

Augusto Rabbia

---

## **Resumen Comunicaciones**

---

Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Licenciatura en Ciencias de la Computación

2023-2024



## Índice

<b>1. Introducción</b>	<b>5</b>
1.1. Protocolo TCP/IP . . . . .	5
1.2. Hardware de Red . . . . .	5
1.3. Software de Red . . . . .	6
1.4. Modelos de Referencia . . . . .	8
1.4.1. Modelo OSI . . . . .	8
1.4.2. Modelo TCP/IP . . . . .	10
1.4.3. Diferencias entre OSI y TCP/IP . . . . .	11
<b>2. Capa Física</b>	<b>11</b>
2.1. Bases teóricas . . . . .	11
2.1.1. Análisis de Fourier . . . . .	11
2.1.2. Tasa de datos máxima de un canal . . . . .	14
2.2. Medios de transmisión guiados . . . . .	15
2.2.1. Medios magnéticos . . . . .	15
2.2.2. Par Trenzado . . . . .	15
2.2.3. Coax . . . . .	16
2.2.4. Líneas eléctricas . . . . .	16
2.2.5. Fibra óptica . . . . .	16
2.3. Medios de Transmisión inalámbricos . . . . .	19
2.3.1. El espectro electromagnético . . . . .	19
2.3.2. Transmisión infrarroja . . . . .	22
2.3.3. Transmisión por ondas de luz . . . . .	22
2.3.4. Satélites de comunicación . . . . .	23
2.4. Modulación digital . . . . .	25
2.4.1. Transmisión en banda base . . . . .	25
2.4.2. Transmisión pasa-banda . . . . .	28
2.5. Multiplexión . . . . .	30
2.5.1. Multiplexión por División de Frecuencia . . . . .	30
2.5.2. Multiplexión por División de Tiempo . . . . .	32
<b>3. Capa de Enlace de Datos</b>	<b>32</b>
3.1. Diseño de la Capa de Enlace de Datos . . . . .	32
3.2. Servicios proporcionados a la capa de red . . . . .	33
3.3. Entramado . . . . .	33
3.4. Control de errores . . . . .	35
3.5. Control de flujo . . . . .	36



3.6. Protocolo Punto a Punto (FALTA) . . . . .	36
<b>4. Subcapa de Control de Acceso al Medio (MAC)</b> . . . . .	<b>36</b>
4.1. El problema de la asignación del canal . . . . .	36
4.1.1. Supuestos para la asignación dinámica de canales . . . . .	37
4.2. Protocolos de Acceso Múltiple . . . . .	37
4.2.1. Aloha . . . . .	37
4.2.2. Protocolos de acceso múltiple con detección de portadora . . . . .	38
4.2.3. Protocolos libres de colisiones . . . . .	39
4.3. Ethernet . . . . .	41
4.3.1. Ethernet clásica . . . . .	41
4.3.2. Ethernet comutada . . . . .	43
<b>5. Redes Inalámbricas</b> . . . . .	<b>44</b>
5.1. WLAN: WiFi . . . . .	44
5.1.1. Capa física . . . . .	45
5.1.2. Capa de enlace . . . . .	47
5.1.3. Asociándose a una red WiFi . . . . .	49
5.2. Bluetooth (FALTA) . . . . .	50
5.3. LTE (FALTA) . . . . .	50
<b>6. Capa de Red</b> . . . . .	<b>50</b>
6.1. IPv4 / Protocolo IP . . . . .	50
6.1.1. Datagrama IP . . . . .	50
6.1.2. Direcciones IPv4 . . . . .	51
6.1.3. Comutación/Switching (Routing) . . . . .	52
6.1.4. Fragmentación . . . . .	53
6.2. ICMP . . . . .	54
6.3. ARP . . . . .	55
6.4. IPv6 . . . . .	56
6.4.1. Cabeceras . . . . .	56
6.4.2. Direcciones IPv6 . . . . .	58
6.4.3. Funcionalidades . . . . .	60
6.4.4. Transición IPv4 a IPv6 . . . . .	61
<b>7. Capa de Transporte</b> . . . . .	<b>61</b>
7.1. Puertos . . . . .	62
7.2. User Datagram Protocol (UDP) . . . . .	63
7.2.1. Formato de los Datagramas UDP . . . . .	63



7.2.2. Multiplexión UDP . . . . .	65
7.3. Transmission Control Protocol (TCP) . . . . .	65
7.3.1. Propiedades de un servicio de envíos confiable . . . . .	65
7.3.2. Proveyendo seguridad . . . . .	66
7.3.3. Ventana deslizante . . . . .	66
7.3.4. El protocolo TCP . . . . .	68
7.3.5. Puertos, Endpoints y Conexiones . . . . .	68
7.3.6. Segmentos, Streams y Números de Secuencia . . . . .	69
7.3.7. Formato de los Segmentos TCP . . . . .	70
7.3.8. Datos fuera de banda . . . . .	71
7.3.9. Retransmisión . . . . .	71
7.3.10. Estableciendo conexiones TCP . . . . .	73
7.3.11. Cerrando conexiones TCP . . . . .	74
7.3.12. Forzando el envío de datos . . . . .	75
7.3.13. El síndrome de la ventana tonta . . . . .	76
<b>8. Capa de Aplicación</b>	<b>77</b>
8.1. DNS . . . . .	77
8.1.1. Implementación . . . . .	77
8.1.2. Zone files . . . . .	79
8.1.3. Queries . . . . .	82
8.1.4. Actualizaciones de zona . . . . .	84
8.1.5. Seguridad - Amenazas y clasificación . . . . .	86
8.1.6. Reverse mapping . . . . .	86
8.1.7. Vistas . . . . .	87
8.1.8. Configuraciones DNS . . . . .	88
8.2. Firewall . . . . .	92
8.2.1. Métodos de ataque . . . . .	93
8.2.2. Firewalls . . . . .	93
8.2.3. IP Filtering . . . . .	94
8.2.4. IP Tables . . . . .	95
<b>9. Conceptos adicionales</b>	<b>97</b>
9.0.1. Hardware de red . . . . .	98
9.1. VLANs . . . . .	99
9.2. BOOTP . . . . .	99
<b>A. Bibliografía</b>	<b>100</b>



## Antes de leer

Este resumen fue creado por Augusto Rabbia e incluye los contenidos de Comunicaciones del año 2023.

Cualquier comentario, mejora, o idea puede ser enviada por mail a rabbiaaugusto1@gmail.com, y haré los cambios pertinentes, actualizando el resumen en Github. Invito a que lo hagan. Cualquier contribución será mencionada y agradecida.

- Última modificación: Abril 2024.



## 1. Introducción

*Los temas de esta sección corresponden a la unidad 1 de los libros Tanembaum y Comer. No se cubren las siguientes secciones al no ser centrales en el cursado, pero se recomienda su lectura de forma introductoria:*

- *Comer: 1.1, 1.3-12*
- *Tanembaum 1.5-9*

Una **red de computadoras** es un conjunto de computadoras interconectadas capaces de realizar un trabajo conjunto.

### 1.1. Protocolo TCP/IP

El **suite de protocolos TCP/IP** fue fundado originalmente por el ARPA (Advanced Research Projects Agency). Este es el suite más utilizado en la actualidad y puede ser utilizado para interconectar cualquier conjunto de redes (Por ejemplo, algunas empresas lo usan para conectarse de forma local incluso si no están conectadas al Internet global, e individuos la usan para comunicarse a través del mundo).

### 1.2. Hardware de Red

- *Tanembaum 1.2*

Las redes pueden calificarse según dos criterios:

#### Según su tecnología de transmisión

- **Enlaces de difusión (broadcast)**: En una red de difusión, todas las computadoras de la red comparten el canal de comunicación. En estas, el paquete enviado por una máquina es recibido por todas, y cada una deberá verificar si el mensaje le correspondía.
- **Enlaces punto a punto (unicast)**: Estos conectan pares individuales de máquinas. Los paquetes pueden necesitar visitar varias máquinas intermedias, pero sólo hay un emisor y un receptor.

#### Según su tamaño físico

- **Personal Area Network (PAN)**: Redes pequeñas de unos pocos metros (i.e. Bluetooth).
- **Local Area Network (LAN)**: Redes que operan dentro de un edificio. Estas tienen un tamaño limitado, pero esto incrementa su simplicidad. Estas pueden ser redes alámbricas (que siguen un estándar llamado IEEE 802.11, o WiFi) o alámbricas (que utilizan el estándar IEEE 802.3, o Ethernet).
- **Metropolitan Area Network (MAN)**: El término está outdated, son redes que cubren toda una ciudad.

- **Wide Area Network (WAN)**: Cubren áreas extensas, en general un país o continente. Conectan, por ejemplo, dos sedes diferentes de una empresa en diferentes lugares físicos. Estas pueden tener múltiples subredes LAN, y tienen dos componentes: las **líneas de transmisión**, el medio que transporta los bits entre dispositivos, y los **elementos de commutación** (hoy conocidos como **routers**).
- **Internet**: A una **colección de redes interconectadas** se le conoce como interred o internet. Llamaremos a la interred mundial **Internet** (con I mayúscula). No hay una buena distinción para una interred, técnicamente, conectar 2 LANs formaría una interred. Sin embargo, se suele dar este nombre cuando hayan al menos dos redes de empresas diferentes conectadas.

### 1.3. Software de Red

- *Tanenbaum 1.3*

Hoy en día, el software de red está altamente estructurado, existiendo como base del mismo una **jerarquía de protocolos**. Para reducir la complejidad, los protocolos se dividen en capas. La función de cada capa es brindar servicios a las capas superiores a ella.

Cuando la capa  $n$  en una máquina se comunica con la capa  $n$  en otra máquina (cosa que se lo llama iguales o **peers**), a las reglas y convenciones utilizadas en esta conversación se les conoce como **el protocolo de la capa  $n$** .

#### Servicios, protocolos e interfaces

Un **servicio** se puede expresar como un conjunto de **primitivas** disponibles a la capa superior. El servicio indica lo que hace la capa, no cómo lo hace, ni cómo pueden acceder a esto las capas superiores.

Primitiva	Significado
LISTEN	Bloquea en espera de una conexión entrante.
CONNECT	Establece una conexión con un igual en espera.
ACCEPT	Acepta una conexión entrante de un igual.
RECEIVE	Bloquea en espera de un mensaje entrante.
SEND	Envía un mensaje al igual.
DISCONNECT	Termina una conexión.

Figura 1: Seis primitivas de servicios que proveen un servicio orientado a la conexión.<sup>1</sup>

La **interfaz** de una capa indica a los procesos superiores cómo pueden acceder a ella.

Por último, la capa decidirá qué **protocolos** usar, siempre que estos provean los servicios que corresponden. Los protocolos se utilizan para **implementar** los servicios de una capa.

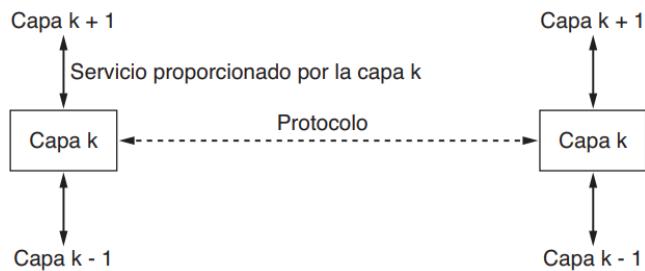


Figura 2: La relación entre un servicio y un protocolo.<sup>1</sup>

## Arquitecturas de Red

Una **arquitectura de red** es el conjunto de capas y protocolos. Esta da la información para que los programadores puedan programar cada capa, pero NO es la implementación en sí. La secuencia de protocolos usados por un sistema se llama pila de protocolos.

Aspectos de diseño para las capas

### 1. Confiabilidad

- Detección de errores.
- Posible corrección de errores.
- Enrutamiento (capacidad de encontrar una ruta hacia el destino de forma inteligente).

### 2. Escalabilidad

- Distribución de los protocolos en capas.
- Direccionalamiento (mecanismo para identificar los emisores y receptores involucrados en un mensaje específico).
- Internetworking (capacidad de comunicar redes con distintas limitaciones).

### 3. Seguridad

- Confidencialidad.
- Autenticación (evitar que alguien intente hacerse pasar por otra persona).
- Integridad (que no se modifiquen mensajes de forma maliciosa).

## 1.4. Modelos de Referencia

### 1.4.1. Modelo OSI

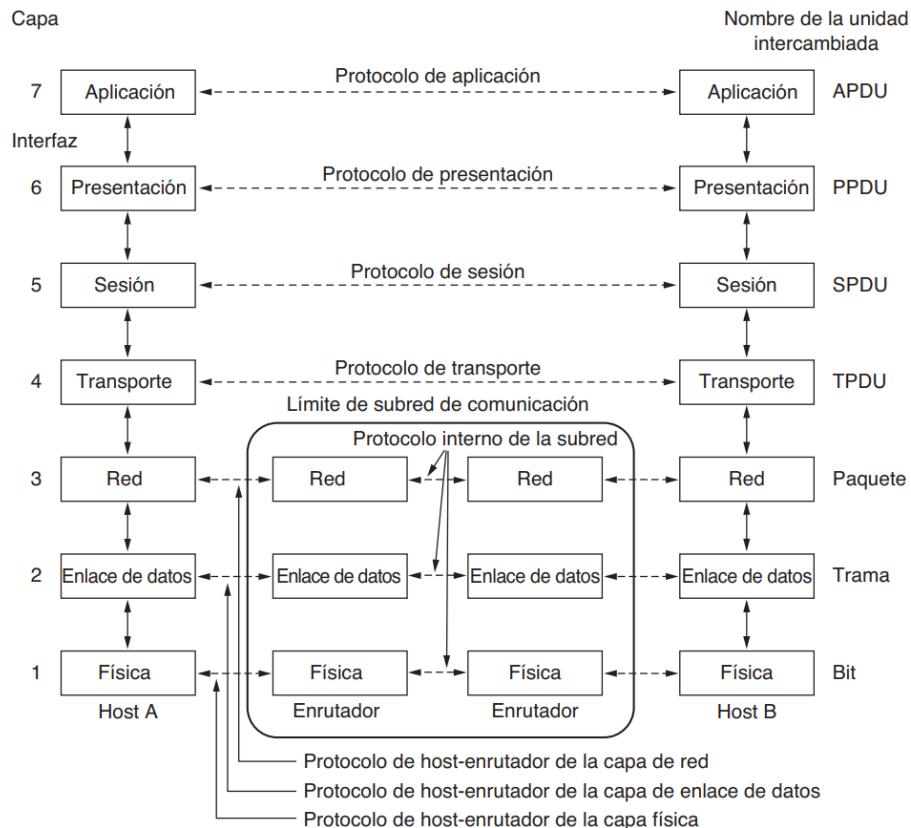


Figura 3: Modelo de referencia OSI.<sup>1</sup>

El modelo **OSI (Open Systems Interconnection)** fue creado en 1995, y fue el primer paso a la estandarización internacional de las diversas capas.

Los principios que se aplicaron para llegar a sus siete capas se pueden resumir de la siguiente manera:

- Se debe crear una capa en donde se requiera un nivel diferente de abstracción.
- Cada capa debe realizar una función bien definida.
- La función de cada capa se elige teniendo en cuenta la definición de protocolos estandarizados internacionalmente.
- Los límites de las capas deben minimizar el flujo de información a través de las interfaces.
- Debe haber suficientes capas como para que ninguna tenga que agrupar funciones distintas. Además, cada capa debe ser pequeña, para que la arquitectura no se vuelva inmanejable.



Las capas del modelo OSI:

## 1) Capa física:

La capa física se encarga de transmitir los bits puros a través de un canal de transmisión. Su diseño tienen que ver con las interfaces mecánica, eléctrica y de temporización, y con el medio de transmisión físico que se encuentra bajo la capa física.

## 2) Capa de enlace de datos

La capa de enlace de datos se encarga de eliminar los errores de transmisión que surgen en la capa física, de tal forma que la capa de red no los vea.

Para lograrlo, el emisor divide los datos en **tramas** y las transmite. Si el servicio es confiable, el receptor confirma la recepción de cada una. También se encarga del control de flujo entre 2 dispositivos.

En las redes de difusión, se tiene una consideración adicional: cómo controlar el acceso al canal compartido. De esto se encarga una subcapa especial de la capa de enlace de datos, la **subcapa de control de acceso al medio**.

## 3) Capa de red:

La capa de red controla la operación de la subred. Se encarga de **enrutar** paquetes desde el origen al destino.

## 4) Capa de transporte:

La capa de transporte tiene como función dividir los datos provenientes de la capa superior en unidades más pequeñas si es necesario, pasárlas a la capa de red y asegurar que todas lleguen al otro extremo correctamente.

## 5) Capa de sesión:

La capa de sesión permite establecer sesiones entre distintos dispositivos. Las sesiones ofrecen diversos servicios, varios ligados a la sincronización (evitar race conditions).

## 6) Capa de presentación:

Estandariza la representación interna de datos de tal forma que computadoras con diferentes arquitecturas puedan comunicarse. Permite además transmitir estructuras de datos abstractas como datos de cuenta bancaria, por ejemplo.

## 7) Capa de aplicación:

La capa de aplicación contiene una variedad de protocolos utilizados por los usuarios. Un protocolo de aplicación muy utilizado es HTTP (HyperText Transfer Protocol), que forma la base para la World Wide Web. Hay otros protocolos de aplicación que se utilizan para transferir archivos, enviar y recibir correo electrónico y noticias.

### 1.4.2. Modelo TCP/IP

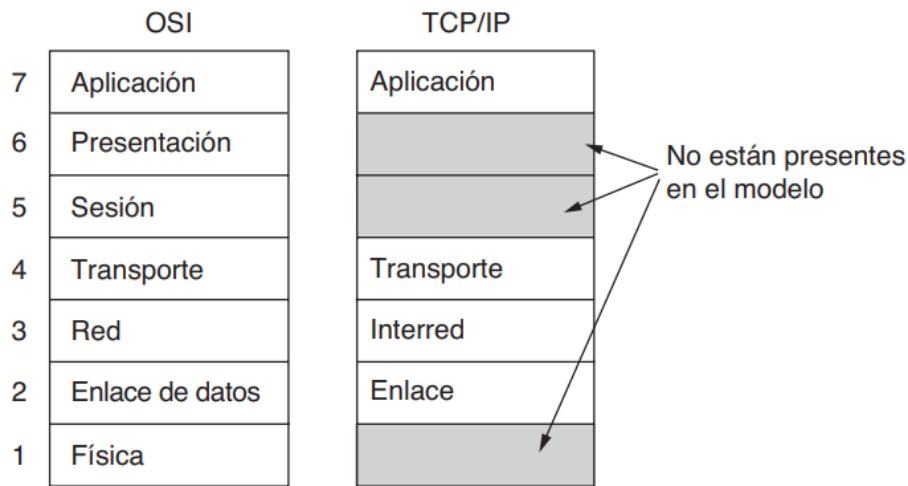


Figura 4: Modelo de referencia TCP/IP.<sup>1</sup>

El **modelo TCP/IP** fue financiado por el Departamento de Defensa de EEUU. Su objetivo principal era que la comunicación se pudiera mantener incluso si algún nodo intermedio a la comunicación fallara (por algún ataque).

#### 1) Capa de enlace

La capa de enlace describe qué enlaces (como las líneas seriales y Ethernet clásica) se deben llevar a cabo para cumplir con las necesidades de la capa de interred. En realidad no es una capa en el sentido común del término, sino una interfaz entre los hosts y los enlaces de transmisión.

#### 2) Capa de interred:

La capa de interred permite a los hosts enviar paquetes en una red hacia un destino. Define protocolos oficiales **IP (Internet Protocol)** e **ICMP (Internet Control Message Protocol)**.

La tarea de la capa de interred es el ruteo paquetes IP.

#### 3) Capa de transporte:

Tiene 2 protocolos de transporte extremo a extremo:

- **TCP (Transport Control Protocol)**: Es un protocolo confiable, orientado a la conexión, que mantiene el orden y se encarga del control de flujo (para que un emisor veloz no saturé a un receptor lento).
- **UDP (User Datagram Protocol)**: Es un protocolo no confiable y que delega el control de flujo a las aplicaciones. Es también usado para consultas de tipo solicitud-respuesta.

#### 4) Capa de aplicación:



En TCP/IP, la capa de aplicación absorbe a las de sesión y presentación en el modelo OSI, ya que estas no eran muy utilizadas.

#### 1.4.3. Diferencias entre OSI y TCP/IP

Los modelos de referencia OSI y TCP/IP tienen mucho en común. Ambos se basan en el concepto de una pila de protocolos independientes. Además, la funcionalidad de las capas es muy similar.

Una diferencia obvia entre los dos modelos está en el número de capas: El modelo OSI tiene siete capas, mientras que el modelo TCP / IP tiene cuatro.

El modelo OSI define tres conceptos básicos: **servicio**, **interfaz** y **protocolo**, que definimos más arriba.

El modelo TCP/IP no tiene una distinción tan clara entre los servicios, las interfaces y los protocolos. Por ejemplo, los únicos servicios que realmente ofrece la capa de interred son send ip packet y receive ip packet. Esto hace que los protocolos del modelo OSI estén mejor ocultos, y a su vez sean más fáciles de modificar.

Otra diferencia es que el modelo OSI soporta comunicación orientada a la conexión como no orientada a la conexión en la capa de red, y sólo soporta comunicación orientada a conexión en la capa de transporte. Por otro lado, en TCP/IP, en la capa de red sólo se soporta el modo sin conexión, pero ambos en la capa de transporte, lo cual tiene sus ventajas.

## 2. Capa Física

La capa física es la capa más baja del modelo OSI. Esta define las interfaces eléctricas, de temporización y demás interfaces mediante las cuales se envían los bits como señales a través de los canales.

### 2.1. Bases teóricas

Mediante la variación de alguna propiedad física, como el voltaje o la corriente, es posible transmitir información a través de cables. Representamos el valor de este voltaje o corriente como una función del tiempo  $f(t)$ .

#### 2.1.1. Análisis de Fourier

Fourier desarrolló un mecanismo para expresar cualquier función periódica  $g(t)$  con período  $T$  se puede representar como una suma (potencialmente infinita) de senos y cosenos:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft)$$

Donde  $f = 1/T$  y  $a_n$  y  $b_n$  son las amplitudes de seno y coseno del  $n$ -ésimo armónico (término) y  $c$  es una constante. A esta descomposición se la llama serie de Fourier.

Luego, si conocemos el período  $T$  y las amplitudes, podemos hallar la función del tiempo  $g(t)$  a través de esta suma. Es posible calcular  $a_n$ ,  $b_n$ , y  $c$ , obteniendo los siguientes resultados:

$$a_n = \frac{2}{T} \int_0^T g(t) \sin(2\pi n f t) dt \quad b_n = \frac{2}{T} \int_0^T g(t) \cos(2\pi n f t) dt \quad c = \frac{2}{T} \int_0^T g(t) dt$$

### Señales de ancho de banda limitado

La relevancia de series de Fourier para la comunicación de datos es que los canales reales afectan a cada señal de frecuencia de una manera diferente.

