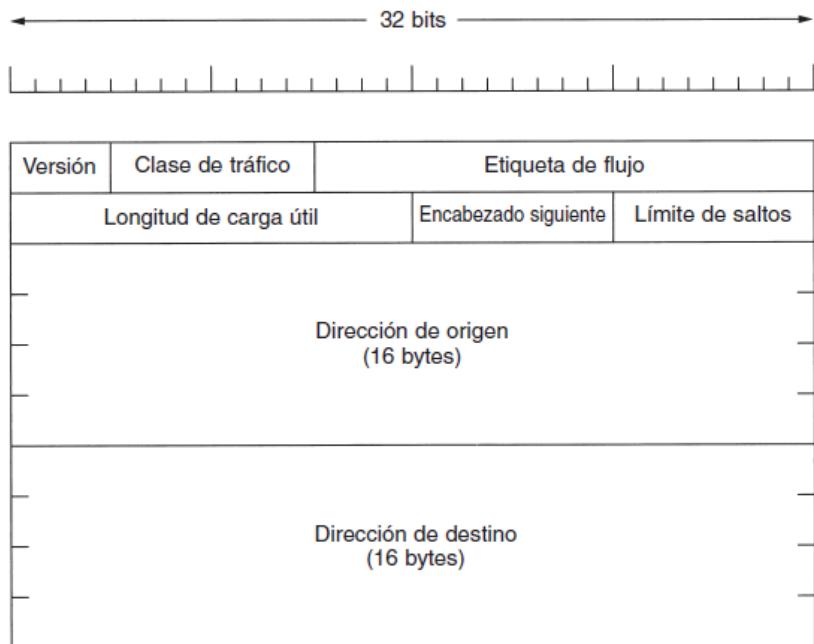


Figura 5-68. Encabezado fijo del IPv6 (obligatorio).

El campo **Clase de tráfico** se usa para distinguir entre los paquetes con requisitos diferentes de entrega en tiempo real.

El campo de **Etiqueta de flujo** aún es experimental, pero se usará para permitir a un origen y a un destino establecer una pseudoconexión con propiedades y requisitos particulares. Por ejemplo, una cadena de paquetes de un proceso en cierto host de origen dirigido a cierto proceso en cierto host de destino puede tener requisitos de retardos muy estrictos y, por tanto, necesitar un ancho de banda reservado.

El campo de **Longitud de carga útil** indica cuántos bytes siguen al encabezado de 40 bytes.

El campo **Encabezado siguiente** revela el secreto. La razón por la que pudo simplificarse el encabezado es que puede haber encabezados adicionales (opcionales) de extensión. Este campo indica cuál de los seis encabezados de extensión (actualmente), de haberlos, sigue a éste. Si este encabezado es el último encabezado de IP, el campo de **Encabezado siguiente** indica el manejador de protocolo de transporte (por ejemplo, TCP, UDP) al que se entregará el paquete.

El campo de **Límite de saltos** se usa para evitar que los paquetes vivan eternamente. En la práctica es igual al campo de Tiempo de vida del IPv4, es decir, un campo que se disminuye en cada salto.

Luego vienen los campos de **Dirección de origen** y **Dirección de destino**.

Si bien es más grande, tiene más espacio que el otro tiene varios campos menos y es por esto que es más sencillo.

Lo primero que esta puesto es la versión, en IPv4 era 0100, en IPv6 es 0110, para saber a qué protocolo de la capa de red se lo doy, si es al STACK IPv4 o IPv6.

Luego abajo se ven las direcciones que en el caso de IPv4 eran de 32 bits, en este caso para IPv6 es de 128 bits cada dirección. (16 bytes por cada renglón). De haber puesto 33 bits habría direccionado el doble, con un bit más lograría eso, aca son 128, es muchísimo más lo que se puede direccionar.

Aca no está lo de tiempo de vida pero si esta como cantidad de saltos (HOP LIMIT), si aca le pongo 10 se va decrementando cada vez que pasa por un router, si llego a 10 se descarta.

No está el Checksum, cuando llega algo no calcula el Checksum, incremento la cantidad de saltos pero no vuelvo a calcular el Checksum, entonces me estoy ahorrando dos cálculos de Checksum. Además tampoco tengo el dato del largo del encabezado lo cual era necesario para el Checksum. Aca el encabezado es fijo, no tengo que estar haciendo cuentas. No tiene ningún código de detección de errores. Hay una tasa tan pequeña de error que vale la pena sacar este procesamiento.

Aca no puedo fragmentar, hay un MTU, tamaño mínimo de Payload que cualquier capa de enlace debería soportar para IPv6, que es 1240, si es menos que eso la capa de enlace no soporta IPv6 y ahí termina.

Tiene NEXT HEADER que me dice cuál es el protocolo de capa de transporte que voy a usar.

DIFF. SERV. Puedo hacer diferencia de servicio, FLOW LABEL etiqueta de flujo viene siendo al parecido a circuitos virtuales pero IP es sin conexión así que no puedo tener circuitos virtuales.

Todos estos cambios intentan que el router trabaje menos cada vez que recibe un paquete.

Encabezados de extensión (página 469)

IP Version 6 (3)

IPv6 extension headers handles other functionality

Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

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

Figura 5-69. Encabezados de extensión del IPv6.

Estas son algunas opciones que se podrían poner como un encabezado si hubiera, si no lo hay un sin encabezado es un protocolo de capa de transporte de la siguiente capa

REDES – CAPA DE TRANSPORTE (página 481)

Chapter 6

The Transport Layer

The Transport Service

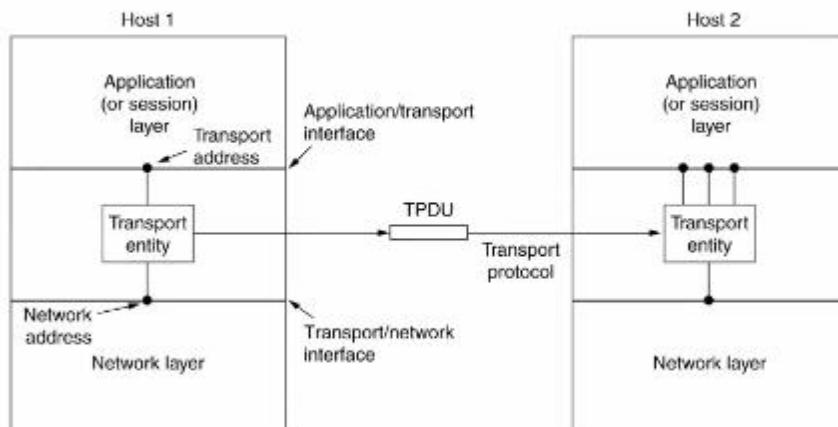
- Services Provided to the Upper Layers
- Transport Service Primitives
- Berkeley Sockets
- An Example of Socket Programming:
 - An Internet File Server
- User Datagram Protocol

Los grises los veremos en otra clase que se refieren a como desde una aplicación puedo utilizar la red. Vamos a ver qué servicios da la capa de transporte y un ejemplo de capa de transporte.

Básicamente en **capa de transporte** hay dos protocolos **USER DATAGRAM PROTOCOL “UDP”** **TRANSPORT TRANSPORT PROTOCOL “TCP”**. La diferencia entre ellos es que uno es orientado a conexión y el otro es sin conexión.

Servicios proporcionados a las capas superiores

Services Provided to the Upper Layers



The network, transport, and application layers.

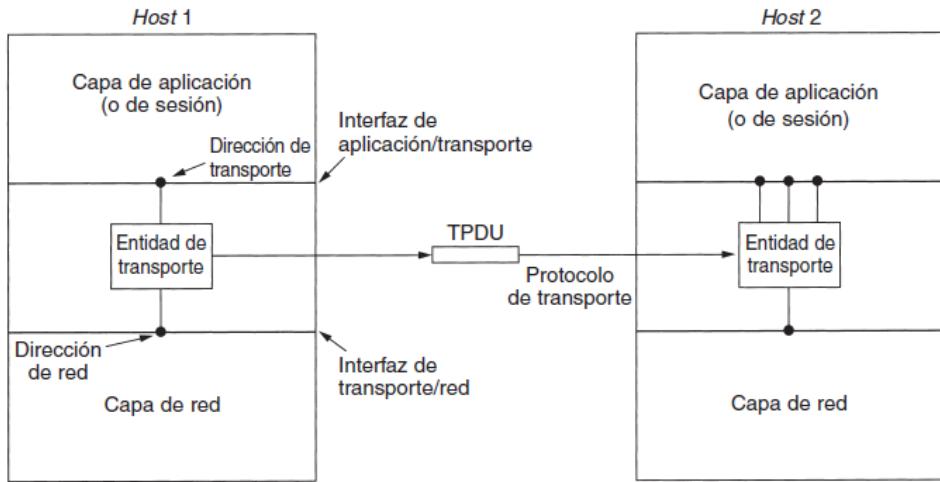


Figura 6-1. Las capas de red, transporte y aplicación.

Aca tenemos una representación de las distintas capas, no están todas, está la capa de red, la de transporte y la de aplicación.

En la **capa de red** hablamos que la unidad de transporte de capa de red eran los datagramas o los paquetes, y en la **capa de transporte** la unidad es TPDU: Unidad de Datos del Protocolo de Transporte, y se le llama segmento.

Cuando hablemos de paquetes o datagramas hablamos de la capa de red, y cuando hablemos de segmentos estamos hablando de la capa de transporte.

Algo importante son los **puntos de entrada entre la capa de transporte y la capa de red**, sería la dirección de red. La **capa de transporte** utiliza la dirección de red para decir de donde va, hacia donde va. Y la **aplicación** utiliza una dirección de transporte para decirle de que proceso a que proceso va.

En **capa de red** estoy interconectando maquinas, el objetivo de la placa de red es que un paquete pueda atravesar distintas capas de enlace para llegar desde un Host origen a un Host destino. Como podrían ser distintas tecnologías de red, **en capa de red lo que hice fue abstraerme de las capas de enlace** y puse un plan de numeración homogéneo a todas las maquinas que estén conectadas en las distintas topologías de red o capas de enlace.

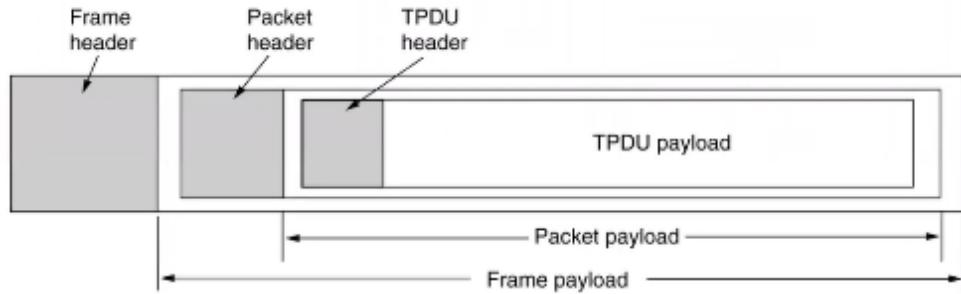
Entonces la capa de red lograba con suerte que un paquete llegue de una maquina a otra sin importar donde estuviera siempre que estén interconectadas con algún tipo de tecnología de capa de enlace.

El **problema** es que esto es un IPC, yo no tengo que interconectar maquinas, tengo que interconectar procesos.

Por ejemplo tengo 50 procesos corriendo en mi máquina y me llega en la capa de red un paquete que viene para mi IP enviado por el Host 1, ¿los datos a que proceso se los mando? La **capa de transporte** se encarga que la comunicación en vez de ser de Host a Host sea de Proceso de un Host a Proceso de otro Host. Pone una capa de fracción más y pone unas direcciones de transporte que las asocia a los procesos. Se crean unas nuevas direcciones que no son de red, no son de capa de enlace, son de transporte y vincula distintos procesos. Se podría decir que la **capa de transporte multiplexa el canal subyacente que sería la capa de red entre los distintos procesos**.

Estas direcciones que asigna a los procesos se llaman **Puertos**, los puertos son direcciones de la capa de transporte donde puedo asociar aplicaciones a un puerto.

Transport Service Primitives (2)



The nesting of TPDUs, packets, and frames.

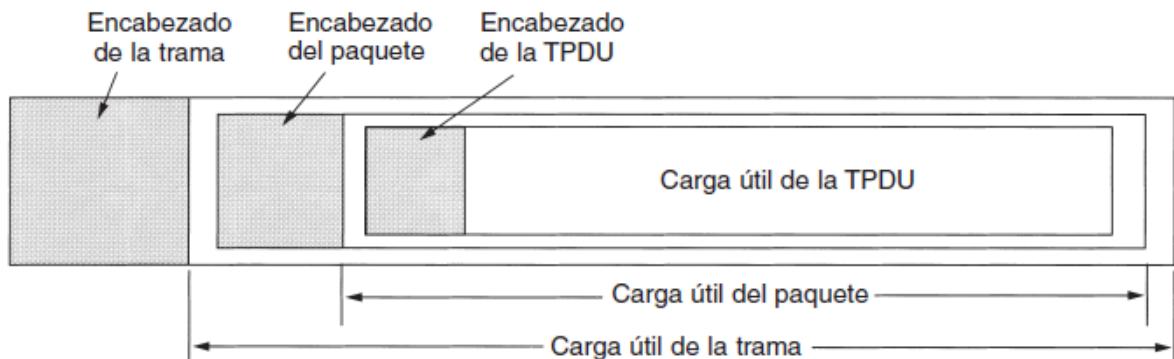


Figura 6-3. Anidamiento de las TPDUs, los paquetes y las tramas.

Podemos ver la encapsulación o el anidado de los distintos protocolos. Esta es una trama que viaja por un cable o por donde sea. Lo primero que se ve es el **encabezado de la trama** que podría ser Ethernet o cualquier de los que vimos en capa de enlace. Luego en el **Payload** voy a tener un paquete IP que tiene un encabezado, luego en el payload vamos a ver un segmento de la capa de transporte que también tendrá un encabezado y luego tendremos los datos.

Estos datos seguro que será un protocolo de capa de aplicación donde también tendrá un encabezado.

ELEMENTOS DE LOS PROTOCOLOS DE TRANSPORTE

Elements of Transport Protocols

- Addressing
- Connection Establishment
- Connection Release
- Flow Control and Buffering
- Multiplexing
- Crash Recovery

Entonces el protocolo de transporte tiene un esquema de direcciones distinto de la capa de red, si es con conexión puede tener mecanismos para establecer la comunicación y para liberarla, podría tener control de flujo y buffer, la multiplexación que fue lo que hablamos recién donde tomo la capa de red y la comparto con varios procesos, y para las que son con conexión también hay recuperación ante caídas.

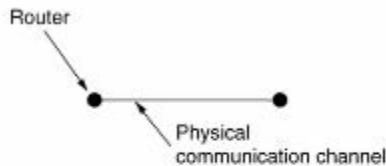
Transport Protocol

Transport Protocols resemble data link protocols:

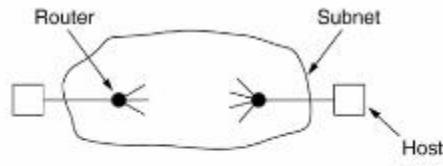
Error control

Sequencing

Flow Control



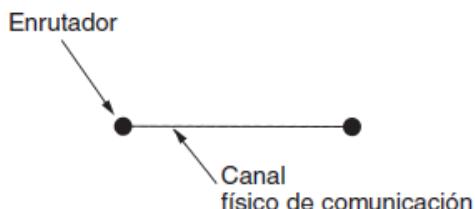
(a)



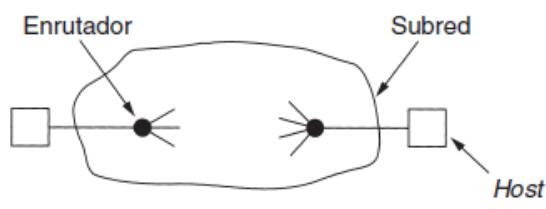
(b)

(a) Environment of the data link layer.

(b) Environment of the transport layer.



(a)



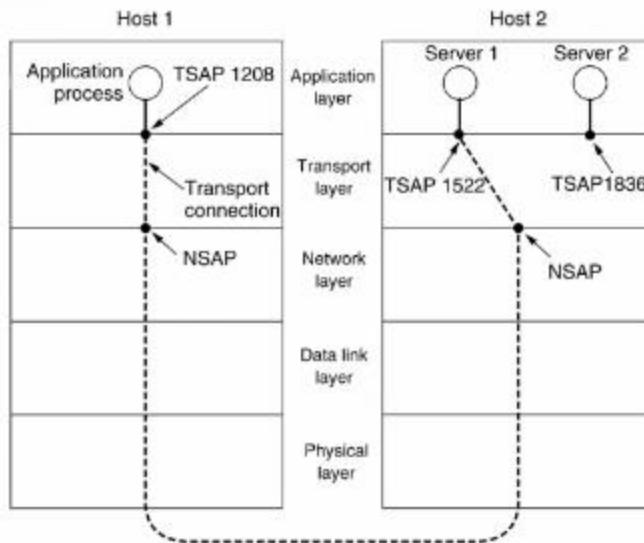
(b)

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

Si comparo entre **capa de enlace** y **capa de transporte** es más o menos lo mismo, (a) podía tener un host conectado y en el mismo medio otro host, entonces tenía manejo de error, de secuencia y control de flujo. Pero si vemos ahora la **capa de transporte** (b), podríamos asumir que toda la parte de red de internet fuese como una capa de enlace, entonces cada host está conectado a una capa de enlace, y es por esto que la capa de transporte se comporta como la capa de enlace. Estoy utilizando un **protocolo común** debajo de la capa de transporte que es el **protocolo IP**, si bien es una abstracción porque no es una sola capa de enlace si no que son muchas, pero de alguna manera la capa de red homogeniza y lo podría ver desde capa de transporte como si fuera una sola capa de enlace, y por eso es que volvemos a tratar los mismos temas de extremo a extremo. (*Un poco confuso este párrafo*).

Direccionamiento (página 493)

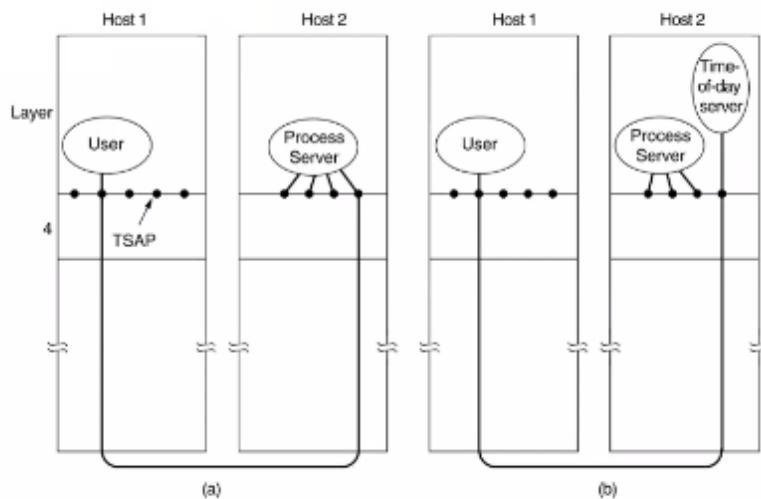
Addressing



TSAPs, NSAPs and transport connections.