Consideremos un ejemplo: La transmisión del carácter ASCII “b” codificado en un byte. El patrón de bits transmitido será 01100010.

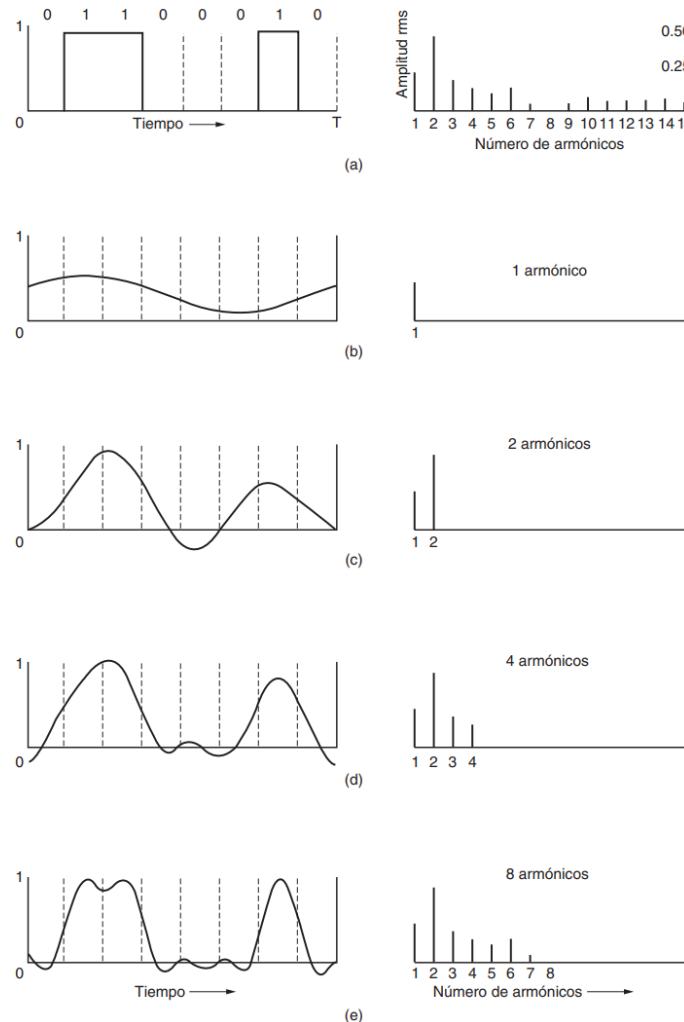


Figura 5: (a) Una señal binaria y sus amplitudes de raíz cuadrada media de Fourier. (b) a (e) Aproximaciones sucesivas a la señal original.<sup>1</sup>



La parte izquierda de la figura 5(a) muestra la salida de voltaje producido por la computadora transmisora. El análisis de Fourier de esta señal produce los siguientes coeficientes:

$$a_n = \frac{1}{\pi n} [\cos(\pi n/4) - \cos(3\pi n/4) + \cos(6\pi n/4) - \cos(7\pi n/4)]$$

$$a_n = \frac{1}{\pi n} [\sin(3\pi n/4) - \sin(\pi n/4) + \cos(7\pi n/4) - \cos(6\pi n/4)]$$

$$c_n = 3/4$$

A su derecha, se muestran las amplitudes de raíz cuadrada media,  $\sqrt{a_n^2 + b_n^2}$ , para los primeros términos. Estos valores son de interés debido a que sus cuadrados son proporcionales a la energía que se transmite en la frecuencia correspondiente.

Los transmisores siempre perderán potencia cuando envían señales, pero no todas las componentes de la serie de Fourier se disminuyen en la misma cantidad, por lo que se introduce **distorsión**.

En un cable, la mayoría de amplitudes se transmiten sin ninguna disminución. El rango de frecuencia que se transmite sin disminuirse “considerablemente” se llama **ancho de banda**, y suele considerarse desde 0 hasta la frecuencia que perdió la mitad de potencia. El ancho de banda es una propiedad física del medio de transmisión, que depende del grosor, longitud, etc... del cable o fibra óptica.

**La información que se puede transportar depende sólo del ancho de banda**, y no de su frecuencia inicial ni final.

- Las señales que van desde cero hasta una frecuencia máxima se llaman señales de **banda base**.
- Las que se desplazan para ocupar un rango de frecuencias más altas, como es el caso de todas las transmisiones inalámbricas, se llaman señales de **pasa-banda**.

Volviendo al ejemplo: La figura 5(b) muestra la señal que resulta de un canal que sólo permite el paso del primer armónico (la fundamental, f). Las figuras 5(c)-(e) muestran los espectros y las funciones reconstruidas para canales de mayor ancho de banda. Se busca enviar una señal con la mínima fidelidad necesaria para reconocer la secuencia de bits que se envió. Como en la figura 5(e) ya es posible, sería un desperdicio enviar una señal más precisa.

Si Tenemos un mensaje de b bits/seg, tardaríamos  $n/b$  segundos en enviar n bits. Entonces, la frecuencia del primer armónico de este mensaje será de  $b/n$  Hz.

Volviendo al ejemplo: Si tenemos que enviar estos 8 bits a una velocidad de 9600 bps, en una línea con calidad de voz, (con frecuencia de 3000 Hz), vemos en la tabla que transformará la figura 2-1(a) en algo parecido a la figura 2-1(c), lo cual dificultará la recepción precisa del flujo de bits original.

Bps	T (mseg)	Primer armónico (Hz)	Núm. de armónico transmitidos
300	26.67	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

Figura 6: Relación entre la tasa de datos y los armónicos para el ejemplo.<sup>1</sup>

### 2.1.2. Tasa de datos máxima de un canal

Se puede demostrar que una señal pasada por un filtro pasa-bajas con ancho de banda  $B$  puede ser reconstruida por completo tomando  $2B$  muestras (exactas, sin ruido) por segundo. En la práctica, tampoco será necesario tomar  $2B$  muestras por segundo, pues los componentes de mayor frecuencia ya se filtraron. Si la señal consiste en  $V$  niveles discretos, el **teorema de Nyquist** establece lo siguiente:

$$\text{Tasa\_de\_datos\_maxima} = 2B \times \log_2(V \times \text{bits/seg})$$

Por lo tanto, un canal sin ruido de 3kHz no puede transmitir señales binarias (de dos niveles) a una velocidad mayor de 6000 bps.

### Ruido

Ahora bien, en la vida real, los canales siempre tendrán ruido (ya sólo por el ruido térmico, si no estamos a 0 kelvin).

La cantidad de ruido se mide con la relación de su potencia con la de la señal, llamada **SNR** (**Signal-to-Noise Ratio**):  $S/N$ .

Se representa en una escala logarítmica llamada **decibéles (dB)**:  $10\log_{10}S/N$ . Luego, una relación S/N de 10 es igual a 10 dB, una relación de 100 es igual a 20 dB, una relación de 1000 es igual a 30 dB y así...

El principal resultado de Shannon es que la tasa de datos máxima (o capacidad) de un canal ruidoso, con ancho de banda  $B$  Hz y relación señal a ruido  $S/N$ , está dada por:

$$\text{Número\_máximo\_de\_bits/seg} = B \times \log_2(1 + S/N)$$

Ejemplo: La ADSL (Línea Asimétrica de Suscriptor Digital), que provee Internet a través de líneas telefónicas, utiliza un ancho de banda de aproximadamente 1 MHz. Suponiendo una SNR de alrededor

de 40 dB (que depende de la longitud del cable, 40 dB es bueno para 1 a 2 km), el canal no podrá transmitir a más de 13 Mbps, sin importar cuántos niveles de señal se utilicen ni con qué frecuencia se tomen las muestras. En la práctica, el servicio ADSL se especifica hasta 12 Mbps, y los usuarios suelen ver tasas más bajas.

## 2.2. Medios de transmisión guiados

La capa física tiene la función de transportar bits entre dispositivos. Existen varios medios para lograr este propósito, cada uno con un ancho de banda, retardo, coste y mantenimiento diferente.

### 2.2.1. Medios magnéticos

Una forma de trasladar datos es utilizando medios removibles, como DVDs o una cinta magnética. Estos pueden no ser muy prácticos, pero son baratos.

### 2.2.2. Par Trenzado

Uno de los medios más antiguos y aún de los más populares es el **par trenzado**. Este consta de dos cables de cobre aislados, y trenzados en forma helicoidal.

El trenzado se debe a que dos cables paralelos constituyen una antena simple. Cuando se trenzan, las ondas se cancelan y el cable irradia con menos efectividad.

Una señal se transmite como la diferencia en el voltaje entre los dos cables. Esto aumenta la inmunidad al ruido externo, pues tiende a afectar ambos cables en la misma proporción, dejando la diferencia intacta.

Adicionalmente, cuando hay muchos pares trenzados paralelos, se suelen trenzar nuevamente y cubrir con otra funda protectora. De otra forma, estos podrían todavía interferir uno con el otro.

Existen diversos tipos de cableado de par trenzado. Uno muy utilizado en edificios se llama cable de categoría 5, o “cat 5”. Este consta de dos cables aislados que se trenzan de manera delicada. Por lo general se agrupan cuatro de esos pares en una funda de plástico para protegerlos y mantenerlos juntos.

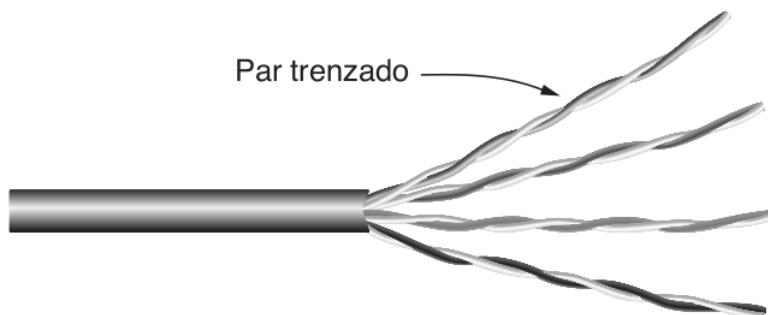


Figura 7: Cable UTP5 con 4 pares trenzados.<sup>1</sup>

## Tipos de enlaces

Los enlaces que se pueden utilizar en ambas direcciones al mismo tiempo se llaman enlaces **full-dúplex**; los que se pueden utilizar en cualquier dirección, pero sólo una a la vez, se llaman enlaces **half-dúplex**; y los que sólo permiten tráfico en una dirección se conocen como enlaces **simplex**.

### 2.2.3. Coax

El cable coaxial (o “coax”) es un cable con mejor blindaje y ancho de banda que los pares trenzados sin blindaje, lo que le permite abarcar mayores distancias a velocidades más altas, y con alta protección contra el ruido. Estos cables tienen un ancho de banda de hasta varios GHz.

Un cable coaxial consiste en alambre de cobre rígido como núcleo, rodeado por un material aislante. El aislante está forrado de un conductor cilíndrico, que por lo general es una malla de tejido fuertemente trenzado. El conductor externo está cubierto con una funda protectora de plástico.

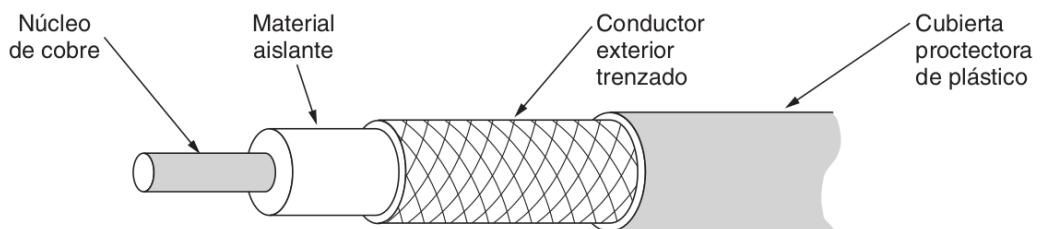


Figura 8: Cable coaxial.<sup>1</sup>

### 2.2.4. Líneas eléctricas

Es posible utilizar el mismo cableado de las líneas eléctricas para transmitir información. Esta tiene una clara ventaja en conveniencia, pues es posible simplemente conectar un televisor, cosa que se haría de todas formas, y por encima de la señal eléctrica de baja frecuencia, se envían datos a la vez que energía. La desventaja es que estos cableados no se construyeron para la comunicación, y atenúan las señales de frecuencias altas (MHz) que se necesitan para la comunicación de datos de alta velocidad. Además, el no estar trenzados incrementa el ruido. Sin embargo, es posible enviar más de 100Mbps a través de una línea eléctrica.

### 2.2.5. Fibra óptica

La fibra óptica es una tecnología capaz de alcanzar un ancho de banda mayor a 50Tbps, con un ruido mínimo, y un coste menor a los cables de cobre.

Un sistema de transmisión óptico tiene tres componentes clave:

- **La fuente de luz:** Donde un pulso de luz indica un 1 y la ausencia de luz indica un 0.
- **El medio de transmisión:** Será una fibra de vidrio ultradelgada.

- **El detector:** Genera un pulso eléctrico cuando la luz incide en él.

Sin embargo, este sistema de transmisión teóricamente tendría fugas de luz, y sería inútil. Ahora bien, por un principio físico, fue posible encontrar un ángulo en el cual la refracción de la luz dentro del cable es absoluta. Así, un rayo de luz con un ángulo igual o mayor al ángulo crítico queda atrapado dentro de la fibra, y se puede propagar por muchos kilómetros prácticamente sin pérdidas.

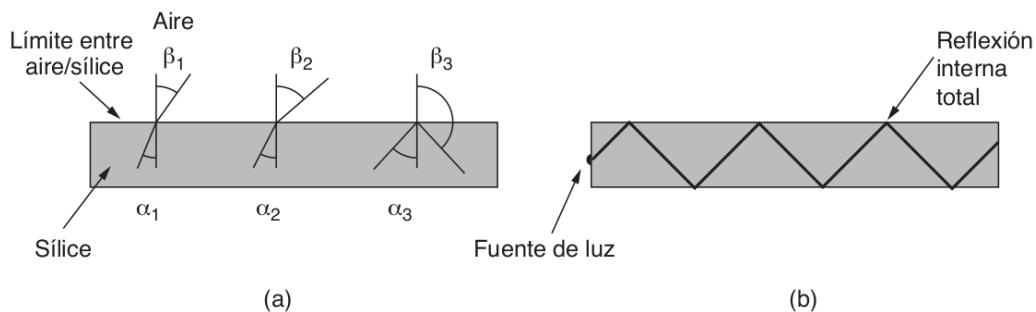


Figura 9: (a) Tres ejemplos de un rayo de luz desde el interior de una fibra de sílice que incide sobre el límite entre aire y sílice a distintos ángulos. (b) Luz atrapada por reflexión interna total.<sup>1</sup>

### Fibras multimodal y monomodo

En un cable de fibra óptica, habrá muchos rayos distintos rebotando con ángulos diferentes. Se dice que cada rayo tiene un **modo** distinto, por lo que a una fibra con esta propiedad se la llama **fibra multimodal**.

Ahora bien, si el diámetro de la fibra se reduce a unas cuantas longitudes de onda de luz, la fibra actúa como una guía de ondas y la luz se puede propagar sólo en línea recta, sin rebotar, con lo que se obtiene una **fibra monomodo**. Estas son más costosas, pero se utilizan mucho para distancias más largas: pueden transmitir datos a 100 Gbps por 100 km sin necesidad de amplificación.

### Transmisión de Luz a través de fibras

Las fibras ópticas están hechas un vidrio ultratransparente para minimizar la atenuación. La atenuación de la luz que pasa por el vidrio depende también de la longitud de onda de la luz:

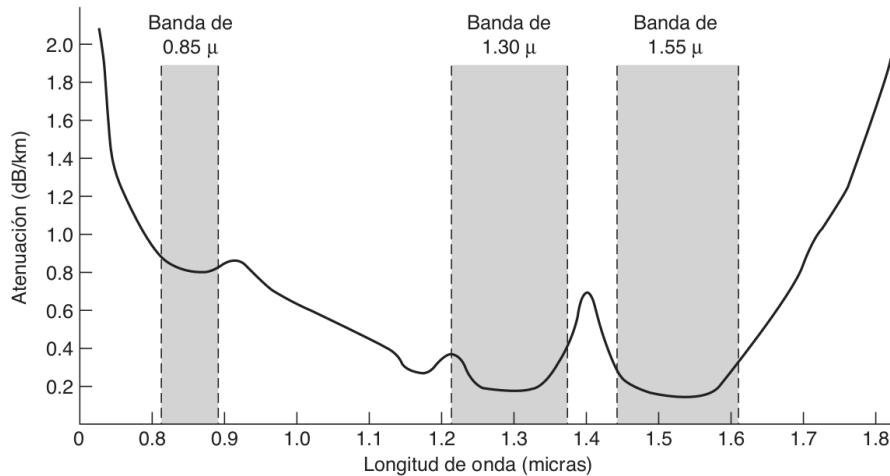


Figura 10: Atenuación de la luz dentro de una fibra en la región de infrarrojo.<sup>1</sup>

Existe un fenómeno llamado **dispersión cromática**, que hace que la longitud de los pulsos de luz enviados a través de fibra óptica se alarguen con la distancia. Para evitar que esto sea un problema, se pueden utilizar pulsos especiales llamados **sólidos** (que tienen algo q ver con enviarlos en una forma relacionada con el recíproco del coseno hiperbólico, no se dice más que esto en el libro).

### Cables de fibra

Los cables de las fibras ópticas llevan varias fibras ópticas en una funda protectora.

Un cable individual de fibra tendrá un núcleo de vidrio en el centro, rodeado de un revestimiento de vidrio, y con una cubierta de plástico. En las fibras multimodales, el núcleo tiene 50 micras de diámetro, y en las monomodo, el núcleo es de 8 a 10 micras.

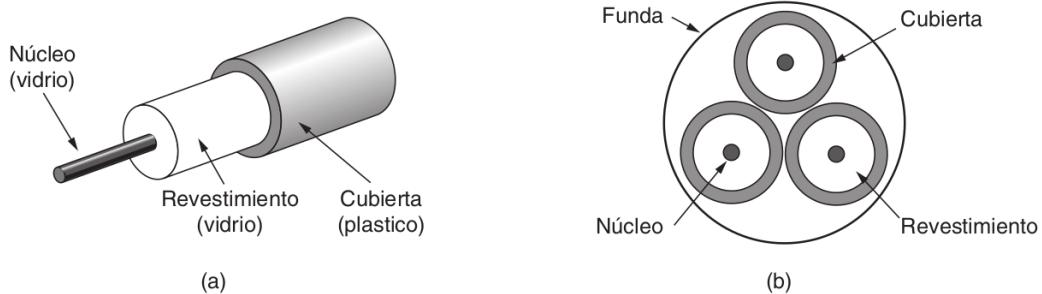


Figura 11: (a) Vista lateral de una sola fibra. (b) Vista de extremo de una envoltura con tres fibras.<sup>1</sup>

El revestimiento de vidrio tiene como fin mantener la luz dentro del núcleo, y lo logra teniendo un índice de refracción más bajo.

Por lo general se utilizan dos tipos de fuentes de luz para producir las señales: **LED** (Light Emitting Diodes) y **lásers semiconductores**, teniendo los segundos una mayor tasa de datos y alcance, pero siendo de un mayor coste y teniendo una mayor sensibilidad a la temperatura y un menor tiempo de vida.



El extremo receptor de una fibra óptica consiste en un fotodiodo, el cual emite un pulso eléctrico cuando lo golpea la luz. Su tiempo de respuesta es de 100 Gbps, lo cual limita la tasa de datos. El ruido térmico es otro inconveniente, por lo que un pulso de luz debe llevar suficiente potencia para detectarlo.

Adicionalmente, cabe destacar que los cables de fibra óptica son unidireccionales, por lo que *se necesitan siempre 2 fibras individuales para crear un canal bidireccional*.

## **2.3. Medios de Transmisión inalámbricos**

### **2.3.1. El espectro electromagnético**

Cuando los electrones se mueven, crean ondas electromagnéticas que se pueden propagar por el espacio (incluso en el vacío). El número de oscilaciones por segundo de una onda es su frecuencia,  $f$ , y se mide en Hz.

La distancia entre dos máximos (o mínimos) consecutivos,  $\lambda$ , se llama longitud de onda. La comunicación inalámbrica se basa en conectar una antena a un circuito eléctrico para que transmita información en la forma de ondas electromagnéticas, que podrán ser captadas hasta cierta distancia.

En el vacío, todas las ondas electromagnéticas se mueven a la misma velocidad: la velocidad de la luz. En el cobre o la fibra, la velocidad baja a  $2/3$  y depende de la frecuencia. La relación fundamental entre  $f$ ,  $\lambda$  y  $c$  (en el vacío) es

$$\lambda f = c$$

Dado que  $c$  es una constante, si conocemos el valor  $f$  podemos encontrar  $\lambda$  y viceversa.

Como regla práctica, cuando  $\lambda$  se da en metros y  $f$  en MHz,  $\lambda f \approx 300$ . Por ejemplo, las ondas de 100 MHz tienen una longitud aproximada de 3 metros, las ondas de 1000 MHz tienen una longitud de 0.3 metros y las ondas de 0.1 metros tienen una frecuencia de 3000 MHz.

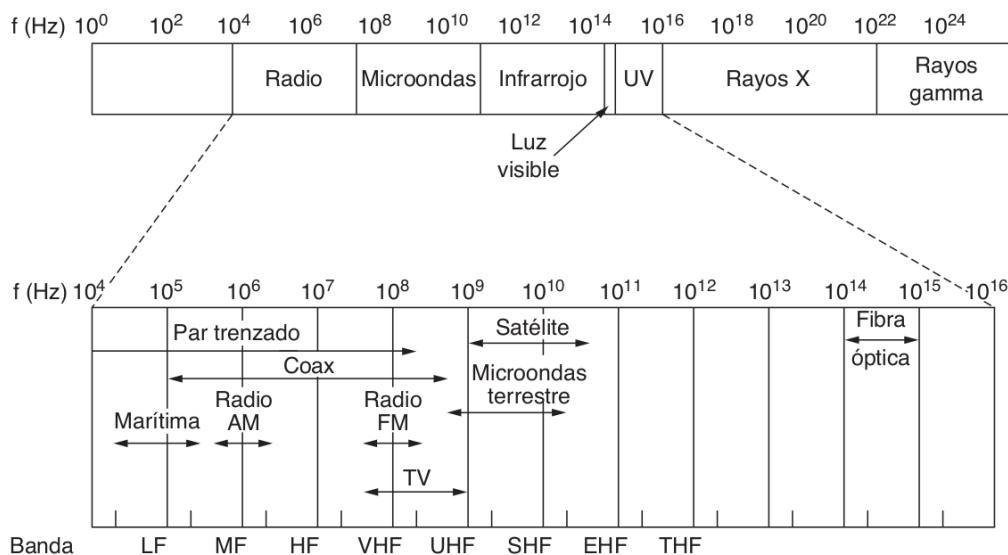


Figura 12: El espectro electromagnético y sus usos para comunicaciones.<sup>1</sup>

Acá en el libro se ven temas de *espectro disperso con salto de frecuencia (FHSS)* y de *secuencia directa (DSSS)*, pero se ve mejor en la parte de WiFi, por lo que no lo añado. Ver en esa sección.