Antes en la capa de red teníamos los NSAP que es el punto de acceso al servicio de red con las direcciones, y ahora en la **capa de transporte** me voy a encontrar con **TSAP (TRANSFER SERVICE ACCESS POINT**, Punto de acceso al servicio de transporte) que son números, estos **son los puertos**. Tengo un proceso y lo puedo asociar al TSAP 1208, puerto 1208, que tiene la IP NSAP, voy usando pares.

Connection Establishment



How a user process in host 1 establishes a connection with a time-of-day server in host 2 (xinetd).

XINETD no se usa más, no hay que verlo. Básicamente era usado para borrar procesos en memoria, entonces tenía un solo proceso que escuchaba varios TSAP y una vez que alguien se conectaba recién ahí creaba un nuevo proceso que sería el que se ve en (b) que atendía al usuario. Era para ahorrar procesos. Hoy no es necesario porque en una máquina pueden correr miles de procesos y no pasa nada. Era para cuando las máquinas tenían pocos recursos.
IGNORAR ESTO.

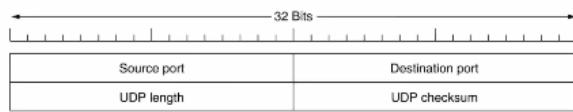
The Internet Transport Protocols: UDP

- Introduction to UDP
- Remote Procedure Call
- The Real-Time Transport Protocol

Vamos a ver un protocolo sin conexión, UDP, este protocolo no establece ninguna conexión antes, tengo un dato y lo mando directamente, ¿Qué puede pasar? Si al proceso al que le estoy enviando no existe porque se terminó o porque no se está ejecutando se pierden esos datos, pero nadie me avisa que se perdieron o que llegaron bien, a no ser que el proceso al que le estoy enviando los datos me conteste. Eso depende de cómo haya escrito el programa si contesta o no.

LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: UDP

Introduction to UDP



The UDP header.

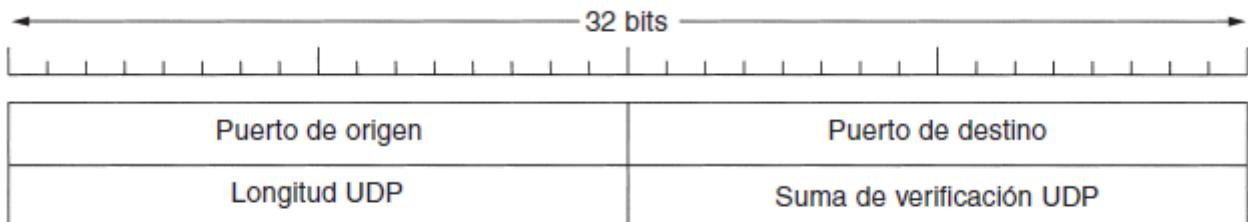


Figura 6-23. El encabezado UDP.

Esto va en el Payload de la capa IP, protocolo IP, tiene solo estos 4 campos.

Tiene la dirección de la capa de transporte origen que sería el puerto origen, y otros 16 bits para la dirección del puerto de destino que es la aplicación destino donde va a ir para asociarlo.

El largo del datagrama UDP y un campo de Checksum del encabezado UDP pero es opcional. Podría estar en cero, y el proceso que lo recibe no lo descarta.

Este protocolo solo me está poniendo el puerto origen y el puerto destino y nada más, es lo que necesito para poder multiplexar entre varios procesos la capa de red.

Se usa bastante, supongamos que estamos transmitiendo cosas en tiempo real, video o audio, no tiene sentido retransmitir más tarde si no llega una parte de la información, porque si me llega después solo metería ruido. No intenta retransmitir, por lo tanto es más rápido y no necesito establecer una conexión antes.

También depende de las aplicaciones, hay aplicaciones donde intercambio muy poca cantidad de datos, imaginen lo que sería tener que establecer una conexión primero, intercambiar datos y luego cortar la conexión, al final esto

me ocupa más lugar y tiempo que lo que demoran los datos que quiero intercambiar, entonces al final es más conveniente hacerlo con un protocolo de capa de transporte sin conexión para ahorrarme tiempo.

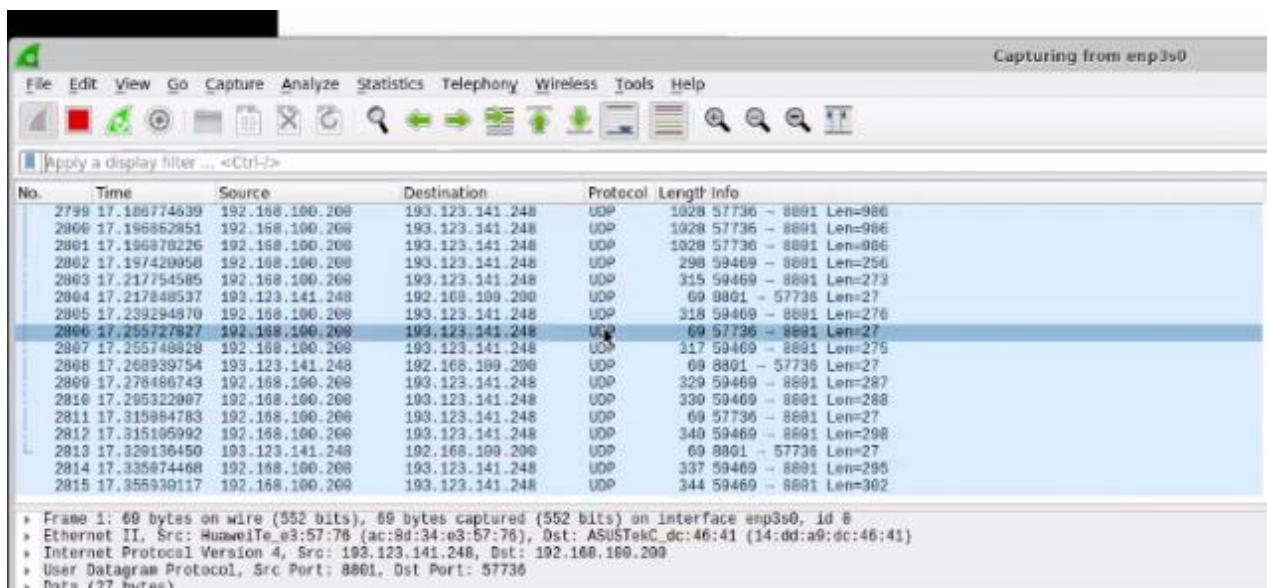
El **protocolo DNS**, una IP son 4 números separados por puntos, recordar los números es complicado, entonces sería mucho más fácil que fueran nombres en vez de números, entonces básicamente lo que hace DNS es que al preguntar por un nombre me dice que dirección IP le corresponde. Esto es de capa aplicación. Pero internamente en capa de transporte usa UDP, porque las consultas son muy sencillas, entonces no tiene sentido perder tiempo estableciendo una conexión y luego cortando una conexión.

Host es un cliente DNS que hace una consulta de DNS que usa UDP. Entonces puedo hacer la consulta:

```
caffeine@maquinola: ~/Downloads/ca
File Edit View Search Terminal Tabs Help
caffeine@m... caffeine@m... caffeine@m... caffeine@m...
caffeine@maquinola:~/Downloads/cap3/3.6$ host www.frm.utn.edu.ar
www.frm.utn.edu.ar is an alias for web.frm.utn.edu.ar.
web.frm.utn.edu.ar has address 200.10.196.21
caffeine@maquinola:~/Downloads/cap3/3.6$
```

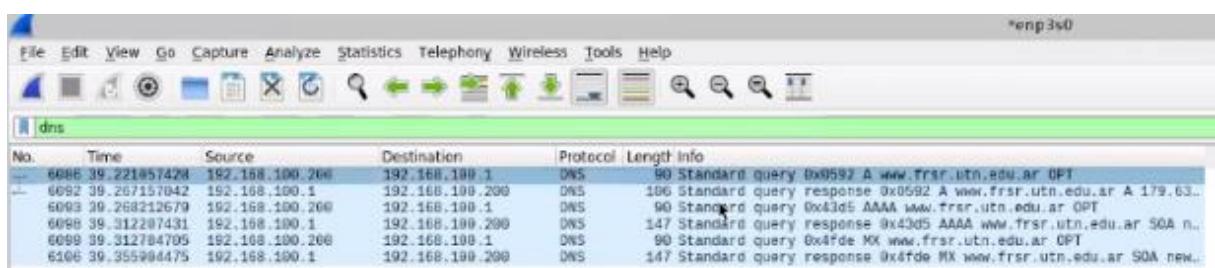
Software muy útil WIRESHARK

Puedo capturar mi placa de red Ethernet, cuando lo hago puedo ver que captura mucho tráfico UDP, se puede ver que es de ZOOM, Zoom es video en tiempo real y usa UDP



La ventana de arriba me da un resumen del tráfico, me dice de que IP a que IP, el protocolo. La ventana del medio me las va separando por capa a la información y en la 3ra ventana inferior me muestra los Bytes que viajan por el cable.

Podemos filtrar todo el tráfico DNS:



Se pueden ver los pocos mensajes de la consulta.

En la 2da ventana me muestra como está compuesto.

```

> Frame 6086: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface enp3s0, id 0
  > Ethernet II, Src: ASUSTekC_dc:46:41 (14:dd:a9:dc:46:41), Dst: HuaweiTe_e3:57:76 (ac:8d:34:e3:57:76)
    > Destination: HuaweiTe_e3:57:76 (ac:8d:34:e3:57:76)
    > Source: ASUSTekC_dc:46:41 (14:dd:a9:dc:46:41)
    > Type: IPv4 (0x0800)
  > Internet Protocol Version 4, Src: 192.168.100.200, Dst: 192.168.100.1
  > User Datagram Protocol, Src Port: 40700, Dst Port: 53
  > Domain Name System (query)

```

En la parte de Ethernet puedo ver que primero está la IP destino, luego se puede ver la IP origen, también se puede ver que muestra una marca, ASUSTekC y HuaweiTE, y esto es porque a los fabricantes se le asigna la mitad más alta de los 48 bits (24bits), y luego cada fabricante va asignando a cada placa que fabrica un numero aleatorio cualquiera.

También me dice que el Payload de Ethernet es IPv4.

```

> Frame 6086: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface enp3s0, id 0
  > Ethernet II, Src: ASUSTekC_dc:46:41 (14:dd:a9:dc:46:41), Dst: HuaweiTe_e3:57:76 (ac:8d:34:e3:57:76)
    > Destination: HuaweiTe_e3:57:76 (ac:8d:34:e3:57:76)
    > Source: ASUSTekC_dc:46:41 (14:dd:a9:dc:46:41)
    > Type: IPv4 (0x0800)
  > Internet Protocol Version 4, Src: 192.168.100.200, Dst: 192.168.100.1
    0100 .... = Version: 4
    ... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 76
    Identification: 0x539e (21406)
  > Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x9ce8 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.100.200
    Destination: 192.168.100.1
  > User Datagram Protocol, Src Port: 40700, Dst Port: 53
  > Domain Name System (query)

```

Si vemos en el Payload vemos el encabezado IP, vemos que es protocolo IPv4, está habilitado no fragmentar, no tiene offset, el tiempo de vida es de 64, dice que sobre IP el Payload que va es UDP.

```

> Frame 6086: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface enp3s0, id 0
  > Ethernet II, Src: ASUSTekC_dc:46:41 (14:dd:a9:dc:46:41), Dst: HuaweiTe_e3:57:76 (ac:8d:34:e3:57:76)
  > Internet Protocol Version 4, Src: 192.168.100.200, Dst: 192.168.100.1
  > User Datagram Protocol, Src Port: 40700, Dst Port: 53
    Source Port: 40700
    Destination Port: 53
    Length: 56
    Checksum: 0x4a64 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 11]
    * [Timestamps]
  > Domain Name System (query)

```

Podemos ver el protocolo UDP, donde se puede ver el puerto origen, puerto destino 53, todos los servidores DNS están esperando que alguien les consulte en el puerto 53, por lo tanto esa aplicación host que corrimos para consultar ya sabía que tiene que ir por el puerto 53. El largo son 56 bytes, también se ve el Checksum.

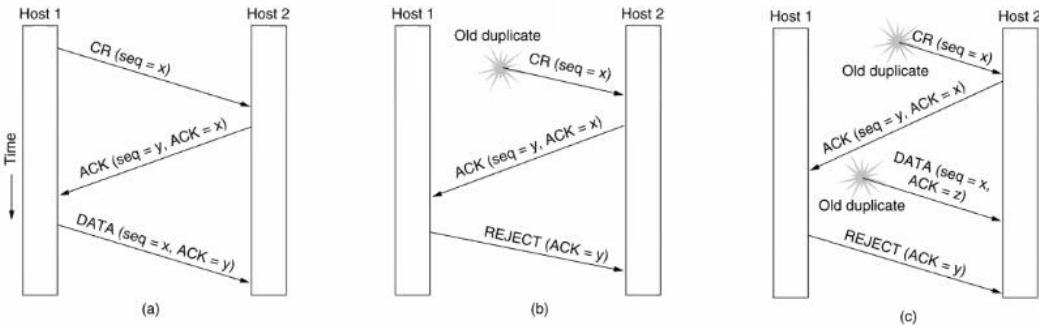
```

> Frame 6086: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface enp3s0, id 0
  > Ethernet II, Src: ASUSTekC_dc:46:41 (14:dd:a9:dc:46:41), Dst: HuaweiTe_e3:57:76 (ac:8d:34:e3:57:76)
  > Internet Protocol Version 4, Src: 192.168.100.200, Dst: 192.168.100.1
  > User Datagram Protocol, Src Port: 40700, Dst Port: 53
    Source Port: 40700
    Destination Port: 53
    Length: 56
    Checksum: 0x4a64 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 11]
    * [Timestamps]
  > Domain Name System (query)
    Transaction ID: 0x0592
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
    Queries
      > www.frsr.utn.edu.ar: type A, class IN
    > Additional records
    [Response In: 6002]

```

Luego abajo vemos el Payload de la capa de aplicación donde se puede ver la información de la consulta que en este caso era saber que IP tiene la www.frsr.utn.edu.ar .

Connection Establishment (3)



Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST.

- (a) Normal operation,
- (b) Old Dup CONNECTION REQUEST appearing out of nowhere.
- (c) Duplicate CONNECTION REQUEST and duplicate ACK.

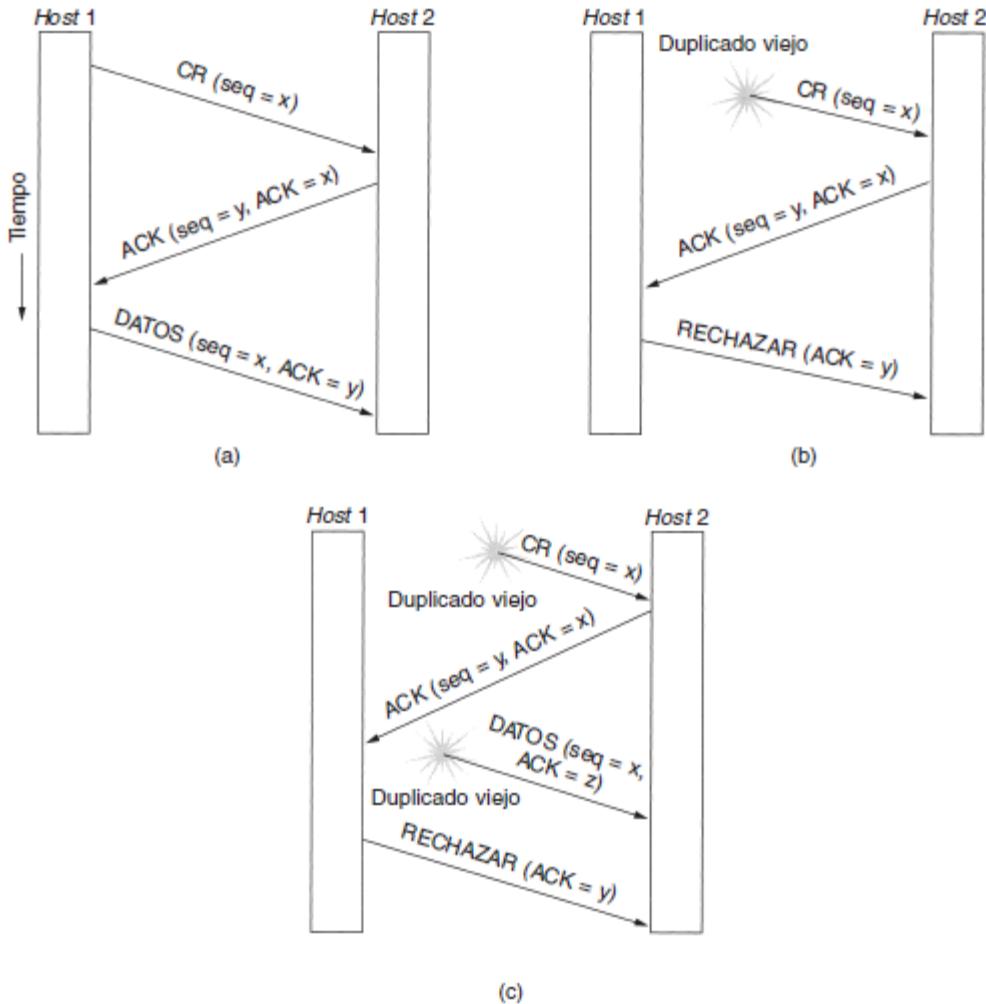


Figura 6-11. Tres escenarios para establecer una conexión usando un acuerdo de tres vías. CR significa CONNECTION REQUEST. (a) Operación normal. (b) CONNECTION REQUEST duplicada vieja que aparece de la nada. (c) CONNECTION REQUEST duplicada y ACK duplicada.

Para un **protocolo de capa de transporte con conexión**, como sería la conexión, cuales son las cosas que negocia. Hay lo que se llama **Saludo de 3 vías** o THREE-WAY HANDSHAKE. Es decir que antes de intercambiar datos necesitaría si o si intercambiar alguna información entre host 1 y host 2.