## Radiotransmisión

Las ondas de **radio frecuencia** (RF) son fáciles de generar, pueden recorrer distancias largas y penetrar edificios con facilidad y son omnidireccionales, lo cual los hace muy populares. Las propiedades de las ondas de radio dependen de su frecuencia:

- A bajas frecuencias, cruzan bien los obstáculos, pero pierden potencia a medida que se alejan.
- A frecuencias altas, las ondas de radio tienden a viajar en línea recta y rebotan en los obstáculos, pero pierden aún más potencia por la distancia.

Las ondas de radio suelen tener un gran rango, por lo tanto, un problema mayor suele ser la interferencia.

En las bandas VLF, LF y MF las ondas de radio siguen la curvatura de la Tierra. La difusión de radio AM utiliza la banda MF.

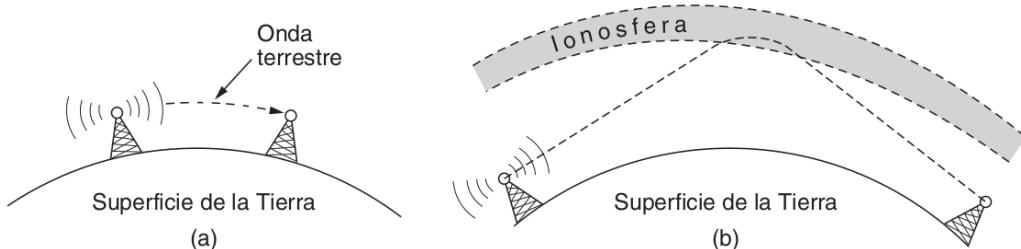


Figura 13: (a) En las bandas VLF, LF y MF, las ondas de radio siguen la curvatura de la Tierra. (b) En la banda HF, rebotan en la ionosfera.<sup>1</sup>

En las bandas HF y VHF, las ondas que chocan con la tierra son absorbidas, pero las ondas que llegan a la ionosfera se refractan y pueden ser recibidas.

### Transmisión por microondas

Al transmitir por encima de los 100 MHz, las ondas viajan en línea recta y en consecuencia, se pueden enfocar en un haz estrecho. Esto permite obtener una mayor relación señal-ruido, y que varios transmisores en fila se comuniquen sin interferencia. Pero requiere que la antena transmisora y receptora estén alineadas.

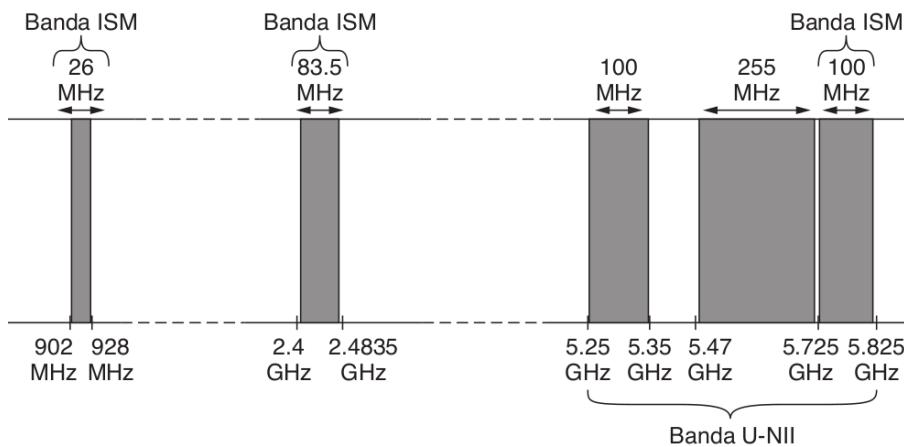
Al moverse en línea recta, la tierra puede bloquear señales entre dos antenas de microondas, por lo que estas suelen ser muy altas, y se utilizan repetidores.

Como agravante adicional, estas ondas no son buenas para atravesar edificios, y hasta pueden refractarse en partículas de la atmósfera, retrasando su transmisión. Estas ondas retrasadas pueden llegar desfasadas con la onda directa y cancelar así la señal. A este efecto se le llama **desvanecimiento por multirayos**.

### La política del espectro electromagnético

Todas las empresas quieren tener una porción del espectro grande. Sin embargo, esto llevaría a una gran cantidad de interferencia y el caos no permitiría a nadie utilizarlo de manera eficiente. Para solucionarlo, se vendieron ciertos rangos del espectro a los mayores postores.

Por otro lado, hay ciertas frecuencias que pueden ser utilizadas libremente, como las U-NII de 5 GHz, o las ISM, usadas para cocheras inalámbricas, etc...

Figura 14: <sup>1</sup>

### 2.3.2. Transmisión infrarroja

Las ondas infrarrojas no guiadas son populares en la comunicación de corto alcance: la utilizan el control remoto de los televisores, aires acondicionados, etc...

Son relativamente direccionales, económicos y fáciles de construir, pero tienen el problema de que no atraviesan objetos sólidos. Esto por otro lado tiene la ventaja de que no genera interferencia y es más seguro contra el espionaje.

### 2.3.3. Transmisión por ondas de luz

Con la imagen queda claro qué es y cómo funciona, y hasta sus problemas y posibles ventajas.

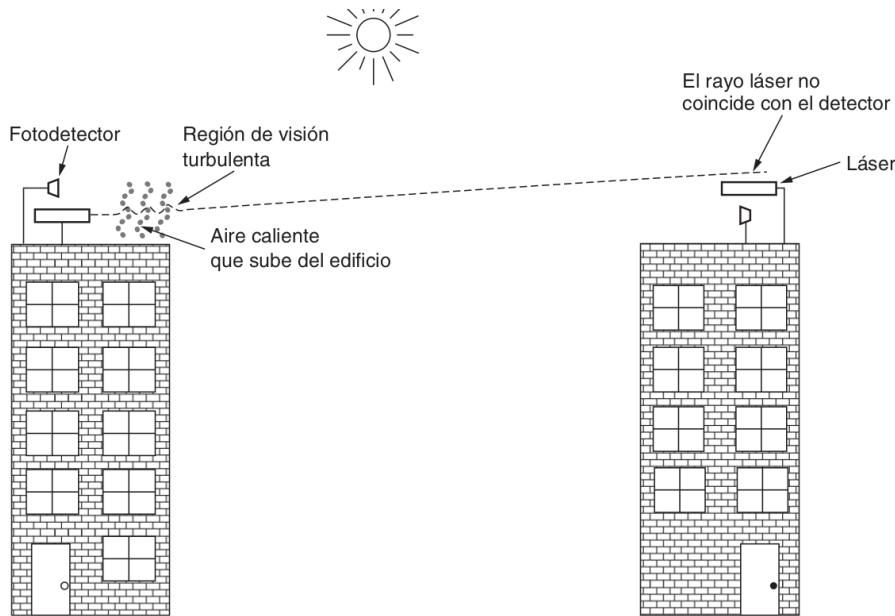


Figura 15: La señalización óptica sin guías, también conocida como óptica de espacio libre. Las corrientes de convección pueden interferir con los sistemas de comunicación por láser. Aquí se ilustra un sistema bidireccional con dos láser.<sup>1</sup>

#### 2.3.4. Satélites de comunicación

Un satélite de comunicación tiene como funcionalidad principal servir como un **tubo doblado**: un repetidor de microondas con varios transportadores, cada uno de los cuales recibe, amplifica y retransmite las señales que recibe, en otra frecuencia, de forma tal que no interfiera con las entrantes. Pero además, puede hacer un procesamiento que manipule o redirija por separado los flujos de datos en toda la banda, mejorando así el desempeño del tubo doblado.

##### ¿Dónde se ubican los satélites?

Mientras más alto esté un satélite, más tardará en orbitar la tierra (su período será más largo). Además, existen áreas de la atmósfera donde las partículas están sobrecargadas y destruirían un satélite. Por lo tanto, existen 3 regiones donde hoy se colocan satélites:

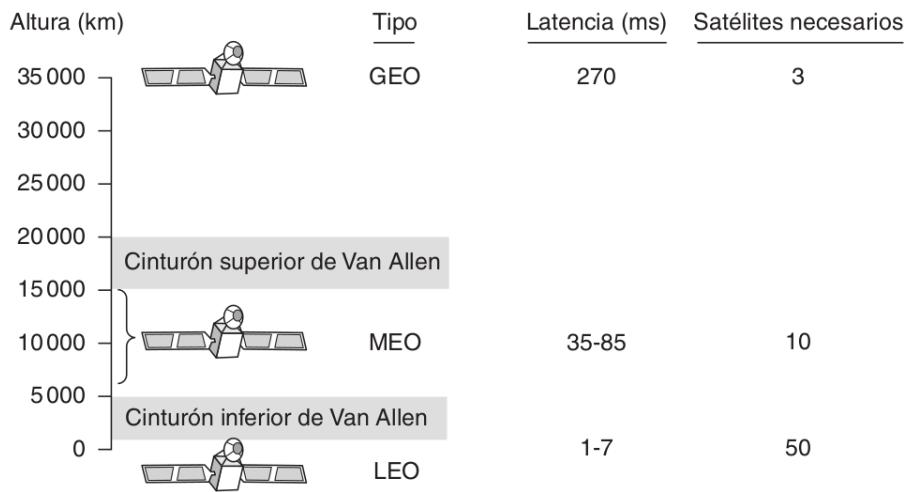


Figura 16: Satélites de comunicaciones y algunas de sus propiedades, incluyendo la altitud sobre la Tierra, el tiempo de retardo de viaje redondo y la cantidad de satélites necesarios para una cobertura global.<sup>1</sup>

### Satélites geoestacionarios

Si se coloca un satélite a una altura específica sobre el ecuador, este parecería estar inmóvil en el cielo, por lo que no se tendría que rastrear. Estos satélites son llamados **GEO** (Geostationary Earth Orbit).

Ahora bien, existe una cantidad limitada de espacio en esta franja sobre el ecuador. Por lo tanto, la ITU se encarga de la asignación de espacio orbital. También será la encargada de designar qué bandas de frecuencias utilizarán los satélites, cada una con sus ventajas y desventajas.

Banda	Enlace descendente	Enlace ascendente	Ancho de banda	Problemas
L	1.5 GHz	1.6 GHz	15 MHz	Bajo ancho de banda; saturada.
S	1.9 GHz	2.2 GHz	70 MHz	Bajo ancho de banda; saturada.
C	4.0 GHz	6.0 GHz	500 MHz	Interferencia terrestre.
Ku	11 GHz	14 GHz	500 MHz	Lluvia.
Ka	20 GHz	30 GHz	3500 MHz	Lluvia, costo del equipo.

Figura 17: Las principales bandas de satélites.<sup>1</sup>

Los satélites GEO tienen antenas y la capacidad de enfocar sus haces a áreas geográficas pequeñas y grandes, permitiendo, por ejemplo, que un satélite de comunicación para Estados Unidos tenga un haz amplio para los 48 estados contiguos, además de haces puntuales para Alaska y Hawái.

**Satélites VSAT** Más recientemente, se comenzaron a utilizar microestaciones **VSAT** (Very Small Aperture Terminals). Estas tienen múltiples antenas pequeñas de poca potencia. Al tener esta potencia limitada, para comunicarse a través de estos, se añade un hub terrestre como intermediario, lo que tiene la desventaja de duplicar el retraso del mensaje. Este recibe y replica la señal enviándola al destino final.

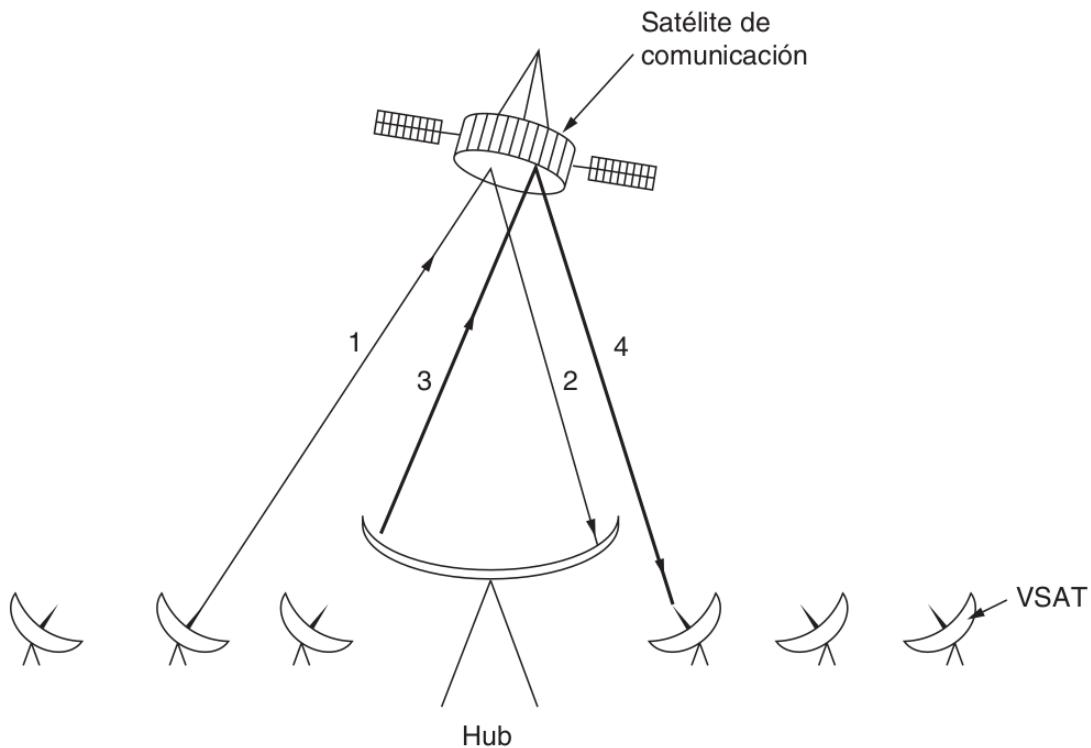


Figura 18: Terminales VSAT utilizando una estación central o hub.<sup>1</sup>

## 2.4. Modulación digital

Para enviar información, es necesario entonces representar bits, que son discretos, a través de señales, que son continuas.

Al proceso de transformar bits en señales y viceversa se conoce como **modulación digital**.

### 2.4.1. Transmisión en banda base

La forma más simple de modulación digital es simplemente representar un 1 con un voltaje positivo, y un 0 con un voltaje negativo, o en el caso de la fibra óptica, la luz un 1, y su ausencia un 0. Este esquema se denomina **NRZ** (Non-Return-to-Zero, el nombre extraño es por cuestiones históricas).

Ahora bien, la señal enviada por un lado del cable no será la misma que se recibirá del otro, producto del ruido y la atenuación del canal utilizado, por lo tanto, para decodificar una señal, se asocia con el valor más cercano.

NRZ es un esquema simple, sin embargo, no es muy utilizado en la práctica por problemas en la ingeniería. En vez, se suele utilizar como una base y combinarlo con otros esquemas, que se llaman **códigos de línea**. Los siguientes esquemas ayudan con la eficiencia del ancho de banda, la recuperación del reloj y el balance de CD.

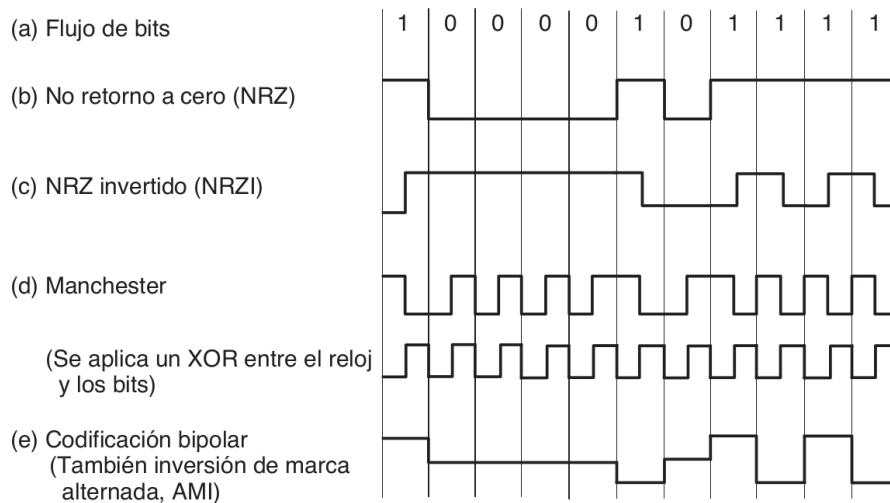


Figura 19: Códigos de línea: (a) Bits, (b) NRZ, (c) NRZI, (d) Manchester, (e) Bipolar o AMI..<sup>1</sup>

### Eficiencia del ancho de banda, tasa de bits y de símbolo

Con NRZ, la señal puede alternar entre los niveles positivo y negativo hasta cada 2 bits (en caso de alternar 1 s y 0 s). Esto significa que necesitamos un ancho de banda de por lo menos  $B/2$  Hz para representar  $B$  bits/seg.

Para aprovechar el ancho de banda de forma más eficiente, se pueden usar más de dos niveles de señalización: Si por ejemplo usamos 4 voltajes, podemos enviar 2 bits como un sólo símbolo. Sin embargo, esto sólo funcionará cuando la señal en el receptor sea suficientemente fuerte como para diferenciar los 4 niveles, pues al dividirlo en más niveles, el ruido es relativamente mayor con respecto a la diferencia entre estos niveles.

De esta forma, la tasa de bits se divide, y por lo tanto el ancho de banda necesario es reducido.

La tasa a la que cambia la señal se denomina **tasa de símbolo** para diferenciarla de la tasa de bits. Entonces, la **tasa de bits** es la tasa de símbolo multiplicada por el número de bits por símbolo.

Hay que tener en cuenta que el número de niveles de la señal no necesita ser una potencia de dos. A menudo no lo es, ya que algunos de los niveles se utilizan como protección contra errores y para simplificar el diseño del receptor.

### Recuperación de reloj

Un problema que surge del esquema NRZ es que el receptor debe conocer bien dónde termina cada símbolo. De otra forma, si recibe 16 0s seguidos, a menos de que tenga un reloj muy preciso (y demasiado costoso para la tarea), será muy difícil diferenciarlos de 15 0s.

Una estrategia es utilizar una línea adicional en los cables que envíe señales de reloj, pero esto sería un desperdicio y se podría usar para enviar más datos. El esquema utilizado en la Ethernet clásica, **Manchester**, consiste en mezclar las señales de reloj con las de datos aplicando un XOR a ambas (como se ve en la figura (d) que está arriba). Así se podrá siempre diferenciar entre bits viendo si se



produce una transición de voltaje de nivel alto a bajo o viceversa. Ahora bien, el reloj realiza una transición en cada tiempo de bit, por lo que tiene que operar al doble de la tasa de bits, y requiere el doble de ancho de banda. Con el tiempo, esto se volvió inaceptable.

Los cables USB por su lado, utilizan **NZRI**, donde los 0 son no transición, y los 1s son una transición. Esto soluciona el problema si hay muchos 1s seguidos. Para solucionar el problema con los 0s, se utiliza un código muy conocido llamado **4B/5B**, donde se asocia a cada patrón de 4 bits uno de 5:

Datos (4B)	Palabra de código (5B)	Datos (4B)	Palabra de código (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

Figura 20: Codificación 4B/5B.<sup>1</sup>

Otra forma que se ha utilizado es una llamada aleatorización o **scrambling**. Se tomaba una secuencia pseudoaleatoria de 1s y 0s llamada **semilla**, y se hacía un XOR entre esta y los datos antes de enviarlos, reduciendo la probabilidad de que se encuentren demasiados 0s seguidos. Y se envía junto a los datos la semilla utilizada.

Sin embargo, a pesar de ser atractiva al no incrementar la carga ni requerir de mayor ancho de banda, esta no puede garantizar que no haya una secuencia de 0s grande. Además, resultó débil contra ataques de usuarios que envíen paquetes de forma maliciosa.

## Señales balanceadas

Cualquier señal que tenga la misma cantidad de voltaje positivo y negativo, se conoce como **señal balanceada**. Si enviamos una señal cuyo promedio no sea cero desperdiciaremos energía.

El balanceo también claramente ayuda con el reloj, ya que hay una mezcla de voltajes positivos y negativos. Además, ayuda a calibrar los receptores, ya que midiendo el promedio de la señal, podemos usarlo como un umbral de decisión para decodificar los símbolos, reduciendo los errores.

Una forma de balancear mensajes es representar los 1s con dos voltajes: +1V y -1V, y los 0s como 0V. Esto se llama **codificación bipolar**.

Otra forma es utilizando un código balanceado, como el código **8B/10B**, que también ayuda con la recuperación del reloj.



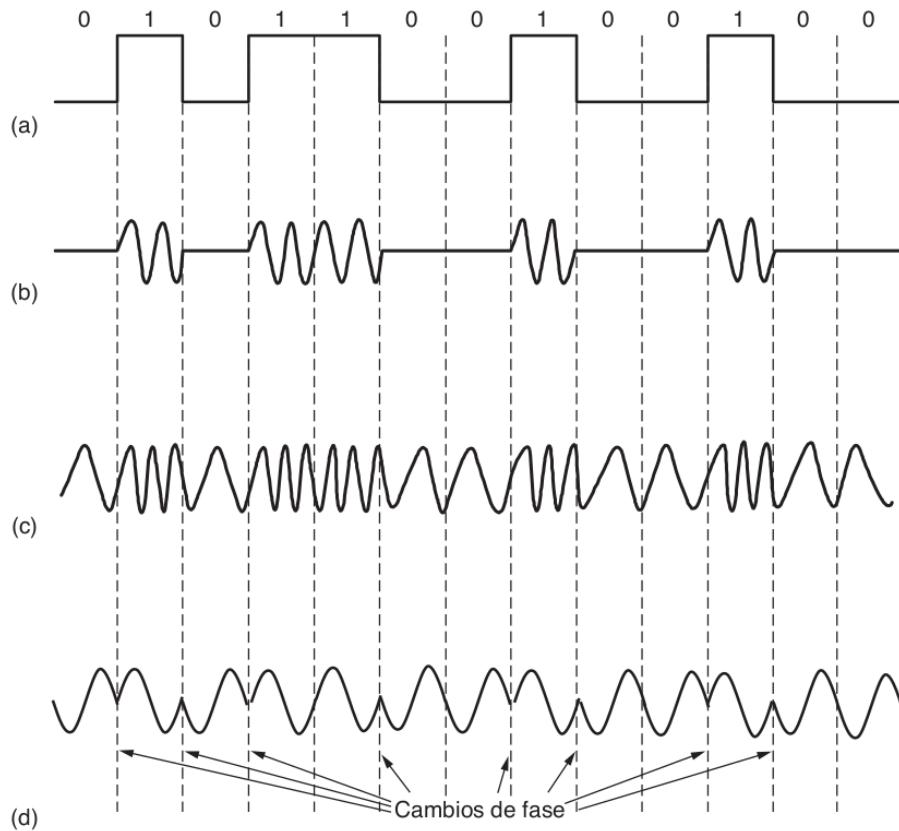
### 2.4.2. Transmisión pasa-banda

En general, al enviar señales es conveniente no utilizar un rango de frecuencias que comience en 0. Principalmente en medios inalámbricos, donde si todos utilizaran transmisión en banda base, que usa un rango comenzando en 0, posiblemente todo el ancho de banda, habría interferencia. Incluso en cables, puede convenir colocar una señal en una banda de frecuencias específica para que pueda coexistir con otras en el cable.