Suponiendo que Host 1 quiere iniciar una conexión, recordando que estas son tramas numeradas, tiene que enviar su número de secuencia o de trama (SEC=x), a lo cual el otro host debería dar la respuesta, un acuse de recibo con el número de secuencia que le llegó (ACK=x) y el número de secuencia del host 2 (SEC=y), normalmente estos números no arrancan en 0, pueden iniciar en cualquiera y esto es más que nada para temas de seguridad. Y una vez que se ponen de acuerdo recién ahí podrían mandar datos. En TCP podría hacer falta un acuse de recibo para informar que ha llegado la información.

El diagrama “b” y “c” muestra posible pérdidas de información y como se recuperan.

En el “b” se ve un acuse de recibo viejo duplicado al cual el Host 2 va a responder, si el host 1 lo había generado mucho antes le va a cortar la conexión (REJECT). Cuando el Host 2 le responda al Host 1, si el 1 no había iniciado la comunicación la corta.

En el caso “c” se pierde la ida, llega la respuesta, el dato también llega duplicado y finalmente es rechazado.

La única manera de que se establezca la comunicación es que los mensajes se intercambien bien.

The Internet Transport Protocols: TCP

- Introduction to TCP
- The TCP Service Model
- The TCP Protocol
- The TCP Segment Header
- TCP Connection Establishment
- TCP Connection Release
- TCP Connection Management Modeling
- TCP Transmission Policy
- TCP Congestion Control
- TCP Timer Management
- Wireless TCP and UDP
- Transactional TCP

Este es el otro protocolo de la capa de transporte muy usado TCP, es orientado a conexión, con confirmación y con retransmisión en el caso de que se pierdan los datos. Me garantiza de alguna manera que no se pierda la información y que llegue en el mismo orden. UDP como no tiene numeración no me garantiza ni siquiera que llegue.

The TCP Service Model

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

Some assigned ports.

¿Cómo eligen las aplicaciones o procesos el número de puerto a usar? Se hace con algo llamado puerto bien conocido, todos los servidores lo asignan siempre al mismo puerto. Por ejemplo, voy a tener un servidor web funcionando en mi máquina y lo asocio al puerto 80, entonces cualquier máquina del mundo que me quiera hacer una consulta a un servidor web que esté corriendo en mi máquina solo tiene que saber la dirección IP o nombre que esté asociado a esa IP, porque luego el cliente sabe que tiene que ir a buscar el puerto de destino 80.

El que inicia la conexión no es el servidor, este está esperando que alguien le haga una consulta, entonces el cliente tiene que iniciar una conexión (si es TCP), y una vez que esté conectado hace la consulta sabiendo a qué puerto remoto conectarse.

Para el correo, tendrá una IP de Gmail por ejemplo, mis datos podrán llegar a ese servidor, tengo que indicar que el puerto destino sea el 25 SMTP, así le llegan al proceso que es un servidor de correo y va a estar esperando datos en ese puerto.

Normalmente los puertos bien conocidos son los puertos bajos, del 0 al 1000, y están estandarizados.

Por consola se puede hacer "less /etc/services" me muestra todos los puertos bien conocidos

```

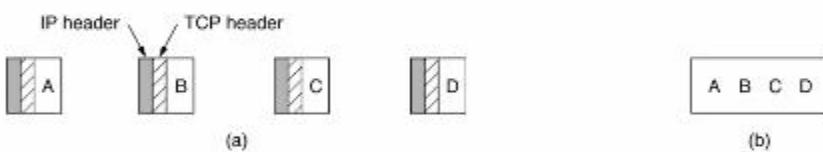
# Network services, Internet style

# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, officially ports have two entries
# even if the protocol doesn't support UDP operations.
#
# Updated from https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml .
#
# New ports will be added on-request if they have been officially assigned
# by IANA and used in the real-world or are needed by a debian package.
# If you need a huge list of used numbers please install the nmap package.

tcpmux      1/tcp          # TCP port service multiplexer
echo        7/tcp
echo        7/udp
discard     9/tcp          sink null
discard     9/udp          sink null
sysstat     11/tcp         users
daytime     13/tcp
daytime     13/udp
netstat     15/tcp
qotd       17/tcp          quote
chargen    19/tcp          ttyst source
chargen    19/udp          ttyst source
ftp-data   20/tcp
ftp        21/tcp          fspd
fspo       21/udp
ssh        22/tcp          # SSH Remote Login Protocol
telnet     23/tcp
astp       25/tcp          xmail
time       37/tcp          timeserver
time       37/udp          timeserver
whois      43/tcp          nickname
tacacs     49/tcp          # Login Host Protocol (TACACS)
tacacs     49/udp
domain    53/tcp          # Domain Name Server
domain    53/udp
bootps    67/udp
bootpc    68/udp
tftp      69/udp
gopher    70/tcp          # Internet Gopher
finger    79/tcp
http      80/tcp          www          # WorldwideWeb HTTP
kerberos  88/tcp          kerberos5  krb5  kerberos-sec  # Kerberos v5
kerberos  88/udp          kerberos5  krb5  kerberos-sec  # Kerberos v5
iso-tsap   182/tcp         tsap          # part of ISO88
scr-nema  184/tcp         dicom         # Digital Imag. & Comm. 300
pop3      110/tcp         pop-3        # POP version 3
sunrpc   111/tcp         portmapper  # RPC 4.0 portmapper
sunrpc   111/udp         portmapper
auth      113/tcp         authentication  tac ident
nntp      119/tcp         readnews  nntp  # USENET News Transfer Protocol
ntp       123/udp         # Network Time Protocol
esmtp     135/tcp         loc-srv     # DCE endpoint resolution
netbios-ns 137/tcp         # NETBIOS Name Service
/jet/services

```

The TCP Service Model (2)



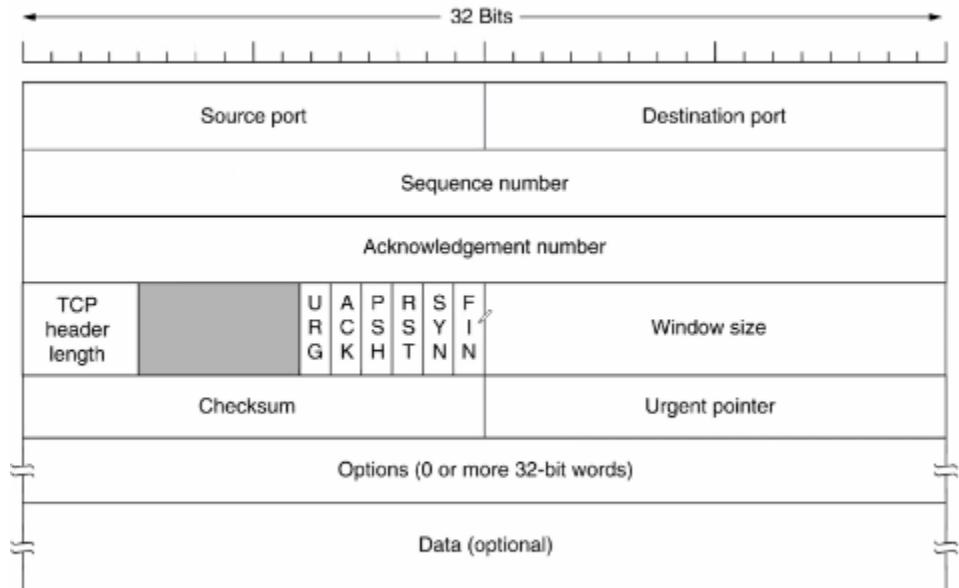
- (a) Four 512-byte segments sent as separate IP datagrams.
 - (b) The 2048 bytes of data delivered to the application in a single READ CALL.

La máquina origen podría enviar 4 segmentos TCP con A, B, C y D de datos en el Payload, podría pasar que en la máquina destino lleguen en distinto orden o se haya perdido C y se retransmite después y por lo tanto va a llegar a A, B, D y luego C, cuando la máquina destino los quiera leer los tendrá en orden.

Se mandan a la red en orden y se leer de la red en el mismo orden, TCP hace eso, UDP no lo puede asegurar.

LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: **TCP** (página 532)

The TCP Segment Header



TCP Header.

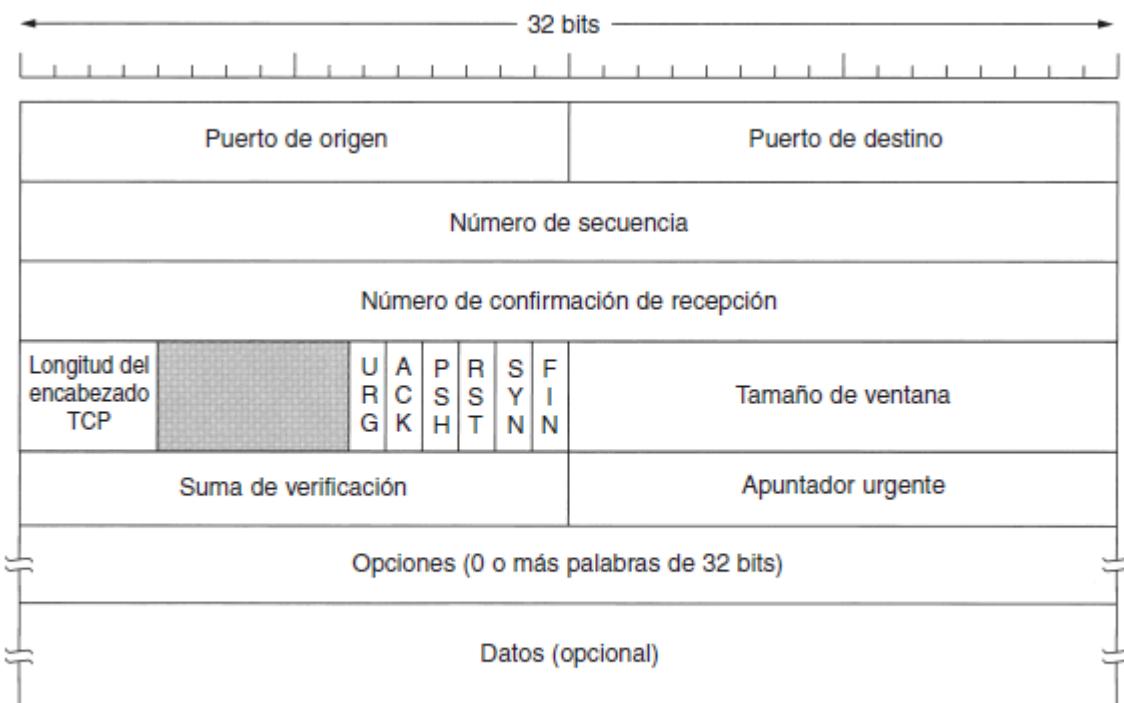


Figura 6-29. Encabezado TCP.