A este tipo de transmisión se la conoce como transmisión pasa-banda. Afortunadamente, es posible cambiar una señal de banda base, ocupando de 0 a N Hz, y desplazarla a una banda de paso de S a S+N Hz, sin perder información, y que el receptor simplemente la desplace de vuelta a su forma de banda base.

Hay tres características que podemos modificar en una señal para lograr esto:

- **ASK (Modulación por Desplazamiento de Amplitud / Amplitude Shift Keying):** Se utilizan amplitudes distintas para representar diferentes valores. En el ejemplo se utilizan dos amplitudes, que representan el 0 y 1, pero pueden usarse más para representar más valores.
- **FSK (Modulación por Desplazamiento de Frecuencia /Frequency Shift Keying):** Se utilizan dos o más tonos distintos. El ejemplo utiliza sólo dos frecuencias.
- **PSK (Modulación por Desplazamiento de Fase, del inglés Phase Shift Keying):** La onda portadora se desplaza una cierta cantidad de grados para representar los valores diferentes. En el ejemplo al tener dos, se usa BPSK (Binary PSK), y se desplaza 0 o 180 grados en cada periodo de símbolo. Para representar 2 bits de información en cada símbolo en un canal con suficiente ancho de banda, se puede utilizar un esquema con 4 desplazamientos (por ejemplo: 45, 135, 225 o 315 grados).



**Figura 2-22.** (a) Una señal binaria. (b) Modulación por desplazamiento de amplitud. (c) Modulación por desplazamiento de frecuencia. (d) Modulación por desplazamiento de fase.

Figura 21: (a) Una señal binaria. (b) Modulación por desplazamiento de amplitud. (c) Modulación por desplazamiento de frecuencia. (d) Modulación por desplazamiento de fase.<sup>1</sup>

Es posible combinar varios esquemas y usar varios niveles para representar más valores por símbolo:

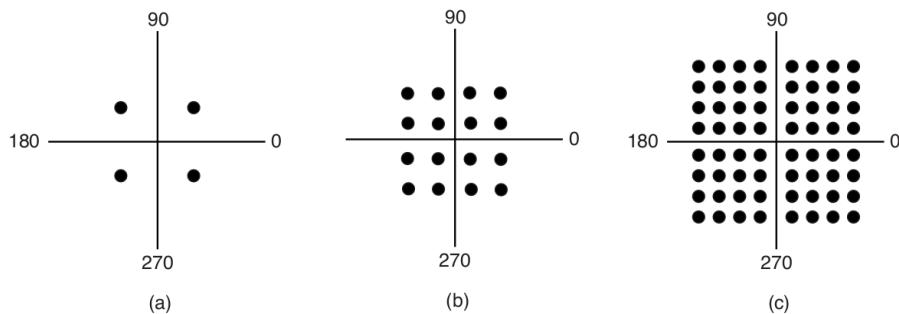


Figura 22: (a) QPSK. (b) QAM-16. (c) QAM-64.<sup>1</sup>

Aquí se muestran ejemplos de la combinación de Modulación por amplitud y fase, donde se representan puntos con mayor amplitud más lejos del centro, y su fase con su ángulo. A este tipo de diagramas se los llama **diagrama de constelación**.

Ahora bien, cuando se representan muchos valores con el mismo símbolo, además de tener una señal más propensa a errores de ruido, este ruido puede tener mayores repercusiones. Con el QAM-16 por ejemplo, si un símbolo representa al 0111 y su símbolo adyacente al 1000, si el receptor eligiera por error el símbolo adyacente, todos los bits estarían incorrectos. Una mejor solución es asociar bits con símbolos de manera que los símbolos adyacentes sólo difieran en 1 posición de bit, lo que es llamado **código Gray**:

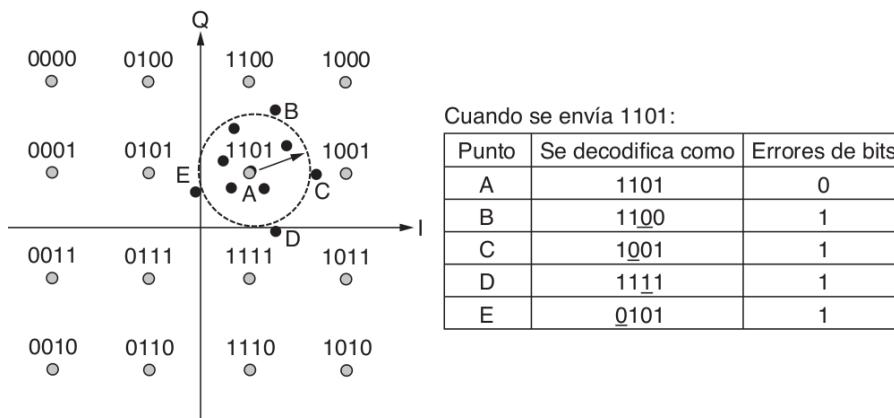


Figura 23: QAM-16 con código Gray.<sup>1</sup>

## 2.5. Multiplexión

### 2.5.1. Multiplexión por División de Frecuencia

La **FDM** (**Multiplexión por División de Frecuencia** / Frequency Division Multiplexing) aprovecha la transmisión pasa-banda para compartir un canal entre usuarios, dándole una **banda de frecuencias** a cada uno.

Cuando se multiplexa, se suelen dejar una **banda de guarda**, un exceso de banda ancha al necesario que mantiene bien separados los canales. Incluso entonces, estos pueden solaparse un poco, pero esto será simplemente detectado como ruido para los canales adyacentes.

En este ejemplo de 3 canales telefónicos, a cada canal de voz, los filtros le limitan el ancho de banda útil a cerca de 3100 Hz, y a cada uno se le asigna 4000 Hz, y tienen 900Hz de banda de guarda.

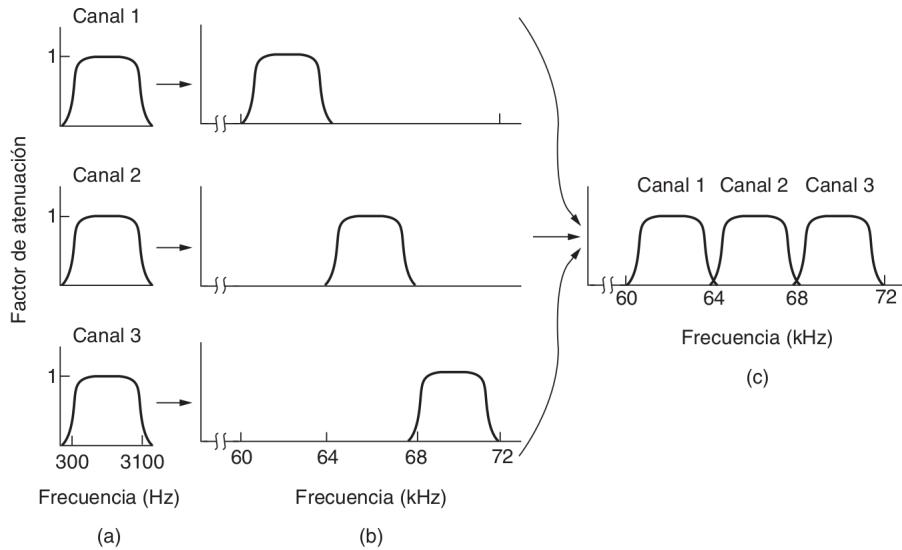


Figura 24: Multiplexión por división de frecuencia. (a) Los anchos de banda originales. (b) Los anchos de banda elevados en frecuencia. (c) El canal multiplexado.<sup>1</sup>

Hoy en día la FDM tiene un uso mucho menor, pues se prefiere multiplexar en el tiempo.

Ahora bien, existe una forma de evitar utilizar bandas de guardas, utilizando **OFDM (Multiplexión por División de Frecuencia Ortogonal)** / Orthogonal Frequency Division Multiplexing), que consiste en dividir el ancho de banda de forma tal que cada usuario envía datos en una manera en que su frecuencia es 0 en el centro de los demás usuarios. Así, se pueden muestrear los datos midiendo en el centro de la porción de la banda de un usuario.

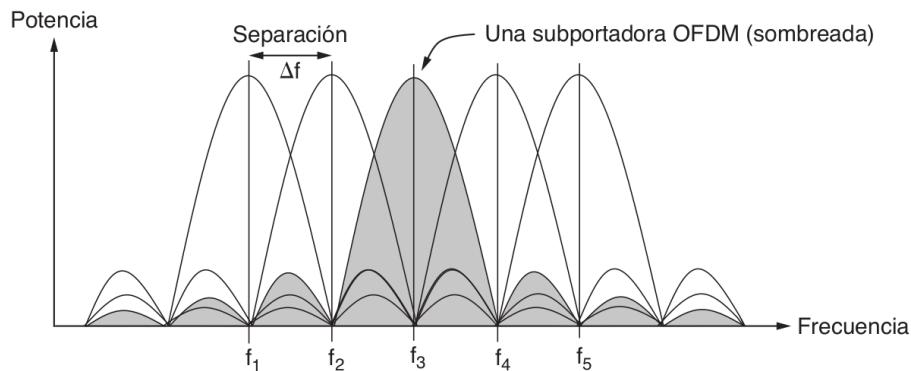


Figura 25: Multiplexión por División de Frecuencia Ortogonal (OFDM).<sup>1</sup>

OFDM es utilizado por redes 802.11 (WiFi), y es posible de implementar como una transformada de Fourier.

### 2.5.2. Multiplexión por División de Tiempo

La **TDM** (Multiplexión por División de Tiempo / Time Division Multiplexing) es una forma de multiplexión donde los usuarios reciben un turno (rotatorio tipo round-robin). Los turnos duran un tiempo fijo llamado ranura de tiempo, durante el cual reciben todo el ancho de banda de manera exclusiva, y se añade un tiempo de guarda entre ellos para diferenciarlos.

*Tanembaum 2.4.2, 2.4.3, 2.5.5, 2.6, 2.7 y 2.8 NO ENTRAN, y no están acá, pero se recomienda darles una leída.*

## 3. Capa de Enlace de Datos

La **capa de enlace de datos** se encarga de lograr una comunicación confiable y eficiente en unidades completas de información llamadas **tramas** (en vez de bits individuales como en la capa física), entre dos máquinas adyacentes.

### 3.1. Diseño de la Capa de Enlace de Datos

Esta capa utiliza los servicios de la capa física para enviar y recibir bits a través de los canales. Sus funciones son:

- 1) Proporcionar a la capa de red una interfaz de servicio bien definida.
- 2) Manejar los errores de transmisión.
- 3) Regular el flujo de datos para que los emisores rápidos no saturen a los receptores lentos.

La capa de enlace de datos toma los paquetes que recibe de la capa de red, y los **encapsula** en tramas para transmitirlos.

Las tramas tienen un **encabezado**, un campo de **carga útil**, y un **terminador/payload**.

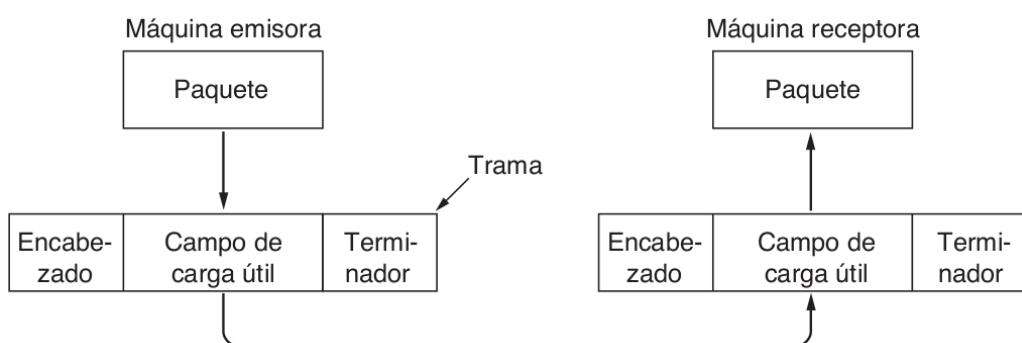


Figura 26: Relación entre paquetes y tramas.<sup>1</sup>



### 3.2. Servicios proporcionados a la capa de red

La función de la capa de enlace de datos es lograr llevar los datos desde la capa de red de una máquina hacia la capa de red de la máquina de destino. Vamos a obviar la función de la capa física y pensaremos que la capa de enlace de datos estará directamente conectada entre dos dispositivos.

Los servicios más comúnmente dados a la capa de red son:

- **Servicio sin conexión ni confirmación de recepción:** La trama se envía sin más. Si el paquete se pierde, no será notificado ni habrá un intento de recuperarlo. Suele utilizarse cuando la tasa de error es lo suficientemente baja, y se deja la detección de errores a capas superiores. Ethernet utiliza este modelo.
- **Servicio sin conexión con confirmación de recepción:** No hay conexiones lógicas, pero se confirma de manera individual la recepción de cada trama enviada. Si el emisor no recibe la confirmación en un intervalo, se puede enviar de nuevo. Este servicio es útil en canales no confiables, como los de los sistemas inalámbricos: 802.11 (WiFi).
- **Servicio orientado a conexión con confirmación de recepción:** El dispositivo de origen y destino establecen una conexión para transferir datos. Cada trama enviada en la conexión está enumerada, garantizando que todas sean recibidas, y que no se recibirán duplicadas.

### 3.3. Entramado

A pesar de que la capa física hace un esfuerzo por minimizar los errores, estos pueden ocurrir, y es entonces responsabilidad de la capa de enlace de datos detectarlos, y si es posible, solucionarlos.

Una forma en que logra esto es utilizando una **suma de verificación**. Si la suma de verificación hecha por la capa de enlace no coincide con la recibida por la capa física, puede desechar la trama y/o devolver un informe de error.

Otra tarea hecha por esta capa es el **entramado**: dividir los bits en tramas.

Algunas de las técnicas utilizadas para lograr esto son:

- 1) **Conteo de bytes:** Consiste en añadir un campo al encabezado que especifique el tamaño de la trama. El problema es que este campo puede ser erróneo por un error en la transmisión.

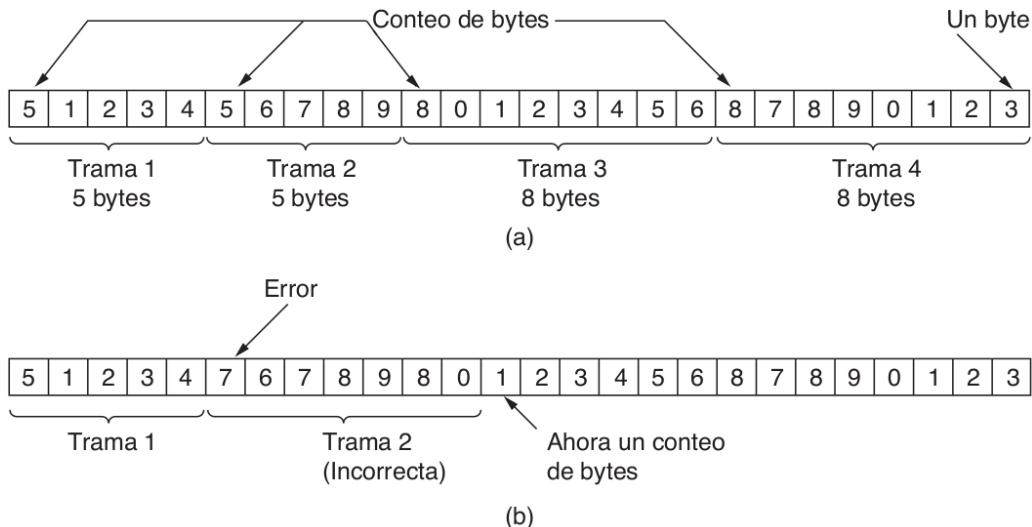


Figura 27: Un flujo de bytes. (a) Sin errores. (b) Con un error.<sup>1</sup>

- 2) **Bytes bandera con relleno de bytes:** Se coloca un **byte de bandera** en el encabezado y el payload de una trama. De esta forma, que hayan dos bytes de bandera juntos significa que terminó una trama y comenzó otra. Para evitar que sea confundido con un byte similar entre los datos, se utiliza una técnica llamada **relleno de bytes**: añadir un **byte de escape** especial (ESC) detrás de cada aparición del byte de bandera entre los datos, permitiendo diferenciar un byte de bandera de un byte entre los datos.

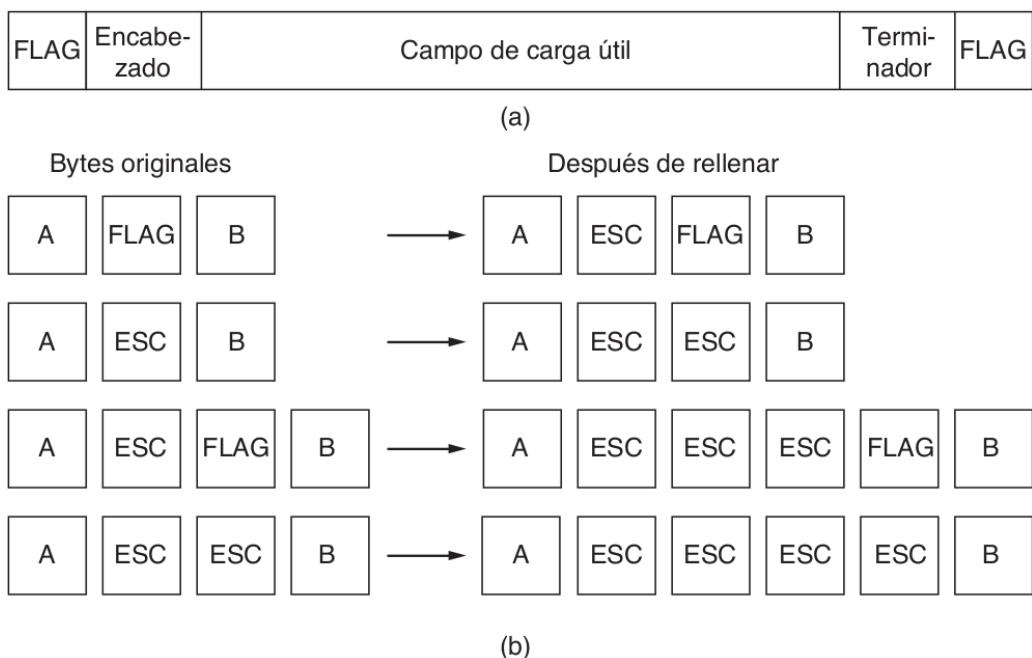
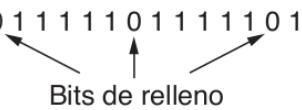


Figura 28: (a) Una trama delimitada por bytes bandera. (b) Cuatro ejemplos de secuencias de bytes antes y después del relleno de bytes.<sup>1</sup>

- 3) **Bytes bandera con relleno de bits:** Se utiliza un **byte bandera** (normalmente 0111 1110), y si la capa de enlace de datos del emisor encuentra 5 bits en 1 segmentos, coloca un 0 en el medio. Esto se llama relleno de bits, y es menos costoso que utilizar un byte entero de escape. Además, ayuda a la capa física al asegurar una densidad mínima de bits en 0. Luego, el receptor eliminará estos bits de relleno.

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0  


(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Figura 29: Relleno de bits. (a) Los datos originales. (b) Los datos, según aparecen en la línea. (c) Los datos, como se almacenan en la memoria del receptor después de quitar el relleno.<sup>1</sup>

Un efecto secundario de estos últimos dos métodos es que el tamaño de las tramas pasa a depender de la información dentro de ellas.

- 4) **Violaciones de codificación de la capa física:** Este consiste en utilizar un atajo de la capa física. Consiste en aprovechar la redundancia creada por la capa física. Esta redundancia significa que habrá ciertos patrones de bits que no ocurrirán en los datos. Por ejemplo, en el código de línea 4B/5B, se asignan 4 bits de datos a 5 bits de señal. Esto significa que no se utilizan 16 de las 32 posibles señales, y entonces, las 16 restantes pueden usarse como señales reservadas para representar el inicio y fin de las tramas, efectivamente usando “violaciones de la codificación” de la capa física.

Nota: Muchos protocolos de enlace de datos utilizan una combinación de estos métodos.

### 3.4. Control de errores

Dependiendo de si estamos en un servicio confiable sin conexión ni confirmación, o en uno confiable, el emisor querrá o no asegurarse de que los datos estén siendo recibidos antes de seguir enviando información.

Esto se logra enviando una **confirmación de recepción positiva o negativa** (donde si fuera negativa, puede reenviarla).

Por otro lado, si la trama se pierde, el emisor nunca recibirá nada. Para que no se quede esperando indefinidamente, se añade un temporizador, que, una vez expirado, determina que el envío fue fallido. En este caso, puede reenviarse la trama, pero no siempre es conveniente, ya que puede ser que el receptor reciba 2 veces la misma trama. Para solucionar esto, se les asigna un número de secuencia a las tramas, que distinguirá las retransmisiones de la transmisión original.



### 3.5. Control de flujo

La capa de enlace de datos se encarga de decidir qué hacer cuando un emisor quiere enviar mensajes más rápido de lo que el receptor puede aceptarlos. Por ejemplo, cuando un servidor muy poderoso envía grandes cantidades de información a un celular.

Hay dos técnicas de control de flujo:

- **Control de flujo basado en retroalimentación:** El receptor regresa información al emisor para autorizarle que envíe más datos o al menos indicarle su estado.
- **Control de flujo basado en la tasa:** Utiliza un mecanismo que limita la tasa de envío al emisor sin recurrir a retroalimentación del receptor.

*Acá se saltean los puntos 3.2 a 3.4.*

### 3.6. Protocolo Punto a Punto (FALTA)

## 4. Subcapa de Control de Acceso al Medio (MAC)

En la U2, vimos los enlaces de red que utilizan enlaces punto a punto. Sin embargo, sabemos que se dividen en dos tipos. En esta unidad, se cubren las redes de difusión y sus protocolos.