Este es el encabezado de un **segmento TCP**, es un poco más largo, usa 16 bits al igual que UDP para puertos origen y destino.

Tiene un **número de secuencia** que es una numeración que le va dando a cada segmento que es de 32 bits, a medida que va mandando segmentos este número se va incrementando, no cuenta los segmentos si no que cuenta los bytes que manda, entonces el TCP lo va incrementando en la cantidad de Bytes que mando.

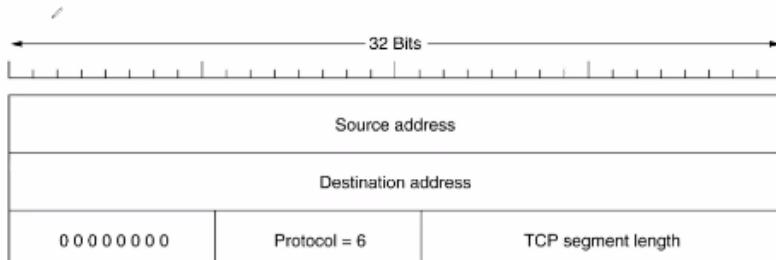
El **número de acuse** de recibo que también son 32 bits, es el que se le envía al que envió la información diciendo cual es el número de secuencia que se le envió antes.

Tiene algunos **bits para hacer inicio y fin de la conexión**, el **SYN** es un bit que se pone en 1, es para indicar que se está iniciando el saludo de tres vías, **FIN** es otro parecido pero es para finalizar la conexión o para liberarla. **RST** es para el caso de que no haya podido iniciar el saludo de 3 vías, **ACK** acuse de recibo es para indicar que el dato que va es válido. **URG** está asociado a **URGENT POINT** en vez de procesar los datos como vienen es para darle prioridad a alguna parte de los datos en especial, entonces con puntero urgente puedo hacer que procese esos datos primero. **PSH** tiene que ver con la aplicación, cuando desde la aplicación escribo algunos datos y le digo al sistema operativo que se encargue de mandarlo muchas veces no lo manda inmediatamente, espera por si hay más datos, para poder generar un segmento más grande, aprovechando con un encabezado enviar más datos, con este PSH la aplicación le dice al operativo que no espere nada y que lo mande, esto lo usan algunas aplicaciones interactivas, TELNET por ejemplo.

Tiene un **Checksum** que en TCP es **obligatorio**, como tiene un encabezado y puede tener opciones, tengo un tamaño de encabezado que es variable.

Por ultimo tengo un **tamaño de ventana** que es el buffer de salida, de datos que tengo esperando, una vez que me mandan el acuse de recibo del destinatario lo saco de ese buffer y puedo seguir transmitiendo nuevos datos. El tamaño de ventana es variable y puedo hacer control de flujo con esto. Entonces si mi host no puede procesar todos los datos que le llegan porque es más lento que el emisor, lo que puedo hacer es mandarle un acuse de recibo poniendo que el tamaño de ventana es 0, entonces el otro host hasta que no modifique el tamaño de ventana no me puede enviar otro dato.

The TCP Segment Header (2)



The pseudoheader included in the TCP checksum.

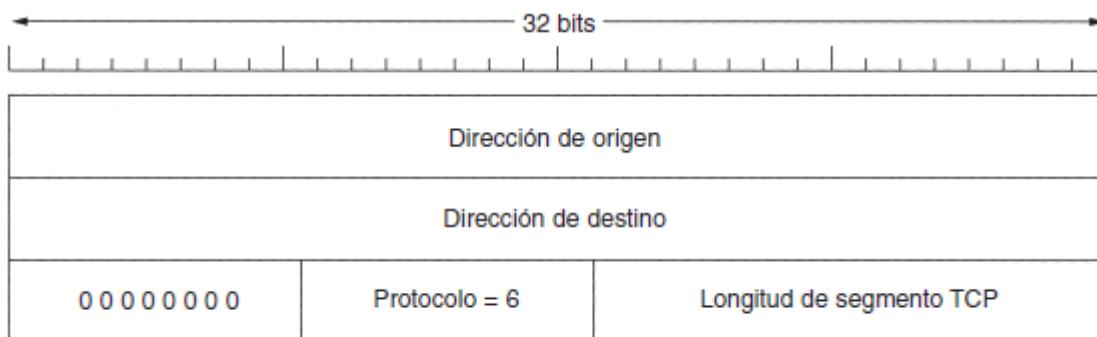
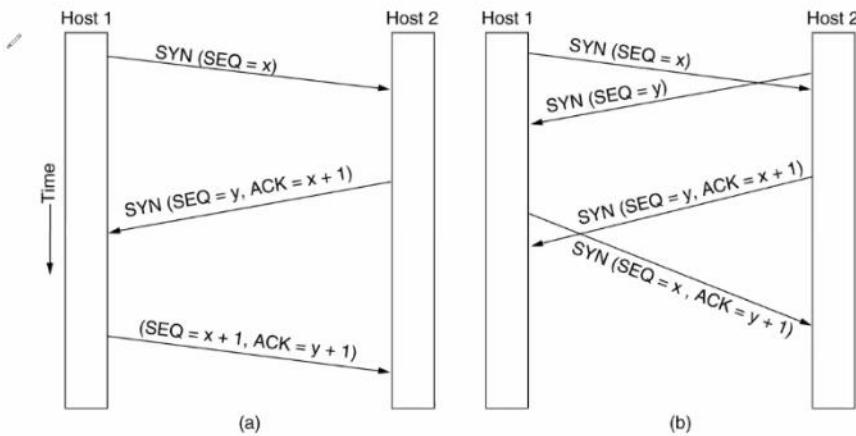


Figura 6-30. Pseudoencabezado incluido en la suma de verificación del TCP.

TCP Connection Establishment



(a) TCP connection establishment in the normal case.
 (b) Call collision.

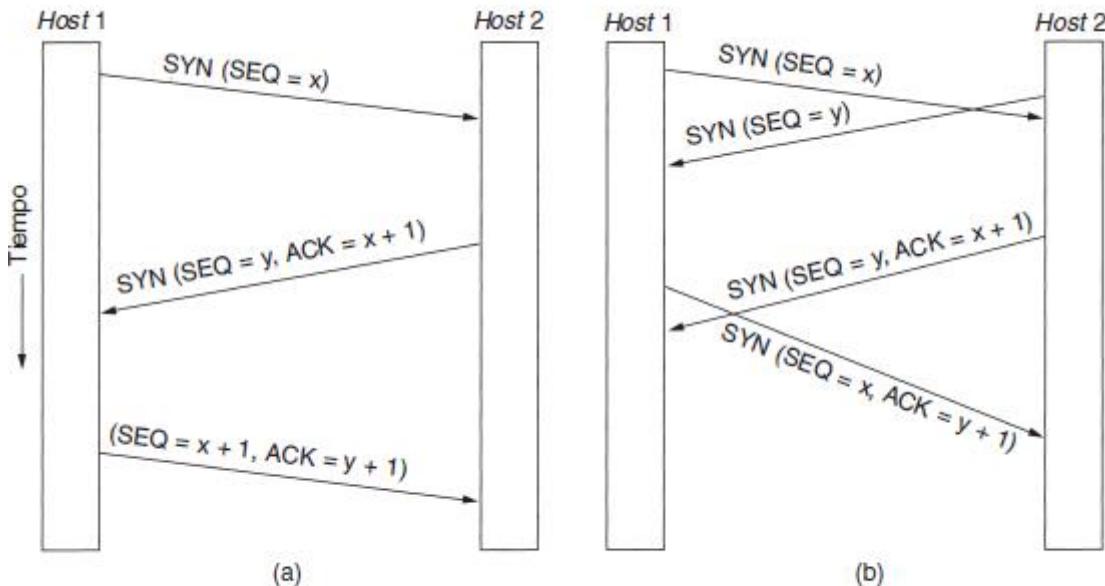
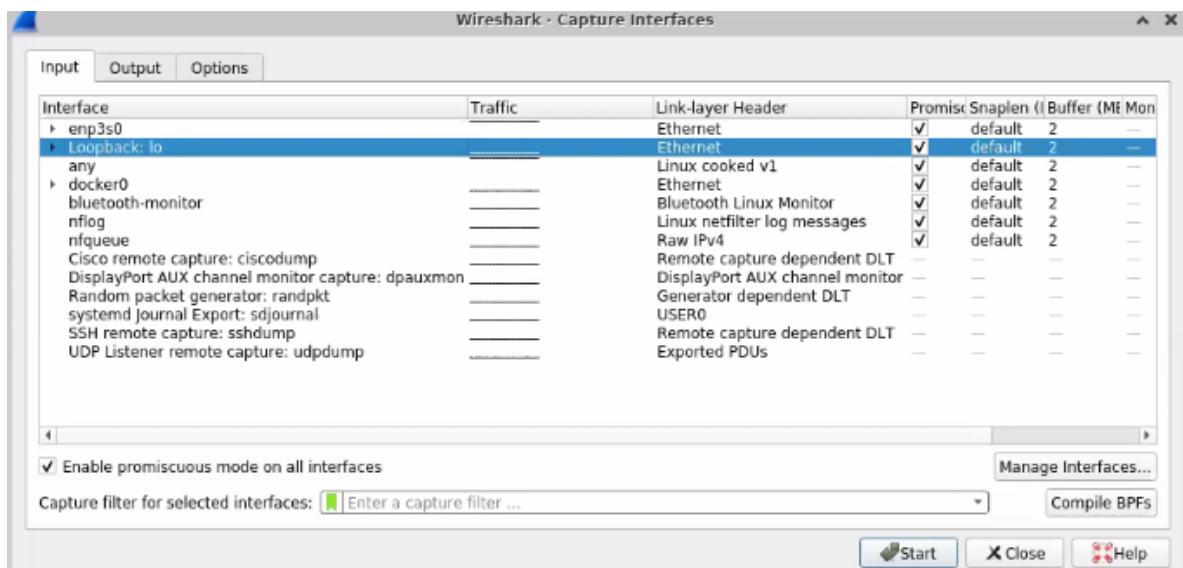


Figura 6-31. (a) Establecimiento de una conexión TCP en el caso normal. (b) Colisión de llamadas.

Aca podemos ver un caso donde se inicia la conexión, uno que funciona bien y otro donde hay colisión.

Vamos a correr dos procesos en la máquina, uno que esté esperando datos y el otro que se conecte y mande datos. Estas aplicaciones son utilitarios



Como vamos a usar la misma maquina no vamos a usar la interfaz de red Ethernet, va a utilizar una placa virtual.

Desde consola vamos a usar algo que se llama **NETCAD o NC**, que es la navaja suiza de los utilitarios de red para probar muchas cosas. Le vamos a pedir que escuche conexiones **asociándolo al puerto 6000**.

```
caffeine@maquinola:~/Downloads/cap3/3.6$ nc -l 6000
```

Por otro lado voy a usar un cliente TELNET que se conecta a un servidor, le tengo que decir a quien y el puerto:

```
caffeine@maquinola:~/Downloads/cap3/3.6$ telnet 127.0.0.1 5000
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
caffeine@maquinola:~/Downloads/cap3/3.6$
```

No.	Time	Source	Destination	Protocol	Length	Info
25	80.59392242	127.0.0.1	127.0.0.1	TCP	74	60596 → 5000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=...
26	80.593950621	127.0.0.1	127.0.0.1	TCP	54	5000 → 60596 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Frame 25: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0
 Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
 Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 ...
 Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
 Total Length: 60
 Identification: 0xd2d5 (53973)
 Flags: 0x4000, Don't fragment
 Fragment offset: 0
 Time to live: 64
 Protocol: TCP (6)
 Header checksum: 0x69d4 [validation disabled]
 [Header checksum status: Unverified]
 Source: 127.0.0.1
 Destination: 127.0.0.1
 Transmission Control Protocol, Src Port: 60596, Dst Port: 5000, Seq: 0, Len: 0

Arriba da un resumen, en la 2da ventana un desglose.

En el primero se puede ver que dice **[SYN]** lo que significa que inicio conexión, mi misma máquina, por un puerto cualquiera y se quiere conectar al **puerto 5000** (tendría que generar un problema).

```

> Frame 25: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
-> Transmission Control Protocol, Src Port: 60596, Dst Port: 5000, Seq: 0, Len: 0
  Source Port: 60596
  Destination Port: 5000
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 0      (relative sequence number)
  Sequence number (raw): 2015311730
  [Next sequence number: 1      (relative sequence number)]
  Acknowledgment number: 0
  Acknowledgment number (raw): 0
  1010 .... = Header Length: 40 bytes (10)
-> Flags: 0x002 (SYN)
  Window size value: 65495
  [Calculated window size: 65495]
  Checksum: 0xfe30 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
-> Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
-> [Timestamps]

```

Se puede ver el puerto origen y el puerto de destino que es el 5000.

```

> Frame 25: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
-> Transmission Control Protocol, Src Port: 60596, Dst Port: 5000, Seq: 0, Len: 0
  Source Port: 60596
  Destination Port: 5000
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 0      (relative sequence number)
  Sequence number (raw): 2015311730
  [Next sequence number: 1      (relative sequence number)]
  Acknowledgment number: 0
  Acknowledgment number (raw): 0
  1010 .... = Header Length: 40 bytes (10)
-> Flags: 0x002 (SYN)
  000. .... .... = Reserved: Not set
  ...0 .... .... = Nonce: Not set
  ....0 .... .... = Congestion Window Reduced (CWR): Not set
  ....0.... .... = ECN-Echo: Not set
  ....0.... .... = Urgent: Not set
  ....0.... .... = Acknowledgment: Not set
  ....0.... .... = Push: Not set
  ....0.... .... = Reset: Not set
  ....0....1.... = Syn: Set
  ....0....0.... = Fin: Not set
  [TCP Flags: .....S..]
  Window size value: 65495
  [Calculated window size: 65495]
  Checksum: 0xfe30 [unverified]
  [Checksum Status: Unverified]

```

0000	00 00 00 00 00 00 00 00 00 00 00 00 00 45 10E.....
0010	00 3c d2 d5 40 00 40 06 69 d4 7f 00 00 01 7f 00	< @ @ 1.....
0020	00 01 ec b4 13 88 78 1f 37 72 00 00 00 00 a0 02	...X 7r.....
0030	ff d7 fe 30 00 00 02 04 ff d7 04 02 08 0a e5 6b	...0.....k.....
0040	c8 67 00 00 00 00 01 03 03 07	g.....

Se pueden ver los **FLAGS** donde SYN esta en 1.

Se puede ver un numero de secuencia (en uno lo muestra relativo), este es del cliente, el que inicia la conexión, numero de acuse de recibo no tiene porque aun no hay respuesta, tampoco van datos.

Luego si vemos la respuesta del otro proceso:

tcp.port == 5000

No.	Time	Source	Destination	Protocol	Length	Info
25	80.593922242	127.0.0.1	127.0.0.1	TCP	74	60596 → 5800 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=...
26	80.593950621	127.0.0.1	127.0.0.1	TCP	54	5800 → 60596 [RST, ACK] Seq=1 Ack=1 Win=0 Len=8

```

> Frame 26: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 5000, Dst Port: 60596, Seq: 1, Ack: 1, Len: 0
  Source Port: 5000
  Destination Port: 60596
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 1 (relative sequence number)
  Sequence number [raw]: 0
  [Next sequence number: 1 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  Acknowledgment number [raw]: 2015311731
  0x01 .... = Header Length: 20 bytes (5)
- Flags: 0x014 (RST, ACK)
  000.... .... = Reserved: Not set
  ...0.... .... = Nonce: Not set
  ....0.... .... = Congestion Window Reduced (CWR): Not set
  ....0.... .... = ECN-Echo: Not set
  ....0.... .... = Urgent: Not set
  ....1.... .... = Acknowledgment: Set
  ....0.... .... = push: Not set
  ....1.... .... = Reset: Set
  ....0.... .... = Syn: Not set
  ....0.... .... = Fin: Not set
  [TCP Flags: ....A-R-]
  Window size value: 0
  [Calculated window size: 0]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0x01ff [unverified]

0000 00 00 00 00 00 00 00 00 00 00 00 00 00 45 10  .....E.
0010 00 28 00 00 40 00 40 06 3c be 7f 00 00 01 7f 00  { .. @ < ...
0020 00 01 13 88 ec b4 00 00 00 00 78 1f 37 73 50 14  ....@.x-7sp.
0030 00 00 01 ff 00 00 .....I

```

Se puede ver que le contesto reset RST, porque no fue posible conectarse, y el acuse de recibo le incremento 1.