Una problemática de la que se encarga la **capa de acceso al medio** es la que surge cuando varios usuarios compitiendo por utilizar un mismo canal de difusión (como analogía, cuando todos quieren hablar en una llamada grupal a la vez). A estos canales se los denomina **canales multiacceso, o de acceso aleatorio**.

El protocolo que se encarga de determinar quién está en un canal multiacceso es una subcapa de la capa de enlace de datos llamada subcapa **MAC (Medium Access Control)**. Esta tiene especial importancia en las redes **LAN**, que suelen ser **inalámbricas**, y las redes inalámbricas son de difusión por naturaleza. Las **WAN** por su parte usan enlaces punto a punto, excepto por las satelitales.

La subcapa MAC es en realidad inferior a la capa de enlace de datos, pero para comprenderla mejor es conveniente saber la función de esta última.

### 4.1. El problema de la asignación del canal

El tema central de este capítulo es la forma de asignar un único canal de difusión entre usuarios competidores, donde cualquier usuario que ocupe todo el canal interferirá con los demás.

#### Asignación estática de canal

Una forma de asignar un canal entre múltiples usuarios es simplemente utilizando uno de los métodos de multiplexión de la unidad 2 (Capa Física). Por ejemplo, en FDM, el ancho de banda se divide en los N usuarios, y cada uno tiene su rango de frecuencia privado y no habrá interferencias.



Ahora bien, FDM no es buena cuando hay un número alto de usuarios o si varían continuamente, o cuando el tráfico se hace en grandes ráfagas. En estos casos, tiene un pobre rendimiento.

(Los profesores explicitaron que las fórmulas no son muy importantes, sin embargo, aclara que acá se saltean varios párrafos de explicación)

#### 4.1.1. Supuestos para la asignación dinámica de canales

- **Tráfico independiente:** Hay  $N$  estaciones, donde cada una enviará una trama, y hasta no recibir una confirmación, no enviará otra.
- **Canal único.**
- **Colisiones observables:** El único error posible es el que ocurre cuando haya colisión entre tramas, y asumimos que las estaciones podrán detectarlo, y retransmitirla si ocurre.
- **Tiempo continuo** o ranurado: Si ranuramos el tiempo, lo dividimos en intervalos discretos, donde cada uno puede tener 0 o más tramas.
- **Con o sin detección de portadora:** Si hay detección de portadora, las estaciones saben si el canal está en uso antes de intentar usarlo.

### 4.2. Protocolos de Acceso Múltiple

Vamos a ver los principales algoritmos para asignar un canal de acceso múltiple. De estos, se va a resumir de más, pues los profesores expresan que hay que: “Leer conceptualmente los puntos 4.1 a 4.3, no es necesario saber calcular las eficiencias de canal, pero si entender y poder explicar los mecanismos de control de acceso y cómo se comparan.”

#### 4.2.1. Aloha

Es el primer protocolo MAC, y tuvo dos principales versiones:

##### Aloha puro

Tiene una idea sencilla: Permitir a los usuarios transmitir datos cuando tengan algo por transmitir, considerando que las colisiones son probables y asegurándose de que el emisor sea notificado. Los sistemas de acceso múltiple con colisiones se llaman de contención.

Cuando una estación envía una trama a la computadora central, esta la reenvía a todas las estaciones. Así, una estación emisora puede escuchar a la computadora central para saber si su trama fue transmitida.

Si la trama se pierde, el emisor espera un tiempo aleatorio y la reenvía. Es importante que el tiempo sea aleatorio para que dos emisores cuyas tramas colisionaron no colisionen en cadena y en loop.

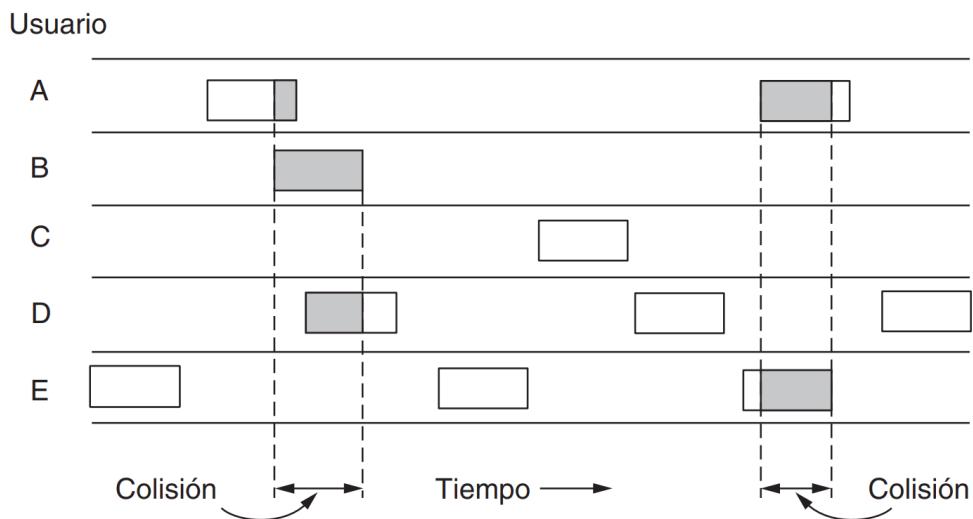


Figura 30: En ALOHA puro, las tramas se transmiten en tiempos completamente arbitrarios.<sup>1</sup>

### Aloha ranurado

Duplica la capacidad de un sistema ALOHA dividiendo el tiempo en intervalos llamados ranuras, que corresponden a una trama. Esto requiere que los usuarios estén sincronizados, lo cual se logra creando una estación especial que envíe una señal al comienzo de cada intervalo, como un reloj.

En vez de que las estaciones envíen datos cada vez que tengan algo para enviar, estas deben esperar al comienzo de una ranura. Esto reduce el tiempo vulnerable a la mitad.

#### 4.2.2. Protocolos de acceso múltiple con detección de portadora

Los protocolo ALOHA no son muy buenos aprovechando el canal. Esto es de esperar, las estaciones no tienen nada en consideración cuando envían datos, simplemente los envían en el momento en que tienen algo para enviar.

Otros protocolos más inteligentes se basan en que las estaciones escuchen a una transmisión y actúen de manera acorde. Estos se llaman **protocolos de detección de portadora**.

#### CSMA persistente y no persistente:

En el **CSMA (Acceso Múltiple con Detección de Portadora / Carrier Sense Multiple Access)** **persistente-1**, antes de enviar datos, una estación escucha el canal para saber si alguien más está transmitiendo en ese momento, esperando si ve que el canal está ocupado.

Claramente, aún existe un problema cuando 2 estaciones intentan enviar datos justo a la vez, o antes de que se notifiquen de que la otra empezó a enviarlos. Sin embargo, el principal problema de el CSMA persistente-1 surge cuando 2 estaciones que estaban esperando a que termine la transmisión de una tercera ven que termina, y envían sus datos a la vez, lo cual genera muchas colisiones.

El **CSMA no persistente** es similar: si el canal está ocupado, la estación espera. Sin embargo, la diferencia es que las estaciones no están continuamente verificando si se libera para enviar información,

lo hacen cada intervalos aleatorios. Así, se reducen las colisiones.

Finalmente, **CSMA persistente-p**, es como el persistente-1, pero cuando un canal se libera, una estación esperando comienza a enviar datos con una probabilidad  $p$ , o se posterga con una probabilidad  $1-p$ , y hará esto hasta que la trama sea transmitida, o hasta que vea que el canal está ocupado nuevamente.

### **CSMA con detección de colisiones**

El protocolo **CSMA/CD (CSMA with Collision Detection)** es la base de la LAN Ethernet clásica. Consiste en que cuando una estación nota que la trama que envía está colisionando con otra, deja de transmitirla, y no desperdicia tantos recursos, esperando un tiempo aleatorio antes de reenviarla.

Esta simplemente se da cuenta de que hubo una colisión cuando comienza a recibir datos que vienen de otra a la vez que envía los suyos. Sin embargo, la detección de colisiones es un proceso analógico, y por lo tanto, se necesita que las señales recibidas no pueden ser mucho más pequeñas que las emitidas, lo cual limita la modulación posible de las señales.

Los modelos CSMA, al igual que muchos otros de LAN, consistirán en **periodos de contención** y **transmisión**, con **periodos de inactividad** que ocurrirán cuando todas las estaciones estén en reposo (por ejemplo, por falta de trabajo).

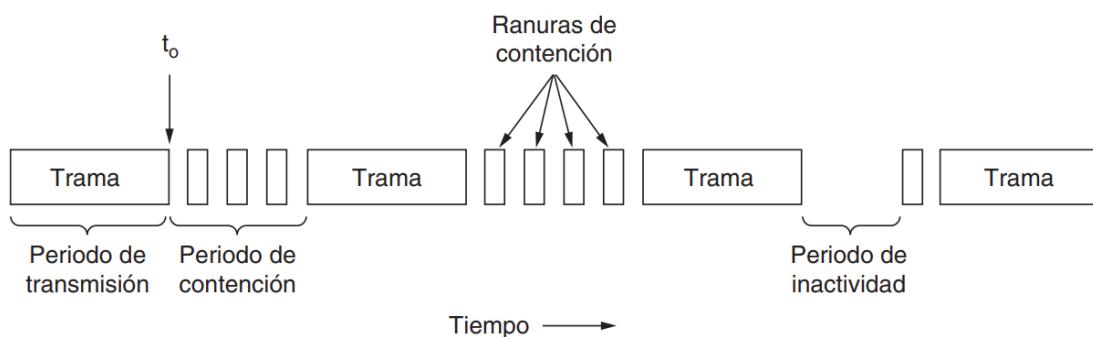


Figura 31: Estado de contención, de transmisión e inactivo, donde hubo una colisión en el momento  $t_0^1$

#### **4.2.3. Protocolos libres de colisiones**

##### **Mapa de bits**

Consiste en que, cuando se libere el canal, exista un período de contención con  $N$  ranuras, cada una previamente asignada a las estaciones. Cuando una estación desea enviar una trama, deja un bit en su ranura, y luego se la dejará enviar su trama. Si más de una ranura tiene un bit, se enviarán una por una las tramas en el orden de las ranuras. Todas las estaciones entonces saben cuántas estaciones desean enviar información, y en qué orden lo harán.

A estos protocolos se los llaman **protocolos de reservación**, porque se reserva un lugar para

enviar información en el futuro.

### Paso de token

Se define un orden de forma previa, y luego, las estaciones se van enviando un mensaje especial llamado **token** en este orden. La estación que tenga el token será la que tenga permiso para enviar una trama. Si no tiene una trama puesta en cola, simplemente pasa el token.

En este algoritmo, las tramas se enviarán junto al token. Para evitar que estas se transmitan de forma indefinida, lo que se hace es que una vez vuelvan a su estación emisora, esta última se encargue de no enviarla nuevamente.

Este algoritmo tiene un rendimiento similar al del mapa de bits.

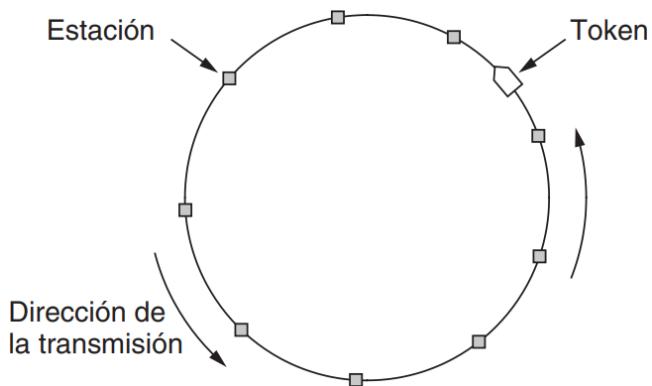


Figura 32: Protocolo *token ring*<sup>1</sup>

### Conteo descendente binario

El problema con los dos anteriores algoritmos, es que no son muy escalables en redes de miles de estaciones.

En este algoritmo, las estaciones envían sus direcciones de estación, todas del mismo tamaño, al canal compartido. Luego, cuando más de una estación envió su dirección, se hace un OR lógico para determinar quién tiene el turno.

El orden que se sigue es arbitrario, pero la forma en que se determina es por el orden de los bits: Por ejemplo, si las estaciones 0010, 0100, 1001 y 1010 están tratando de obtener el canal, en el primer tiempo de bit las estaciones transmiten 0, 0, 1 y 1, respectivamente. Luego, todas aplican un OR lógico, y las dos que enviaron un 0 ya sabrán que no es su turno, pues hay estaciones de mayor orden intentando tomar el canal. Luego, de la misma forma se determinará que la estación 1010 será la que tenga el permiso primero entre estas cuatro.

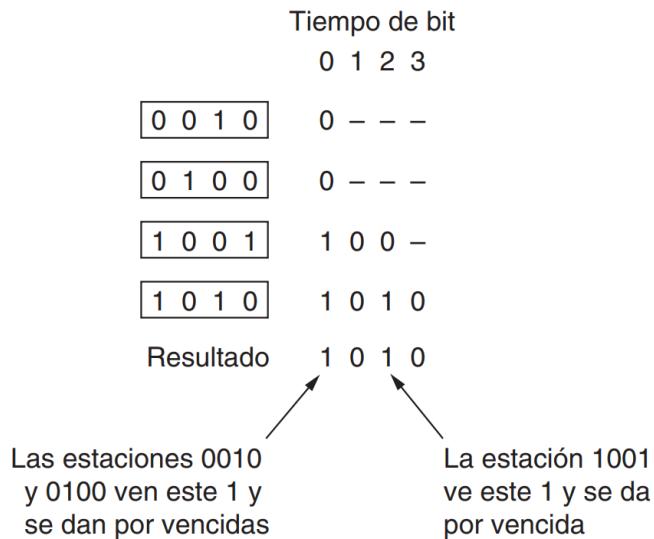


Figura 33: Protocolo de conteo descendiente binario. Un guión indica un silencio.<sup>1</sup>

Este método es **óptimo en la teoría** y casi óptimo en la práctica.

### 4.3. Ethernet

Existen dos tipos de Ethernet: Ethernet clásica, que resuelve el problema de acceso múltiple mediante el uso de las técnicas que se ven en esta unidad; y Ethernet conmutada, que usa switches para conectar a las distintas computadoras.

#### 4.3.1. Ethernet clásica

Capa física

Las primeras versiones de Ethernet utilizaban un único cable que conectaba todas las computadoras (Evolucionó de Ethernet gruesa a delgada según el grosor del cable que usaba). Tenía una velocidad de 3Mbps en sus inicios, y necesitaba de repetidores en redes grandes.

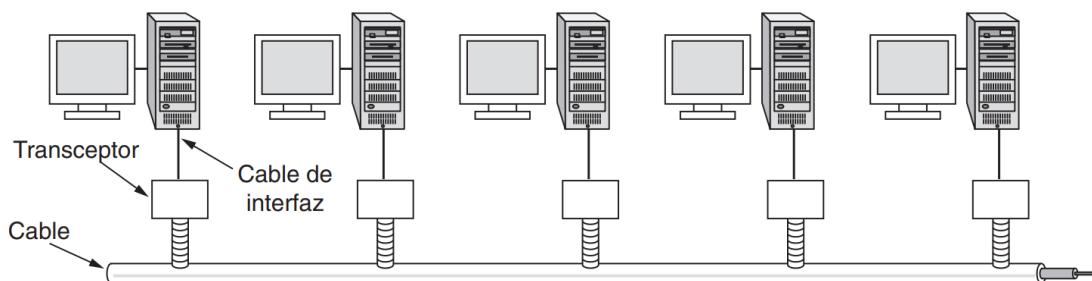


Figura 34: Arquitectura de Ethernet clásica.<sup>1</sup>

## Capa MAC

Las tramas tenían el siguiente formato:

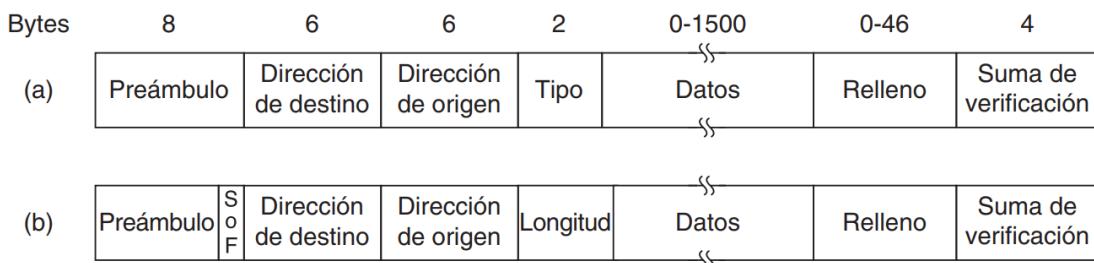


Figura 35: Formatos de trama. (a) Ethernet (DIX). (b) IEEE 802.3.<sup>1</sup>

- El **preámbulo** tenía 8 bytes, todos de la forma 10101010 (excepto el 8vo, que termina con 1). Estos indicaban el comienzo de una nueva trama y permitía sincronizar el reloj.

- Luego, están las **direcciones de destino y origen**.

Si el primer bit de una dirección de destino era un 0, se trataba de una dirección normal, y si era un 1, era una dirección de grupo. El envío a un grupo de estaciones se llama multidifusión/- multicasting, y la dirección especial con todos 1s, está reservada para difusión/broadcasting, que hace que un paquete sea aceptado por todas las estaciones en la red.

Una característica de estas direcciones es que son **globalmente únicas**, y asignadas por el IEEE.

- A continuación está el campo **Tipo** o **Longitud**, dependiendo de si la trama es Ethernet o IEEE 802.3.

### CSMA/CD con retroceso exponencial binario

La Ethernet clásica utiliza el algoritmo CSMA/CD persistente-1, con una modificación: El tiempo es ranurado y por cada colisión, esperan con cierta probabilidad exponencialmente mayores intervalos (en la i-ésima colisión, de 0 a  $2^{i-1}$  ranuras). Esto se llama retroceso exponencial binario.

En este algoritmo, si no hubieron colisiones, se asume que el paquete se envió bien. Por lo tanto, en Ethernet no se realiza una confirmación de recepción.

### Eficiencia de Ethernet

*No lo agregué, está en la sección 4.3.3 de Tanembaum*

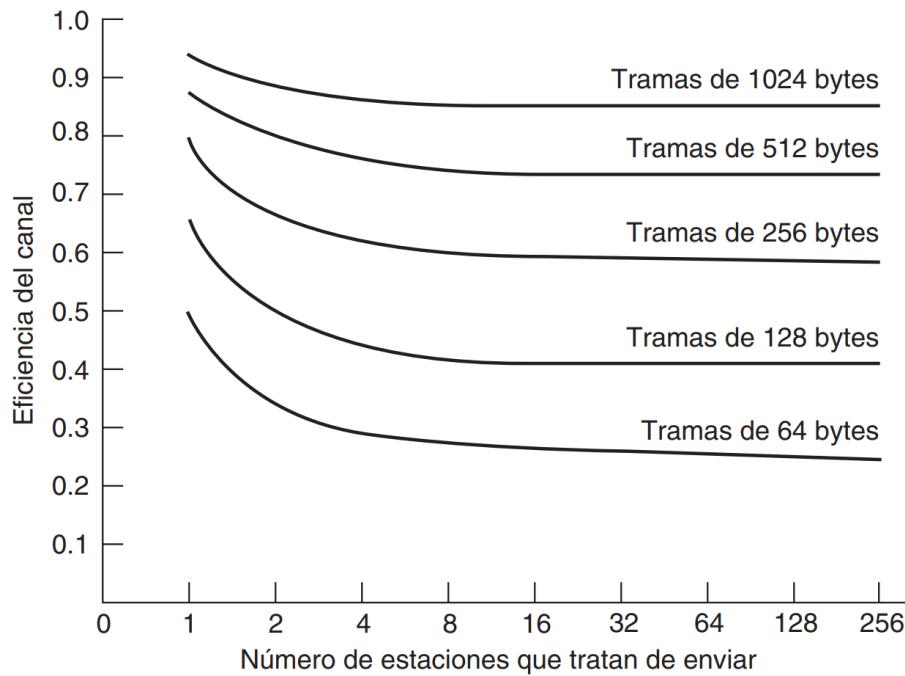


Figura 36: Eficiencia de Ethernet a 10 Mbps con tiempos de ranuras de 512 bits.<sup>1</sup>

#### 4.3.2. Ethernet comutada

Ethernet pronto dejó de usar el modelo de un único cable largo conectando a todos los usuarios. Entonces, comenzaron a utilizarse **hubs**, que básicamente conectan a todos los cables, como si los soldara. Esto hacía más fácil añadir estaciones, pero no incrementaban la capacidad, ya que son lógicamente equivalentes al cable extenso individual de la Ethernet clásica. La única forma que tenían de enviar más información eran cables con mayor capacidad.

Más tarde, comenzó a utilizarse una nueva forma de tratar con el aumento de carga: una **Ethernet comutada**. El corazón de este sistema es un **conmutador (switch)**.

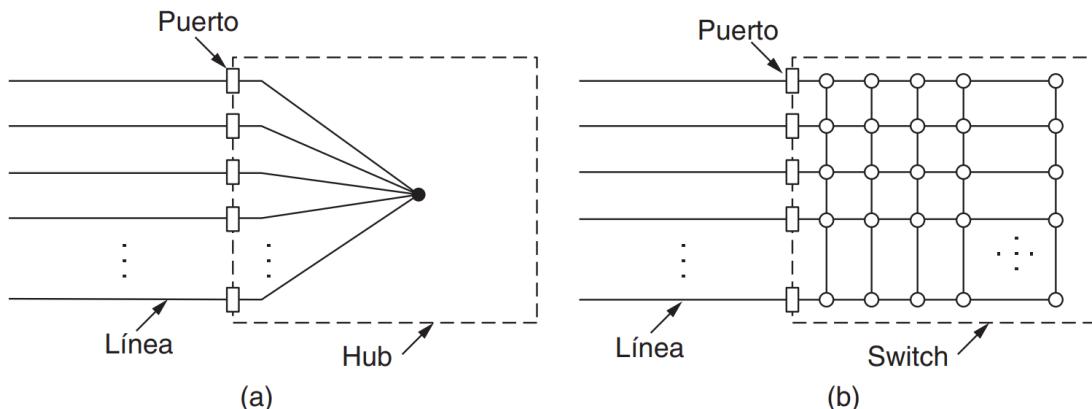


Figura 37: (a) Hub. (b) Switch.<sup>1</sup>

Los switches se ven idénticos a los hubs, y mantienen la ventaja de agregar estaciones de forma sencilla.

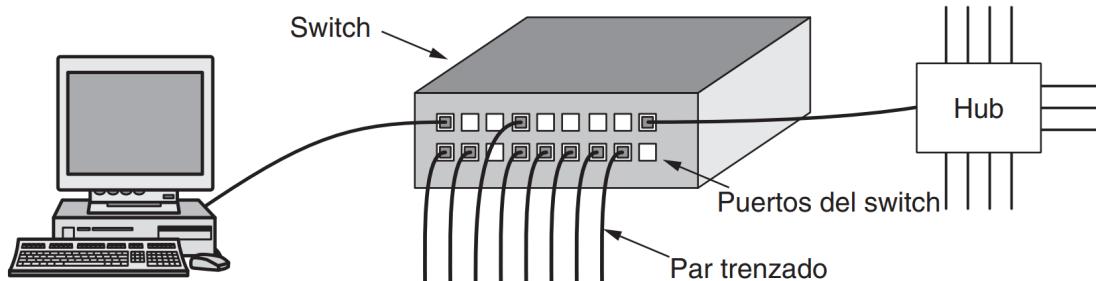


Figura 38: Switch Ethernet.<sup>1</sup>

Los switches implementan una optimización muy importante: sólo envían tramas a los puertos para los cuales están destinadas. De esta forma, no habrán colisiones incluso si dos estaciones envían una trama a la vez.

Se dice que en un hub, todas las estaciones están en el mismo **dominio de colisión**, y en un switch, *cada puerto es su propio dominio de colisión independiente*.

Los switches cuentan con un buffer interno para cuando hay dos paquetes que deben ser enviados hacia una misma dirección. Además, el hecho de que no todos los paquetes sean enviados hacia todas las estaciones implica una gran mejora en la seguridad del sistema. Eólo existirá una colisión en un cable que sea half-duplex.

*Acá en el libro se habla de Fast Ethernet (10 Mbps), Gigabit Ethernet, y 10 Gigabit Ethernet. Pero no lo cubro porque sólo habla de cómo evoluciona el Ethernet con estos. Si interesa, se puede leer desde Tanembaum 4.3.5.*

## 5. Redes Inalámbricas

*Esta sección viene de las transparencias dados por los profesores y el cuadernillo de redes inalámbricas.*

### 5.1. WLAN: WiFi

El **protocolo IEEE 802.11** o **WiFi** define las capas físicas y de enlace de datos para la implementación del modelo OSI en redes inalámbricas.

#### Arquitectura de WiFi

La arquitectura WiFi consiste en un conjunto de nodos conectados inalámbricamente con un **Access Point (AP)**, que a su vez está conectado con un router, que los conecta con el Internet. Este sistema en conjunto se conoce como **Sistema de Distribución**.

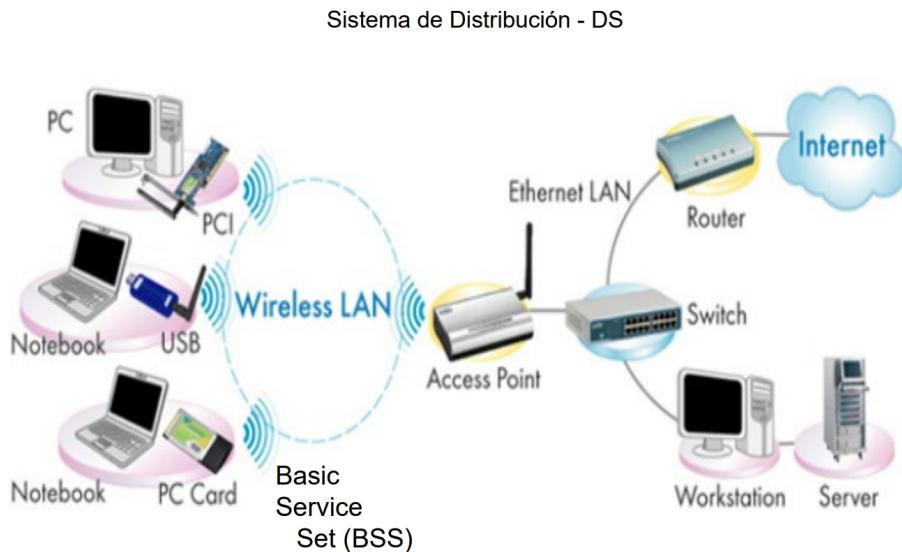


Figura 39: Arquitectura de una red WiFi.

### 5.1.1. Capa física

#### Funciones:

- Codificación/Decodificación de las señales
- Generación y remoción de la Cabecera (Sincronización)
- Trasmisión/Recepción de bits
- Especificaciones del medio de transmisión

El estándar de WiFi utiliza tres formas de aprovechar el espectro electromagnético:

- **Espectro Disperso por Salto de Frecuencia (Frequency-Hopping Spread Spectrum, FHSS)**

Consiste en transmitir una parte de la información en una determinada frecuencia durante un intervalo, y luego, cambiar la frecuencia de emisión y se sigue transmitiendo en otra frecuencia. El orden de los saltos de frecuencia es una secuencia pseudoaleatoria sólo conocida por el emisor y receptor de los mensajes.

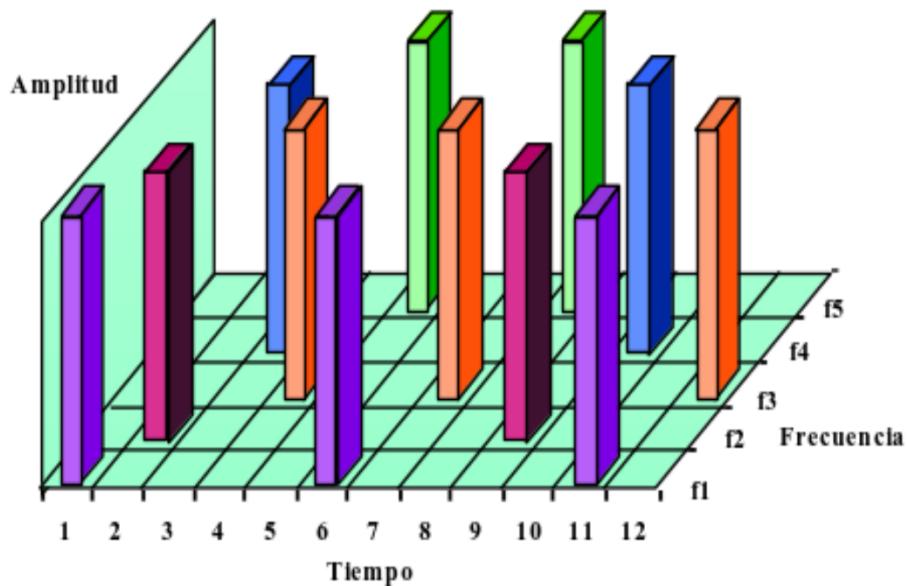


Figura 40: FHSS.

- **Espectro Disperso con Secuencia Directa (DSSS)**

Consiste en que cada bit en la señal se represente con un patrón de bits que sólo el emisor y receptor conocen, llamado secuencia de Barker, con el objetivo de añadir redundancia que haga a la señal más resistente a la interferencia.

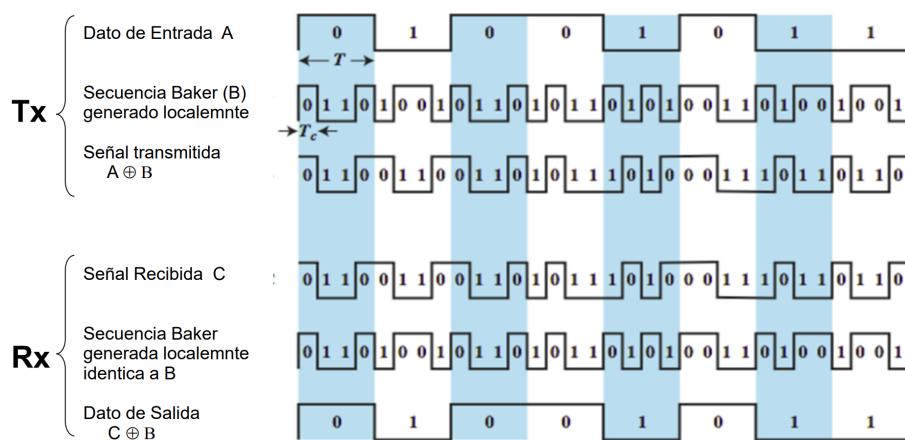


Figura 41: DSSS.

- **Múltiple Entrada / Múltiple Salida (MIMO)**

Consiste en utilizar múltiples antenas de transmisión y recepción para reducir la tasa de error por bit.

### 5.1.2. Capa de enlace

En las redes inalámbricas, a diferencia de las alámbricas, donde se asume que un paquete que se haya enviado llegó correctamente, es posible que por interferencia o ruido hayan problemas en el envío.

Por esto, en WiFi se utiliza un servicio con confirmación de recepción, que utiliza **acknowledgements (ACK)** para lograrlo.

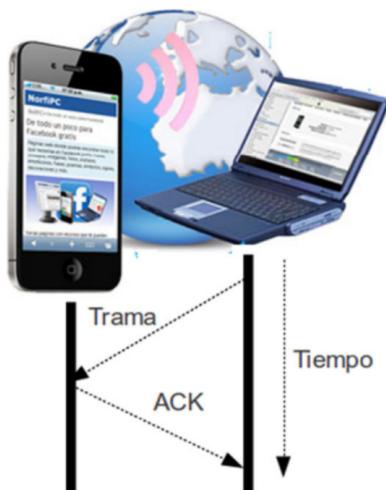


Figura 42: Confirmación de recepción.

Ahora bien, como estamos en una red de difusión, será necesario elegir un protocolo para el control del acceso al medio. Sin embargo, tenemos un problema adicional conocido como el de la **estación oculta**, donde, como se ve en la representación, puede ocurrir una colisión en las tramas enviadas por dos estaciones A y C hacia B, estando ambas a rango de B, pero sin ser capaces de detectarla al estar ambas fuera del alcance de la otra.

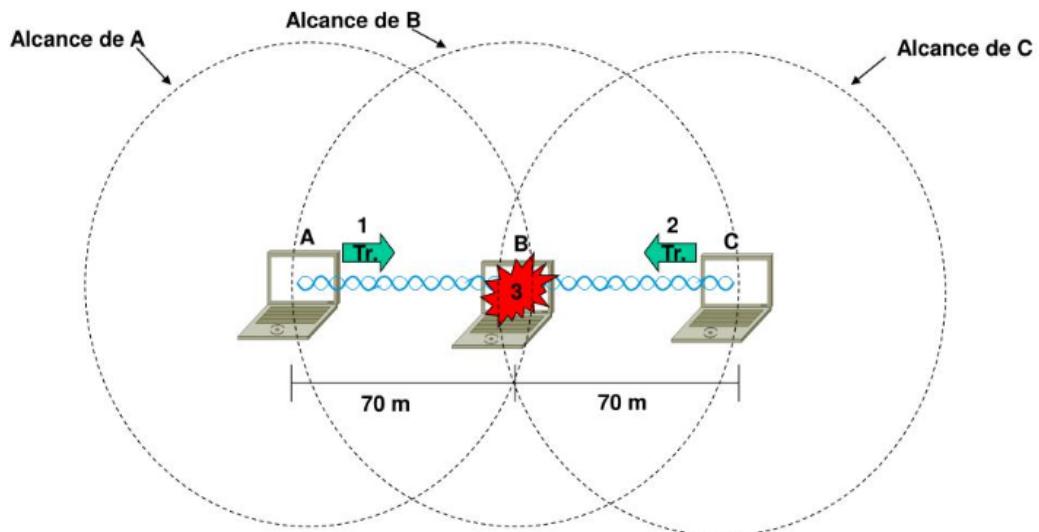


Figura 43: El problema de la estación oculta.

IEEE 802.11 implementa señales para pedir y ceder la totalidad del enlace y prevenir colisiones, utilizando señales **RTS** y **CTS** (request to send y clear to send).

La detección de portadora se utiliza para verificar si el canal se encuentra libre para el envío de datos, y son provistas por el **Vector de Asignación de Red (NAV)**. El NAV es un temporizador que indica la cantidad de tiempo por el cual el canal estará reservado. Cuando se envía un RTS, se coloca el NAV con la cantidad de tiempo por el cual se estima que se ocupará el medio, y las demás estaciones inician una cuenta regresiva hasta que el NAV llegue a 0.

Cuando el NAV termina, las estaciones esperan un tiempo aleatorio antes de intentar ocupar el canal, para evitar colisiones (similar a CSMA persistente-1).

De esta forma, la única colisión que podría ocurrir sería cuando varias estaciones envían RTS a la vez. En este caso, la estación central no enviaría el CTS, y las demás detectarían esto, y esperarán un tiempo aleatorio antes de reintentar.

En el ejemplo, la estación A enviaría una señal RTS, y comenzaría su NAV, y, suponiendo que no hubo una colisión con un RTS similar de B, recibiría un CTS de parte de B, pudiendo enviar su trama de forma segura, y B respondería con ACK cuando la haya recibido por completo.

## Tramas

**Existen 3 tipos de tramas:**

- 1) **Tramas de datos:** Las que contienen los datos.
- 2) **Tramas de control:** ACKs, CTSs, RTSs, por ejemplo. (Para ayudar a mi memoria las estudié como *de control* (de acceso al medio), al tener esta función)
- 3) **Tramas de gestión:** Son tramas usadas para la administración de la red.

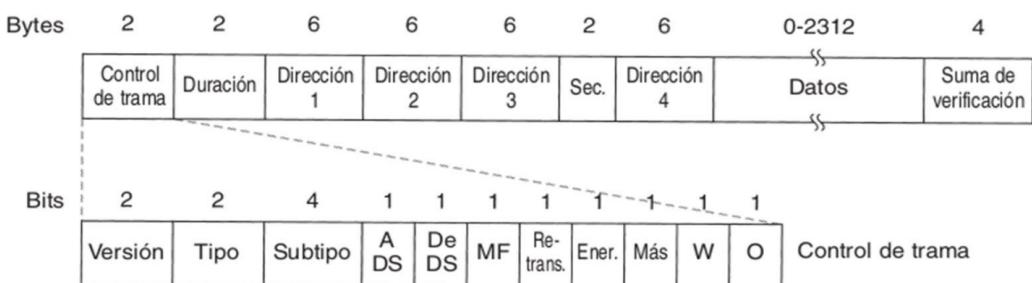


Figura 44: Trama de datos 802.11.

El campo de versión será 0, pues no se creó ninguna otra versión a día de hoy. El tipo de trama se determina por el campo **tipo**, y el **subtipo** detalla la acción realizada (ACK, CTS, RTS, asociación, autenticación, test, Beacon, etc...). Además, existe un bit **MF** (More Fragments) y **Más** (More Data), y uno que indica si un paquete está siendo retransmitido. Finalmente, el bit **O** (Order Matters) indica que el orden de los fragmentos y tramas debe mantenerse.

El campo de **duración** indica el tiempo estimado que se utilizará el NAV.

Los campos de **dirección**, en orden, son la dirección de destino, origen, la del origen, en caso de que la trama tenga destino en el sistema de distribución, y la 4ta se utiliza para cuando un AP transmite datos a otro AP.

Finalmente el campo de **Secuencia** indica el número de fragmento y de secuencia del paquete.

### Espacio intertrama

Entre las tramas se deja un espacio, que dependerá del tipo de transmisión.

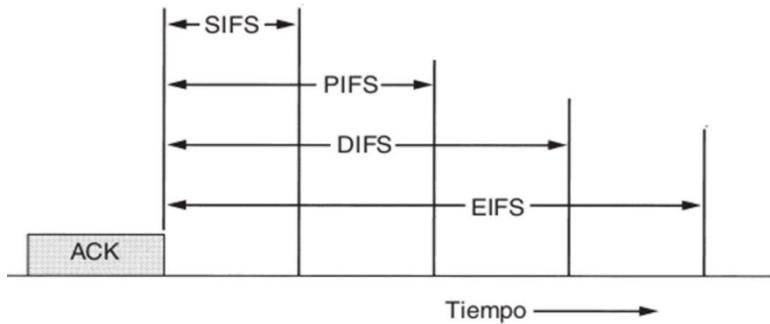


Figura 45: Espacio intertrama 802.11.

Por ejemplo, **SIFS** (Short InterFrame Space) es utilizado para transmisiones de alta prioridad, tales como tramas RTS/CTS y ACK positivas. El **PIFS** es cuando se hace una operación libre de contienda, el **DIFS** es el tiempo mínimo que se debe esperar para servicios basados en contienda. Una vez pasado el DIFS, las estaciones tienen acceso libre al medio. Y el **EIFS** es usado sólamente cuando hay algún error en la transmisión de una trama.

#### 5.1.3. Asociándose a una red WiFi

Para comunicarse en una red WiFi, los hosts deben asociarse a la red deseada. Las redes se identifican por su **SSID** (Service Set Identifier), un nombre único de una red para distinguirla de las demás.

Hay dos formas en que un host puede conocer las SSIDs: La estación base envía periódicamente tramas **beacon** con su SSID y su dirección MAC para ser descubierta. O la estación base no hace esto y el administrador es responsable de configurar el SSID. Esto no es por seguridad, ya que si captas mensajes de un host hacia la estación igual podrías ver el SSID.

#### Pasos para asociarse a una SSID:

- 1) Escanear periódicamente por SSIDs.
- 2) El host se autentifica y asocia: Para esto debe pedir autorización al AP. Muchas redes requieren una contraseña.
- 3) El host ahora puede enviar tramas a los demás hosts BSS o al router.



## 5.2. Bluetooth (FALTA)

## 5.3. LTE (FALTA)

*No incluí estos últimos dos temas. Ambos están en el cuadernillo de redes inalámbricas, y Bluetooth sí está en el resumen de docs.*

# 6. Capa de Red

## 6.1. IPv4 / Protocolo IP

Durante esta sección, hablaremos del “protocolo IP” de forma generalizada. Sin embargo, cabe destacar que la bibliografía (Libro Comer) prácticamente lo usa de forma intercambiable con IPv4. Muchas propiedades y definiciones que valen para IPv4 valdrán también para IPv6. De todas formas, en general, no quedan muchas confusiones acerca de cuándo el texto se refiere a IP y cuándo específicamente a IPv4.

El **Internet Protocol (IP)** es un protocolo de entrega de paquetes **no confiable, sin conexión**, que se denomina de **best effort**.

El protocolo IP da 3 definiciones importantes:

- La unidad básica de información se llamará **paquete o datagrama**.
- El protocolo IP provee la función de **routing**, es decir, elegir el camino que seguirá el paquete para llegar a su destino.
- Da una serie de reglas que definen un sistema de entrega de paquetes **no confiable**

### 6.1.1. Datagrama IP

Los datagramas IPv4 tienen una cabecera de entre 20 y 60 bytes:

0	4	8	16	19	24	31		
<b>VERS</b>	<b>HLEN</b>	<b>SERVICE TYPE</b>	<b>TOTAL LENGTH</b>					
			<b>IDENTIFICATION</b>	<b>FLAGS</b>	<b>FRAGMENT OFFSET</b>			
			<b>TIME TO LIVE</b>	<b>PROTOCOL</b>	<b>HEADER CHECKSUM</b>			
<b>SOURCE IP ADDRESS</b>								
<b>DESTINATION IP ADDRESS</b>								
<b>IP OPTIONS (IF ANY)</b>					<b>PADDING</b>			
<b>DATA</b>								
...								

Figura 46: Cabecera IPv4.<sup>2</sup>

Los campos del header son los siguientes:



- 1) **Vers:** 4 (IPv4).
- 2) **Header Len (HLEN):** En palabras de 32 bits (mínimo 5, máximo 15)
- 3) **TOS:** Los primeros 3 bits marcan la prioridad (0-7), luego, hay 3 bits llamados D, T y R: D pide retardos cortos, T alto data flow, R alta confiabilidad. (Los últimos 2 bits no son utilizados).
- 4) **Longitud total:** Mide la cabecera y los datos, en octetos, máximo  $2^{16} = 65535$  bytes.
- 5) **Identificación, DF (Don't Fragment), MF, Fragment Offset:** Campos de fragmentación en múltiplos de 8 bytes; x bit sin uso
- 6) **Time To Live:** Contador de saltos. Este comienza en un valor n y se descuenta 1 en cada salto. El paquete se descarta cuando el TTL es cero.
- 7) **Checksum:** De la cabecera (no incluye los datos).

### 6.1.2. Direcciones IPv4

En el protocolo IP, los hosts son identificados a través de **direcciones IP**.

En el caso de IPv4, a cada host se le asigna un int de 32 bits como dirección IP. Estas se representan con 4 enteros del 0 a 255 representando cada octeto de la dirección.

Conceptualmente, las direcciones IP son un **par (netid, hostid)**. Las direcciones IPv4 tienen diferentes **clases**, cada una de las cuales tienen una **máscara de red** diferente, que representa el tamaño del netid.

Clase	Rango	Máscara	Direcciones privadas
Clase A	1-127	255.0.0.0	10.0.0.0 – 10.255.255.255
Clase B	128-191	255.255.0.0	172.16.0.0 – 172.31.255.255
Clase C	192-223	255.255.255.0	192.168.0.0 – 192.168.255.255
Clase D	224-239		Direcciones reservadas para multicast.
Clase E	240-255		Direcciones reservadas para investigación.

Cuadro 1: Clases de direcciones IP.

Adicionalmente, existen ciertas direcciones privadas. Esto es en parte para el intento de reutilizar lo más posible las direcciones IPv4 representables, pues no existen suficientes direcciones IP para cada dispositivo conectado al internet.

También existen algunas direcciones IP reservadas para ciertos usos:

- Loopback: 127.0.0.1/8
- Broadcast a la red: netid.255(.255.255)
- Broadcast red local: 255.255.255.255/32



- Host en la red: 0.hostid/8

## Subnetting

Ahora bien, supongamos que se nos es asignado cierto rango de direcciones, supongamos, en una red de clase C, tenemos asignada para nuestra red la dirección 192.168.1.0/24. Si quisieramos tener más de una red en nuestra organización, podríamos simplemente tomar bits del hostid para extender nuestro netid:

Máscara de subred	Binario	Número de subredes	N.º de hosts por subred	Ejemplos de subredes (x=a.b.c por ejemplo, 192.168.1)
255.255.255.0	00000000	1	254	x.0
255.255.255.128	10000000	2	126	x.0, x.128
255.255.255.192	11000000	4	62	x.0, x.64, x.128, x.192
255.255.255.224	11100000	8	30	x.0, x.32, x.64, x.96, x.128, ...
255.255.255.240	11110000	16	14	x.0, x.16, x.32, x.48, x.64, ...
255.255.255.248	11111000	32	6	x.0, x.8, x.16, x.24, x.32, x.40, ...
255.255.255.252	11111100	64	2	x.0, x.4, x.8, x.12, x.16, x.20, ...
255.255.255.254	11111110	128	0	ninguna posible
255.255.255.255	11111111	256	0	ninguna posible

Figura 47: Subnetting.