Ahora vamos a hacer bien las cosas:

```

caffeine@maquinola:~/Downloads/cap3/3.6$ telnet 127.0.0.1 5000
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
caffeine@maquinola:~/Downloads/cap3/3.6$ telnet 127.0.0.1 6000
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
I

```

Antes había puesto que no se pudo conectar, ahora en el puerto 6000 si.

```

caffeine@maquinola:~/Downloads/cap3/3.6$ telnet 127.0.0.1 5000
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
caffeine@maquinola:~/Downloads/cap3/3.6$ telnet 127.0.0.1 6000
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
holaaaa
chauuuuu
I

```

Este es el cliente, mando un hola y recibo la respuesta chau que envío desde la otra consola

```

caffeine@maquinola:~/Downloads/cap3/3.6$ host www.frm.utn.edu.ar
www.frm.utn.edu.ar is an alias for web.frm.utn.edu.ar.
web.frm.utn.edu.ar has address 200.10.196.21
caffeine@maquinola:~/Downloads/cap3/3.6$ host www.frsr.utn.edu.ar
www.frsr.utn.edu.ar has address 179.63.240.122
caffeine@maquinola:~/Downloads/cap3/3.6$ less /etc/services
caffeine@maquinola:~/Downloads/cap3/3.6$ nc -l 6080
holaaaa
chauuuuu

```

tcp.port == 6000						
No.	Time	Source	Destination	Protocol	Length	Info
7	1.405206958	127.0.0.1	127.0.0.1	TCP	74	47936 → 6000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=
8	1.4952505642	127.0.0.1	127.0.0.1	TCP	74	6000 → 47936 [SYN, ACK] Seq=0 Ack=1 Win=65493 Len=0 MSS=65495
9	1.485284792	127.0.0.1	127.0.0.1	TCP	66	47936 → 6000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=38492143..
10	12.132698545	127.0.0.1	127.0.0.1	TCP	75	47936 → 6000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=9 TSval=384..
11	12.132732570	127.0.0.1	127.0.0.1	TCP	66	6000 → 47936 [ACK] Seq=1 Ack=10 Win=65536 Len=0 TSval=3849225..
12	20.731797866	127.0.0.1	127.0.0.1	TCP	75	6000 → 47936 [PSH, ACK] Seq=1 Ack=10 Win=65536 Len=0 TSval=38..
13	20.731827231	127.0.0.1	127.0.0.1	TCP	66	47936 → 6000 [ACK] Seq=10 Ack=10 Win=65536 Len=0 TSval=384923..
14	33.640181120	127.0.0.1	127.0.0.1	TCP	66	47936 → 6000 [FIN, ACK] Seq=10 Ack=10 Win=65536 Len=0 TSval=3..
15	33.640294518	127.0.0.1	127.0.0.1	TCP	66	6000 → 47936 [FIN, ACK] Seq=10 Ack=11 Win=65536 Len=0 TSval=3..
16	33.640326742	127.0.0.1	127.0.0.1	TCP	66	47936 → 6000 [ACK] Seq=11 Ack=11 Win=65536 Len=0 TSval=384924..

En el software voy a filtrar todo lo del puerto 6000, lo que se ve gris es el inicio de sesión y el fin de sesión.

Frame 7: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 47936, Dst Port: 6000, Seq: 0, Len: 0
Source Port: 47936
Destination Port: 6000
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Sequence number (raw): 1035567837
[Next sequence number: 1 (relative sequence number)]
Acknowledgment number: 0
Acknowledgment number (raw): 0
1010 = Header Length: 40 bytes (10)
▼ Flags: 0x002 (SYN)
000. = Reserved: Not set
...0 = Nonce: Not set
...0.... = Congestion Window Reduced (CWR): Not set
...0.... = ECN-Echo: Not set
...0.... = Urgent: Not set
...0.... = Acknowledgment: Not set
...0.... = Push: Not set
...0.... = Reset: Not set
...0.... = 1 = Syn: Set
...0.... = 0 = Fin: Not set
[TCP Flags:S.]
Window size value: 65495
[Calculated window size: 65495]
Checksum: 0xfe30 [unverified]
[Checksum Status: Unverified]

El primer paquete podemos ver que tiene **SYN en 1**, el número de secuencia es el primero para el cliente y el número termina en 837, y no manda datos.

En el 2do paquete se puede ver que dice **[SYN, ACK]** porque le está dando el acuse de recibo el proceso que está en el puerto 6000 que es el servidor, al cliente.

Frame 8: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 6000, Dst Port: 47936, Seq: 0, Ack: 1, Len: 0
Source Port: 6000
Destination Port: 47936
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Sequence number (raw): 671764228
[Next sequence number: 1 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Acknowledgment number (raw): 1835567838
1010 = Header Length: 40 bytes (10)
▼ Flags: 0x012 (SYN, ACK)
000. = Reserved: Not set
...0 = Nonce: Not set
...0.... = Congestion Window Reduced (CWR): Not set
...0.... = ECN-Echo: Not set
...0.... = Urgent: Not set
...1.... = Acknowledgment: Set
...0.... = Push: Not set
...0.... = Reset: Not set
...0.... = 1 = Syn: Set
...0.... = 0 = Fin: Not set
[TCP Flags:A-S.]
Window size value: 65483
[Calculated window size: 65483]
Checksum: 0xfe30 [unverified]
[Checksum Status: Unverified]

Entonces el segundo paquete se puede ver que sale del 6000, ahora tiene otro numero de secuencia que termina en 228, nada que ver con el otro, porque cada uno tiene su numero de secuencia, pero el acuse de recibo es el 838, que fue el que le llego pero incrementado en 1. Pone el **SYN en 1** y el **ACK también en 1**.

```

> Frame 9: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 47936, Dst Port: 6000, Seq: 1, Ack: 1, Len: 0
    Source Port: 47936
    Destination Port: 6000
    [Stream index: 0]
    [TCP Segment Len: 8]
    Sequence number: 1 (relative sequence number)
    Sequence number (raw): 1035567838
    [Next sequence number: 1 (relative sequence number)]
    Acknowledgment number: 1 (relative ack number)
    Acknowledgment number (raw): 671764229
    1000 .... = Header Length: 32 bytes (B)
    Flags: 0x010 (ACK)
        000. .... = Reserved: Not set
        ...0 .... = Nonce: Not set
        .... 0.... = Congestion Window Reduced (CWR): Not set
        .... 0.... = ECN-Echo: Not set
        .... 0.... = Urgent: Not set
        .... 1.... = Acknowledgment: Set
        .... 0.... = Push: Not set
        .... 0.... = Reset: Not set
        .... 0.... = Syn: Not set
        .... 0.... = Fin: Not set
        [TCP Flags: .....A....]
    Window size value: 512
    [Calculated window size: 65536]
    [Window size scaling factor: 128]
    Checksum: 0xfe28 [unverified]

```

El 3er segmento tiene el acuse de recibo del servidor, el número de secuencia es el 838, y el ACK espera datos del 229, incremento en 1 el que mando el 2do. Hasta aca tampoco tenemos datos.

Se puede ver en el 4to que esta el PSH, porque una vez que doy Enter para mandar datos no quiero que el operativo se quede esperando para ver si hay algo más para enviar.

```

Sequence number: 1 (relative sequence number)
Sequence number (raw): 1035567838
[Next sequence number: 10 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Acknowledgment number (raw): 671764229
1000 .... = Header Length: 32 bytes (8)
    Flags: 0x018 (PSH, ACK)
        000. .... = Reserved: Not set
        ...0 .... = Nonce: Not set
        .... 0.... = Congestion Window Reduced (CWR): Not set
        .... 0.... = ECN-Echo: Not set
        .... 0.... = Urgent: Not set
        .... 1.... = Acknowledgment: Set
        .... 1... = Push: Set
        .... 0.... = Reset: Not set
        .... 0.... = Syn: Not set
        .... 0.... = Fin: Not set
        [TCP Flags: .....AP...]
    Window size value: 512
    [Calculated window size: 65536]
    [Window size scaling factor: 128]
    Checksum: 0xfe31 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    [SEQ/ACK analysis]
    [Timestamps]
    TCP payload (9 bytes)
    TCP segment data (9 bytes)

0000  00 00 00 00 00 00 00 00 00 00 00 00 00 45 10  ....E.
0010  00 3d 88 93 40 00 40 06 b4 15 7f 00 00 01 7f 00  = @ @ ....
0020  00 01 bb 40 17 70 3d b9 82 de 28 0a 4f 05 80 18  ..@ p= { 0 ...
0030  02 00 fe 31 00 00 01 01 00 0a e5 6e 83 83 e5 6e  .1.....n..n
0040  59 9b 68 6f 6c 61 61 61 8d 0a  Y holaaa B ..

```

Y en este 4to si tenemos datos, si vamos al Payload podemos ver entre los datos el “holaaa”.

El paquete 10 manda el hola, el 11 envía el ACK del hola, 12 lo genera el servidor y se lo manda al cliente “chau”, en el 14 inicia un fin de conexión, a lo cual el servidor en el paquete 15 le confirma

GLOSARIO:

INTERRED: Conjunto de redes interconectadas. LANs conectadas por una WAN

PROTOCOLO: Es un acuerdo entre las partes en comunicación sobre cómo se debe llevar a cabo la comunicación.

ARQUITECTURA DE RED: Un conjunto de capas y protocolos.

OSI: Interconexión de sistemas abiertos

TCP: Protocolo de Control de Transmisión

UDP: Protocolo de Datagrama de Usuario

TTL: Tiempo de vida

DNS: Sistema de Nombres de Dominio

WWW: World Wide Web

ISPs: Proveedores de servicios de Internet

ATM: Modo de Transferencia Asíncrona

MAC: Control de Acceso al Medio

FDM: Multiplexión por División de Frecuencia

TDM: Multiplexión por División de Tiempo

LAN: Red de Área Local

MAN: Red de Área Metropolitana

WAN: Red de Área Amplia

CSMA: Acceso Múltiple con Detección de Portadora

CSMA/CD: Acceso Múltiple con Detección de Portadora y Detección de Colisiones

WDMA: Acceso Múltiple por División de Longitud de Onda)

MACA: Acceso Múltiple con Prevención de Colisiones

MACAW: MACA Inalámbrico

RTS: Solicitud de Envío

CTS: Libre para Envío

Multidifusión: Multicast

Difusión: Broadcast

LLC: Control Lógico del Enlace

MTU: Unidad máxima de transferencia

NAT: Traducción de Dirección de Red

TPDU: Unidad de Datos del Protocolo de Transporte

TSAP: Punto de Acceso al Servicio de Transporte

NSAP: Punto de Acceso al Servicio de Red

CIDR: Enrutamiento interdominios sin clases (Classless)