### 6.1.3. Comutación/Switching (Routing)

El ruteo es el proceso elegir el camino que un paquete seguirá para llegar a su destino, y los routers son las computadoras que se encargan de esta tarea.

Un host está conectado siempre a una única red, y una red puede contener más de un router. Los routers siempre están conectados a 2 o más redes.

#### Entrega directa e indirecta

En la **entrega directa**, el paquete se entrega de un host a otro dentro de una misma red. Esto es, que ambos hosts estén en el mismo sistema de transmisión físicamente.

La **entrega indirecta** ocurre cuando el destino no está conectado directamente a la red del origen, forzando al origen a enviar el paquete a un router. En esta, el paquete será pasado de router en router hasta llegar a un router que pueda entregar el paquete de forma directa.

#### Ruteo

Los algoritmos de ruteo utilizan una **tabla de ruteo**, que, utilizando el **neitd** de un paquete, decide a través de qué puerto enviar un paquete. Como tanto los hosts como los routers realizan tareas



de ruteo, ambos tendrán sus propias tablas de ruteo, y estás serán consultadas *siempre que se tenga que hacer una tarea de ruteo*.

### Next-Hop routing

Ahora bien, si las tablas de ruteo tuvieran información de ruteo para todas las IPs serían demasiado grandes.

Usar únicamente la netid en vez de toda la IP resulta más eficiente, además de permitirnos ocultar la información de los hosts específicos.

Las tablas de ruteo contienen pares (N, R), con N la netid de destino y R la IP del router a donde se debe enviar el paquete. Al router R se lo llama **next hop**, y la idea de usar tablas de ruteo que únicamente contienen el próximo salto se llama **next-hop routing**.

### Default routes

Otra manera en que se reduce el tamaño de las tablas de ruteo y se esconde información es que, cuando un mensaje tiene un destino que no está en la tabla de ruteo, este se envía a una **default router**.

Cabe destacar que es posible insertar entradas para hosts específicos en las tablas de ruteo como casos especiales. Esto es usualmente usado para debugging, y permite mayor control al administrador de la red.

### Ruteo con direcciones IP

Cuando se rutea, no se modifica el datagrama (más que decrementar el TTL y recalcular el checksum). El paquete siempre tendrá la IP de origen y la de destino final. Por lo tanto, durante el ruteo, a menos que se pueda hacer entrega directa, el paquete se envía a otra dirección IP diferente, perteneciente a un router.

Como nota adicional, la razón por la que los softwares de ruteo utilizan direcciones IP en vez de físicas (MAC), es por claridad: las direcciones IP son más fáciles de asociar a redes específicas y por lo tanto facilitan el debugging; y para ocultar las direcciones físicas de los dispositivos involucrados, tanto por seguridad y porque la capa de Red no debería interactuar con las capas inferiores, y el punto del protocolo IP es abstraer las redes subyacentes.

#### 6.1.4. Fragmentación

Idealmente, los datagramas entran en el marco físico de las redes por las que se transmiten. Sin embargo, es posible que el tamaño de un mensaje que se desea enviar sea mayor al tamaño máximo de los paquetes que puede transmitir en una red, llamado **MTU (Maximum Transfer Unit)**.

Para lograr que un mensaje más grande que el MTU pueda ser transmitido por la red, se divide el datagrama inicial en múltiples datagramas, proceso que se llama **fragmentación**. Estos fragmentos

son luego ensamblados en el destino. Técnicamente ensamblar los paquetes en un nodo intermedio es posible, pero no suele realizarse, pues no es recomendable, por razones varias.

Sabemos además que los datagramas pueden llegar fuera de orden. Por esto es que existe el campo **offset** en la cabecera de los datagramas, este marca la posición de los datos en el mensaje (en octetos).

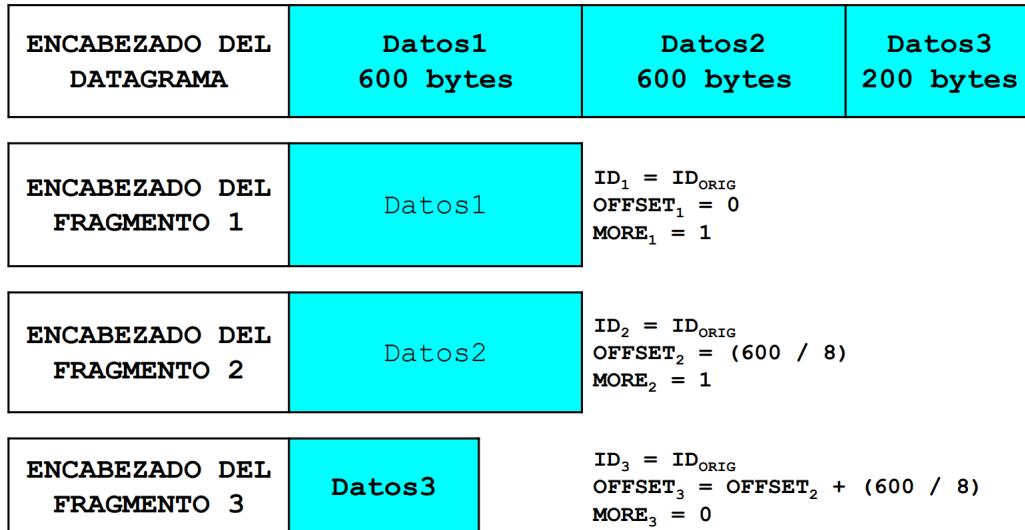


Figura 48: Funcionamiento de los campos de ID, Fragment Offset y MF de la cabecera IPv4.

En general, siempre se desea transmitir datagramas del tamaño máximo posible, por lo que se intenta que el datagrama tenga el tamaño del mínimo MTU de todas las redes por las que pasa, y no menos.

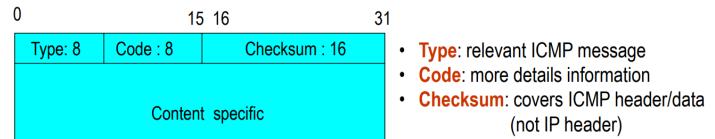
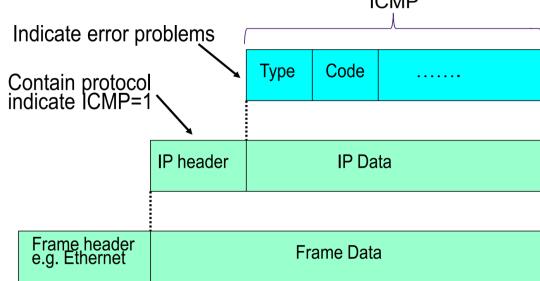
## 6.2. ICMP

El **protocolo ICMP** permite a los routers enviar mensajes de error/control hacia otros routers o hosts.

Sabemos que IP es un protocolo **no confiable**. Esto implica que pueden ocurrir errores en el envío de paquetes. ICMP permite a los routers comunicar errores de transmisión (TTL excedido, congestión, destino inalcanzable, etc...), enviando un mensaje ICMP al origen del datagrama.

Cabe destacar que ICMP se limita a reportar errores y no los soluciona.

Como todo el tráfico, *los mensajes ICMP viajan en la porción de datos de un datagrama*, y estos no tienen destino en ninguna aplicación del host de origen, sino que están dirigidos su Software IP, quien luego puede comunicarlo a la aplicación pertinente.



Campo de tipo	Tipo de mensaje ICMP
0	Respuesta de eco (Echo Reply)
3	Destino inaccesible (Destination Unreachable)
4	Disminución del tráfico desde el origen (Source Quench)
5	Redirigir (cambio de ruta) (Redirect)
8	Solicitud de eco (Echo)
11	Tiempo excedido para un datagrama (Time Exceeded)
12	Problema de Parámetros (Parameter Problem)
13	Solicitud de marca de tiempo (Timestamp)
14	Respuesta de marca de tiempo (Timestamp Reply)
17	Solicitud de máscara (Addressmask)
18	Respuesta de máscara (Addressmask Reply)

(b) Formato del segmento de datos de un mensaje ICMP .

Notas:

- Los mensajes ICMP reciben mayor prioridad por los routers.
- No se envían mensajes ICMP de mensajes ICMP.

### 6.3. ARP

El protocolo **ARP** (**A**ddress **R**esolution **P**rotocol) permite determinar la dirección MAC de un host teniendo su dirección IP, siempre que este **esté en la misma red**.

Para lograrlo, el protocolo ARP envía un **ARP Request**, que deberá claramente ser un Broadcast, a la red, y su respuesta, **ARP Reply**, contendrá la dirección MAC de la computadora cuya dirección IP coincida con la del ARP Request, y esta será Unicast.

Antes de enviar un ARP Request, el protocolo verifica si la dirección ya se encuentra en la memoria caché. Si no la encuentra, envía el ARP Request, y en caso de obtener una respuesta, la cachea.

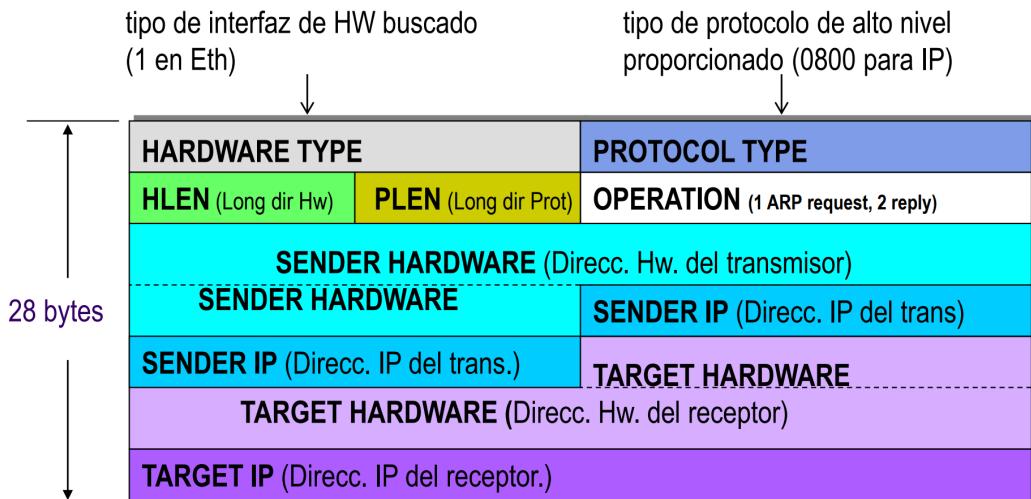


Figura 50: Los mensajes ARP se encapsulan en la trama física, y similar a ICMP, tienen su propia cabecera.

## 6.4. IPv6

IPv6 es un protocolo que se crea con el fin de solucionar muchos de los problemas que IPv4 tenía, pero en particular, porque se estaban agotando las direcciones IPv4 existentes.

### 6.4.1. Cabeceras

IPv6 tiene un formato de cabeceras diferente a IPv4.

Todo paquete IPv6 contará con una cabecera base. Esta contiene la información mínima necesitada para que el datagrama pueda ser enviado.

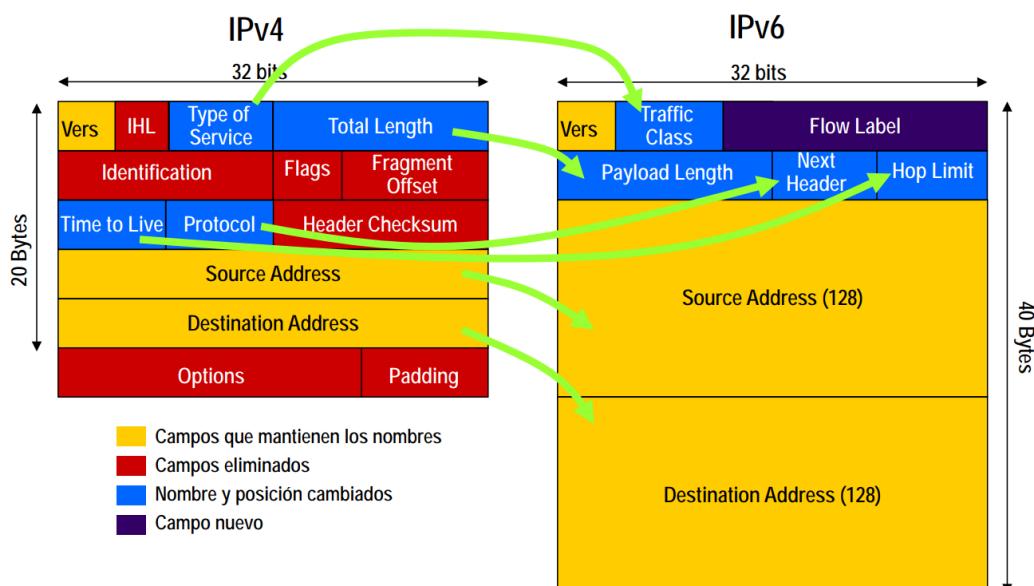


Figura 51: La cabecera IPv6 vs IPv4.

Las principales diferencias entre las 2 cabeceras son las siguientes.

IPv4	IPv6
20-60 bytes	40 bytes
Direcciones de 32 bits	Direcciones de 128 bits
Checksum, banderas, offset, y otros...	No tiene esta información.
Type of Service	Reemplazado por Traffic Class
Total Length	Payload Length
Time to Live	Hop limit
Protocol	Next Header

Ahora bien, IPv6 añade el concepto de **Cabeceras de Extensión**. Estas son cabeceras que contendrán toda la información adicional que se desee tener en el header.

Estas son identificadas por el campo **next header**, que tiene como valor el **tipo de header** que sigue. El orden es importante en estas, por razones de eficiencia: En cada nodo sólo se analizan Hop-by-Hop y Routing

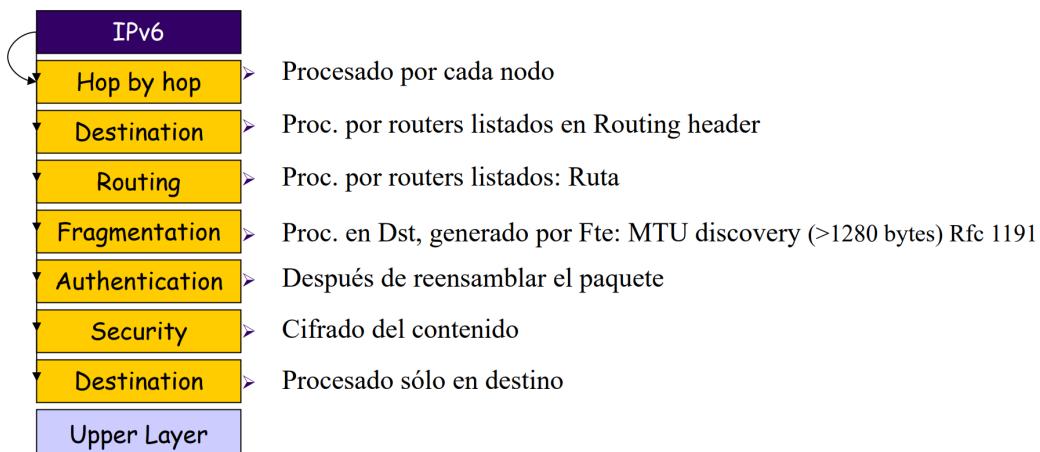


Figura 52: Cabeceras de extensión.

### Cabecera Hop-by-Hop

Esta se identifica por Next Header=0, y es procesada por todos los nodos en la ruta del paquete.

Su primer campo, **como en toda cabecera de extensión, será el campo Next Header**.

El segundo, la longitud de la cabecera Hop-by-Hop. Luego, tiene un campo de opciones con todas las opciones que deseé especificar. Los primeros 2 bits del campo de opciones especifican qué hacer si el nodo no las reconoce (ignorar y continuar procesando, descartar el paquete, descartar y enviar mensaje ICMP).

### Cabecera Routing

Esta se identifica por Next Header=43, y sirve para dirigir a un paquete a uno o más nodos intermedios antes de dirigirlo al destino.



## Cabecera Fragment

Esta se identifica por Next Header=44, se utiliza para la fragmentación en datagramas IPv6: Tiene un campo de fragment offset, de 13 bits, un campo de 1 bit M, que marca si hay más fragmentos, y finalmente un campo de fragment ID, que permite diferenciar fragmentos de diferentes mensajes.

### 6.4.2. Direcciones IPv6

Las direcciones IPv4 están compuestas por **128 bits**. (Me ahorro la explicación de cómo se representan)

Como en IPv4, las direcciones IP no se asignan a los nodos, sino que a sus interfaces.

Existen 3 tipos de direcciones IPv6 según su alcance:

- **Global:** Estas son direcciones globales en internet.
- **Unique Local:** Ruteos intranet (en sitios).
- **Link Local:** De subred, sólo válidas dentro de la red física.

Además, existen:

- **Unicast:** identifican a una interfaz de un solo nodo (de uno a otro).
- **Anycast:** identifican a un conjunto de interfaces, en general de nodos distintos (de uno a alguno).
- **Multicast:** identifican a un conjunto de interfaces (de uno a todos los del grupo).

Loopback: ::1

## Unicast

- 1) **Globales** Las direcciones Unicast Global identifican a una interfaz de un solo nodo (envíos a un destino único). Estas son globalmente ruteables, similar a las direcciones IPv4 públicas.

Su rango es de la dirección 2000 a 3fff (los 2 primeros bits en 0 y el 3ro en 1 marca Unicast Global)

Luego, la cantidad de subredes que se pueden asignar dependerá del tamaño del prefijo asignado por el RIR, llegando a 0 en el caso que el prefijo sea /64.

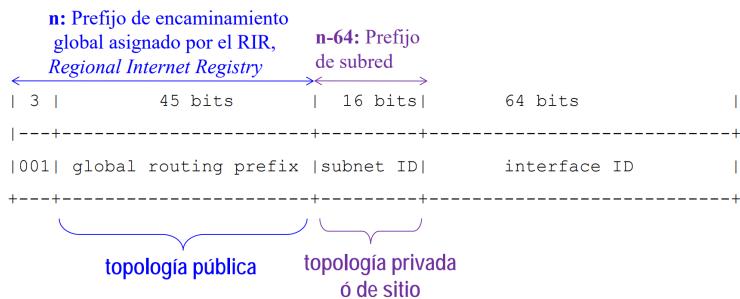


Figura 53: Formato de las direcciones Unicast Globales.

## 2) Unique Local

Se usa para ruteos internos dentro de un conjunto de enlaces. Un enlace /48 puede asignar direcciones independientemente del ISP.

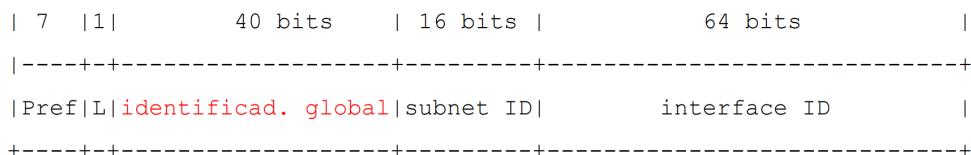


Figura 54: Formato de las direcciones Unicast Unique-Local, donde el identificador global es pseudo-aleatorio y probablemente único.

### 3) Link-Local

Sólo son válidas en el enlace local donde la interfaz está conectada. Su uso es para tareas administrativas, como el descubrimiento de vecinos, y se generan de forma automática. Se identifican por tener el siguiente formato: FE80:0:0:0/64

## Anycast y Multicast

**Anycast** identifica a un grupo de interfaces, que fueron asignadas a partir de direcciones unicast (tienen la misma sintaxis).

Sus usos incluyen el descubrimiento de routers y servicios en la red, y el balanceo de carga. Un paquete entregado a esta dirección es entregado al router más cercano al origen dentro de la red, y todos los routers deben aceptarlo.

Son direcciones de la forma **prefijo\_de\_subred+ID\_de\_interfaz=0**.

**Multicast** también identifica a un grupo de interfaces, y se representa utilizando FF en los primeros 8 bits:



Figura 55: Formato de las direcciones Multicast.

Los próximos 4 bits son banderas, y los bits de scope marcan el alcance del multicast. Los 112 bits restantes identifican el *grupo multicast*.

Valor Scope	Descripción	Dirección	Alcance	Descripción
1	Interfaz (loopback)	FF01::1 FF01::2	Interfaz Interfaz	Todas las interfaces ( <i>all-nodes</i> ) Todos los routers ( <i>all-routers</i> )
2	Enlace	FF02::1 FF02::2 FF02::5 FF02::6 FF02::9 FF02::D FF02::1:2 FF02::1:FFXX:XXXX	Enlace Enlace Enlace Enlace Enlace Enlace Enlace Enlace	Todos los nodos ( <i>all-nodes</i> ) Todos los routers ( <i>all-routers</i> ) Routers OSPF Routers OSPF designados Routers RIP Routers PIM Agentes DHCP <i>Solicited-node</i>
3	Subred			
4	Admin (configurado)			
5	Site			
8	Organización			
E	Global			
0,F	Reservados			
6,7,9,A	No distribuidos			
B,C,D				

Figura 56: *Scope* de las direcciones multicast.

#### 6.4.3. Funcionalidades

##### ICMPv6

Este tiene las mismas funcionalidades que ICMPv4, aunque no son compatibles, y se ubica luego de las cabeceras de extensión. También ayuda con funcionalidades de IPv6, como la gestión de grupos Multicast. Entonces, sus funcionalidades incluyen estas, la comunicación de errores, descubrimiento de vecinos y de MTU, etc...

##### Neighbor Discovery

###### ND-A

Similar a ARPv4, es utilizado para obtener la dirección MAC de los nodos de la red, utilizando una dirección multicast solicited-node.

###### ND-B

Utilizada para encontrar routers en la red, con una dirección Multicast all-nodes ::1, con scope link local (2): FF02::1.

###### ND-G

Usada para que un host autoconfigure su dirección IP.

**Path MTU Discovery** Se usa para descubrir el MTU de un camino, así poder utilizar fragmentos del tamaño máximo posible e incrementar la eficiencia.

Funciona enviando un paquete con un MTU inicial, si es mayor al MTU de alguna red intermedia, el router lo rechaza y responde con un mensaje ICMP indicando el MTU de la red. Así, se hace con cada red intermedia y cuando el paquete llegue al destino, se conocerá el MTU del camino con certeza.

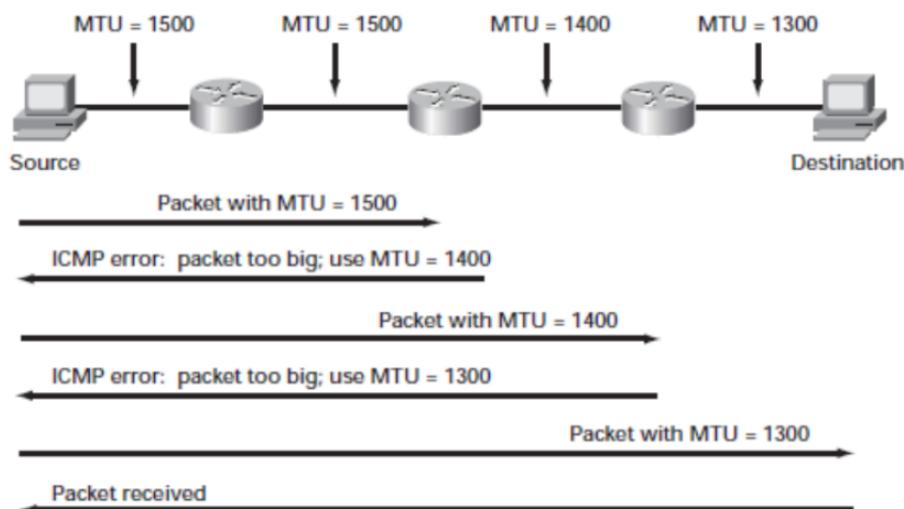


Figura 57: Path MTU Discovery.

**Calidad de servicio** La cabecera base de IPv6 tiene 2 campos para la QoS: **Traffic Class** (prioridad del paquete) y **Flow Label**. (*FALTA*)

#### 6.4.4. Transición IPv4 a IPv6

Para la transición, se utilizan varias técnicas.

- **Doble pila:** Un dispositivo soporta tanto IPv4 como IPv6.
- **Tunelización:** Encapsula paquetes IPv6 en paquetes IPv4 para que puedan pasar por redes IPv4.
- **Traducción:** Permite la comunicación entre nodos que solo soportan IPv6 y nodos que solo soportan IPv4, convirtiendo las direcciones de un protocolo al otro con ciertas reglas.

## 7. Capa de Transporte

*Esta sección corresponde a los capítulos 12 y 13 del libro Comer.*



## 7.1. Puertos

Cuando se envía un paquete, si el receptor fuera un proceso, se volvería demasiado complejo el poder identificarlo, pues puede morir, ser reemplazado, la computadora podría reiniciarse, cambiando la identificación de todos los procesos, y hasta dentro de un mismo proceso, podríamos encontrarnos que tenga múltiples hilos, y necesitaríamos identificarlo también. Por estos problemas, en vez de que el receptor sea un proceso, se utilizan un conjunto de direcciones abstractas llamadas **puertos**, que serán lo único que el emisor deberá saber. El sistema operativo por su parte le dará a los procesos una interfaz para interactuar con estos.

La data en los puertos se guarda en un **buffer**, de forma que si no hay ningún proceso intentando leerlo en un momento, no se pierda la información, y este funciona como una cola.

Los programas negocian sus puertos con el SO y se asocian a ellos. Una vez tengan asignado un puerto, pueden comenzar a utilizarlo, enviando mensajes con ese número de puerto en el campo UDP/TCP SOURCE PORT.

Cuando los protocolos reciben datos, chequean que el puerto de destino coincida con alguno de los puertos en uso. Si no lo hace, se envía un error *ICMP port unreachable*. Si coincide con alguno, encola el mensaje en el puerto.

### Asignación de puertos

Existen puertos que fueron asignados como un estándar. Existen dos formas de asignar puertos:

- **Well-known port assignments:** Que sean asignados por una autoridad central, y todo el software debe ser creado de acuerdo a estos puertos.
- **Asignación dinámica:** Los programas no saben a priori su puerto, y le piden uno al SO, quien se lo asignará dinámicamente.

El protocolo TCP/IP usa una implementación híbrida donde existen puertos pre designados para ciertas funciones, pero deja un rango de puertos disponible para cualquier otra aplicación.



Decimal	Keyword	UNIX Keyword	Description
0	-	-	Reserved
7	ECHO	echo	Echo
9	DISCARD	discard	Discard
11	USERS	systat	Active Users
13	DAYTIME	daytime	Daytime
15	-	netstat	Network status program
17	QUOTE	qtd	Quote of the Day
19	CHARGEN	chargen	Character Generator
37	TIME	time	Time
42	NAMESERVER	name	Host Name Server
43	NICNAME	whois	Who Is
53	DOMAIN	nameserver	Domain Name Server
67	BOOTPS	bootps	BOOTP or DHCP Server
68	BOOTPC	bootpc	BOOTP or DHCP Client
69	TFTP	tftp	Trivial File Transfer
88	KERBEROS	kerberos	Kerberos Security Service
111	SUNRPC	sunrpc	Sun Remote Procedure Call
123	NTP	ntp	Network Time Protocol
161	-	snmp	Simple Network Management Proto
162	-	snmp-trap	SNMP traps
512	-	biff	UNIX comsat
513	-	who	UNIX rwho daemon
514	-	syslog	System log
525	-	timed	Time daemon

Figura 58: Puertos UDP reservados en la mayoría de sistemas UNIX.<sup>2</sup>

## 7.2. User Datagram Protocol (UDP)

El protocolo TCP/IP es capaz de transmitir **datagramas**, cada uno con uno de los cuales se enrutar hacia la dirección de destino.

El **protocolo UDP** provee mecanismos para que dos programas de aplicación puedan comunicarse. Los mensajes UDP contendrán en ellos tanto la dirección del emisor y receptor, que usará aprovechando el protocolo IP (Internet Protocol), como los puertos de los programas de aplicación.

El protocolo UDP proporciona un servicio de entrega **sin conexión, poco confiable**, utilizando IP para transportar mensajes entre máquinas, y agregando la capacidad de distinguir entre múltiples destinos dentro de una computadora host determinada. (Sobresimplificando, se podría pensar en UDP como tomar IP y agregarle puertos.)

Un programa de aplicación que utiliza UDP se responsabilizará de manejar la confiabilidad, incluyendo problemas como la pérdida, retraso, desorden, o duplicación de mensajes.

### 7.2.1. Formato de los Datagramas UDP

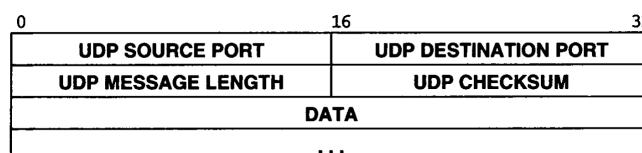


Figura 59: Formato de un datagrama UDP.<sup>2</sup>

### Pseudo-encabezado UDP

UDP añade además un **pseudo-header** que permite localizar el dispositivo de destino utilizando su dirección IP.

Para poder calcular el **checksum**, UDP puede agregar al preudoheader un byte con 0s, de forma tal de que el capo de datos tenga un múltiplo de 16 bits, y computa el checksum sobre todo el mensaje, incluyendo el header y pseudoheader. Sin embargo, estos 0s no se envían junto al mensaje, sólo se agregan para calcular el checksum.

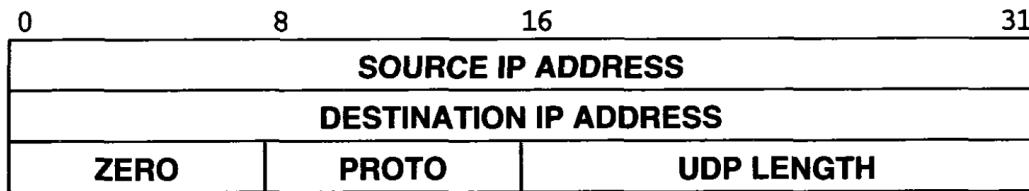


Figura 60: Pseudo-encabezado UDP usado para el cálculo del checksum.<sup>2</sup>

### Encapsulación y jerarquía de capas

Conceptualmente, las aplicaciones utilizan el protocolo UDP, que a su vez utiliza el IP para enviar y recibir datagramas.

Jerarquía conceptual: Aplicación- $\downarrow$ UDP- $\downarrow$ Internet(IP)- $\downarrow$ Interfaz de red

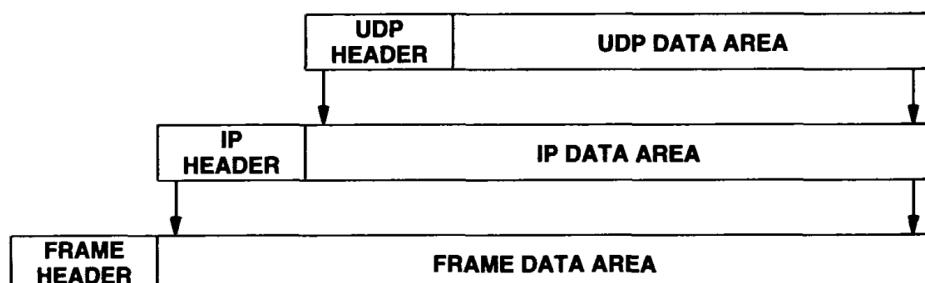


Figura 61: Encapsulación.<sup>2</sup>

Cuando se envía un paquete, cada protocolo añade su encabezado, que le permite realizar sus operaciones. En el destino, se remueven cada uno por la capa correspondiente.

De esta forma, logran crear una división de tareas clara:

- La **capa IP** es responsable *sólo* de transferir datos entre un par de hosts en una internet, y la **capa UDP** es responsable *sólo* de diferenciar entre los diferentes procesos dentro del host de destino.

Ahora bien, si el header de IP está en una capa inferior a UDP, entonces, por qué la capa UDP tiene un pseudo-header con la dirección IP? Esto se debe a que la dirección de origen de un paquete IP depende de la ruta que elija el protocolo IP para el datagrama, ya que la dirección IP de origen



identifica la interfaz de red a través de la cual se transmite el datagrama. Por lo tanto, UDP no puede conocer la dirección IP de origen a menos que interactúe con la capa IP.

Hay dos métodos para solucionar este problema:

- El primero consiste en que el software de UDP solicite a la capa IP que calcule las direcciones de origen y (posiblemente) de destino, las utilice para construir una pseudo-cabecera, calcule el checksum y luego descarte la pseudo-cabecera antes de pasar el datagrama UDP a la capa IP para su transmisión.
- Un enfoque más eficiente, consiste en que la capa UDP encapsule el datagrama UDP en un datagrama IP, obtenga la dirección de origen de la capa IP, almacene las direcciones de origen y destino en los campos apropiados de la cabecera del datagrama, calcule el checksum de UDP y luego pase el datagrama IP a la capa IP, que solo necesita completar los campos restantes de la cabecera IP.

Ahora bien, hacer interactuar a la capa UDP con la capa IP no es una violación del protocolo? Sí, pero es una violación aceptada por razones de funcionalidad.

### **7.2.2. Multiplexión UDP**

**El protocolo UDP realiza multiplexión y demultiplexión:** Acepta múltiples datagramas de diferentes programas de aplicación y los envía a la capa IP, y recibe datagramas de la capa IP y los demultiplexea, enviándolos a los programas correspondientes.

## **7.3. Transmission Control Protocol (TCP)**

El protocolo TCP es un protocolo **seguro y orientado a la conexión**. TCP es una necesidad para el envío de grandes cantidades de datos, y **asegura la recepción en orden de los paquetes**.

### **7.3.1. Propiedades de un servicio de envíos confiable**

Distinguiremos 5 propiedades:

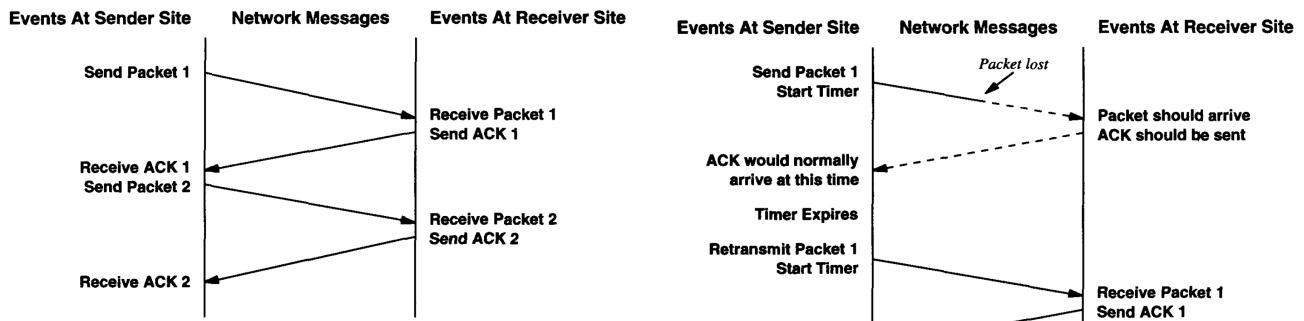
- 1) **Orientado al flujo (Stream):** Los datos se envían en forma de flujo de bytes. El receptor recibirá el mismo stream de bytes que fueron enviados por el emisor.
- 2) **Conexión de circuitos virtuales (Virtual Circuit Connection):** Antes de comenzar el envío de datos, ambos lados expresan su deseo de establecer una conexión, estableciendo los detalles y características de la comunicación. Durante la transmisión, el software de protocolo de ambas computadoras se continúa comunicando, para verificar que los datos se reciben correctamente. De estas formas, de ocurrir un error, ambas máquinas lo detectarán. El término circuito virtual se utiliza porque a esta conexión, los programas de aplicación la ven de una forma similar a un circuito dedicado de hardware.

- 3) **Buffered Transfer:** TCP envía datos en forma de stream. El software de protocolo se encarga de crear paquetes de tamaños razonables, permitiendo enviar un mensaje de gran tamaño dividido en múltiples paquetes. Pero también ocurre cuando se envían múltiples mensajes de pocos bytes cada uno: el software de red puede buffear estos mensajes en un único paquete y enviarlos. Esto incrementa considerablemente la eficiencia en la red. Cuando un programa de aplicación desea enviar datos que quizás no llenen el buffer, estos tienen la opción de forzar la transferencia, permitiendo que los datos sean transmitidos sin ningún retraso.
- 4) **Streams sin estructura:** TCP/IP no ofrece ninguna forma de distinguir dos mensajes diferentes en un stream de datos. Los programas de aplicación deberán acordar un formato de los streams antes de formar la conexión.
- 5) **Conexión Full Duplex:** Las conexiones TCP permiten transmitir datos en ambas direcciones de forma concurrente. El servicio de streams permite que una aplicación termine el flujo en una dirección mientras continúa en la otra, creando una conexión half duplex.

### 7.3.2. Proveyendo seguridad

Cómo es posible que el software de protocolo provea una conexión confiable cuando se basa en sistemas no confiables?

La técnica que utiliza TCP se llama **positive acknowledgement with retransmission**. Se basa en que el receptor confirme la recepción de paquetes enviando un mensaje de **acknowledgement (ACK)**. El emisor comenzará un timer cuando envía un paquete, de forma tal que si no recibe una confirmación de recepción y el timer expira, reenvía el mensaje.



(a) Confirmación de recepción positiva con retransmisión. En este ejemplo se muestra su versión más simple, donde se espera a recibir un ACK antes de resumir la transmisión.<sup>2</sup>

(b) El Timeout y la retransmisión que ocurre cuando un paquete se pierde.<sup>2</sup>

### 7.3.3. Ventana deslizante

Hasta ahora vimos un protocolo de acknowledgement positivo básico. Ahora bien, en la realidad, si tenemos varios paquetes, no se puede esperar a recibir una confirmación para enviar un segundo paquete, esto sería altamente ineficiente.

La técnica de **ventana deslizante** (sliding window) permite enviar varios paquetes a la vez sin esperar a recibir un ACK para cada uno.

Una manera de pensar en cómo funciona es pensando en los mensajes como una secuencia de paquetes a transmitir. El protocolo coloca una **ventana** de tamaño fijo en la secuencia y envía todos los paquetes que estén dentro de ella.

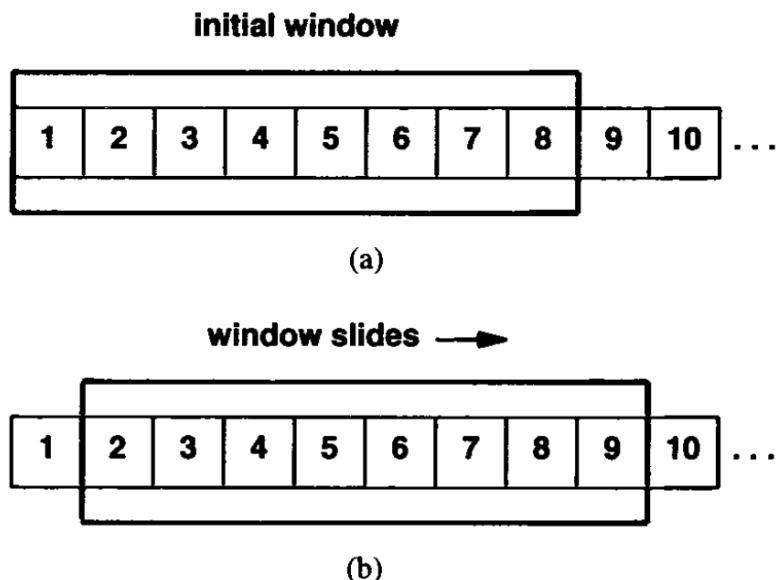


Figura 63: (a) Una ventana deslizante con 8 paquetes en ella. (b) La ventana deslizándose, permitiendo el envío del paquete 9, cuando se recibió el ACK del paquete 1.<sup>2</sup>

**Sólo los paquetes *no confirmados* (*unacknowledged*) serán retransmitidos.** Entonces, la ventana siempre tendrá tantos paquetes no confirmados en todo momento como el tamaño de la ventana.

El rendimiento de la ventana deslizante depende del tamaño de la misma. Si la ventana tuviera tamaño 1, sería similar al ejemplo dado en la sección anterior. Así, es posible eliminar el tiempo muerto en una red con una ventana de tamaño suficientemente grande, y el emisor será capaz de transmitir paquetes tan rápido como la red puede transmitirlos.

Del lado del receptor habrá una ventana análoga, que acepta y “acknowledgea” (para no decir “reconoce”) paquetes a medida que son recibidos.

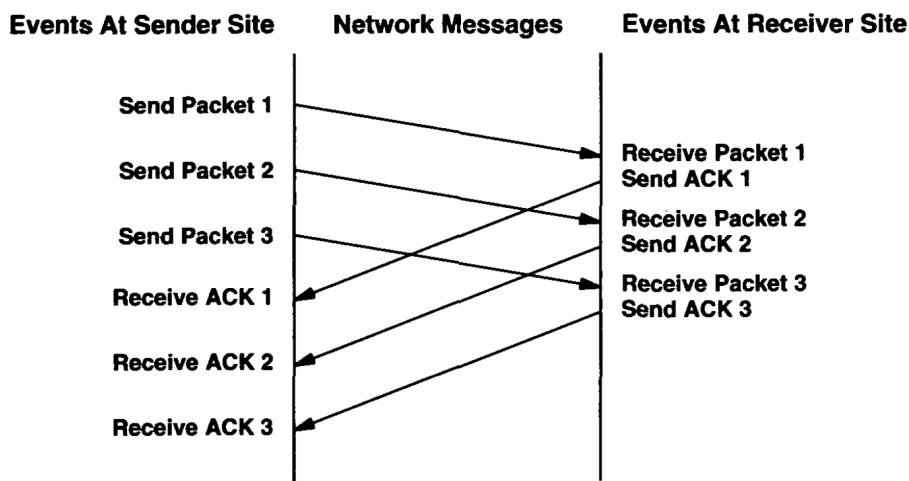


Figura 64: Tres paquetes siendo transmitidos usando una ventana deslizante.<sup>2</sup>

#### 7.3.4. El protocolo TCP

Ahora podemos comenzar a entrar en el servicio de entrega confiable de streams del protocolo TCP/IP.

#### 7.3.5. Puertos, Endpoints y Conexiones

Similar al protocolo UDP, TCP utiliza **puertos de protocolo** para distinguir el programa de aplicación al que enviar los datos dentro de una máquina de destino. Sin embargo, los puertos TCP son más complejos, ya que un número no le corresponderá a un único objeto. Esto es porque TCP se construyó sobre la idea de abstraer conexiones, donde los “objetos” a identificar son conexiones virtuales en vez de puertos individuales.

TCP utiliza **conexiones**, no puertos de protocolo, donde las conexiones están distinguidas por un par de **endpoints**. Los endpoints son un *par de enteros* (**host, port**), donde host es la dirección IP de un host y port el puerto TCP en ese host.

Luego, una conexión será un par de endpoints, de esta forma, pueden existir dos conexiones que originan en un mismo puerto, por ejemplo,

“(128.9.0.32, 1184) y (128.10.2.3,53)”,

y

“(128.2.254.139, 1184) y (128.10.2.3,53)”

Son conexiones que pueden existir simultáneamente, y, a pesar de que ambas conexiones utilizan el puerto TCP 53 y la ip 128.10.2.3, no existe ambigüedad.

Así, es posible para el protocolo TCP mantener varias conexiones simultáneas utilizando el mismo número de puerto TCP.

#### Puertos TCP Reservados

Como UDP, TCP utiliza puertos dinámicos, pero tiene ciertos puertos reservados estáticos para funcionalidades comunes (como dns, http, etc...).

### 7.3.6. Segmentos, Streams y Números de Secuencia

TCP ve los datos en octetos de bits (bytes), que divide en **segmentos** para la transmisión, donde generalmente cada segmento se translate en el internet como un único datagrama IP.

TCP utiliza un modelo de ventana deslizante que le ayuda a resolver dos problemas: **transmisión eficiente** y **control de flujo**. *La ventana deslizante de TCP trabaja con octetos, no segmentos*, y le permite enviar varios segmentos a la vez, y permite manejar el control de flujo, ya que el receptor puede restringir la transmisión hasta que tenga suficiente espacio en su buffer.

**La ventana consiste de 3 punteros:** El primero marca el comienzo de la ventana, separando los octetos acknowledged y los no acknowledged; El segundo marca el mayor octeto en la secuencia que está esperando a un acknowledgement (ya enviados y no enviados); El tercero separa los octetos que pueden ya ser enviados, de los que todavía no.

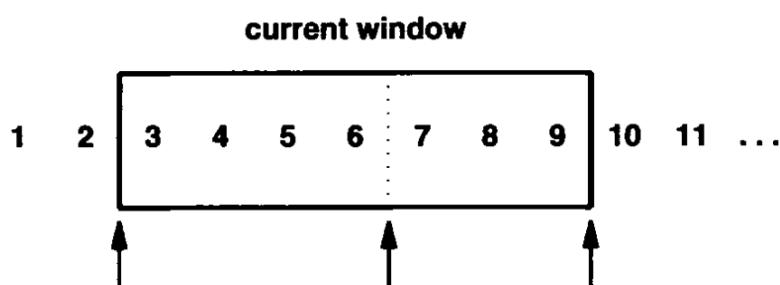


Figura 65: Ventana deslizante TCP.<sup>2</sup>

Ahora bien, como las conexiones TCP son full duplex, el software de TCP mantendrá dos ventanas en cada extremo por cada conexión (para un total de 4), donde una envía datos, y la otra los recibe.

#### Ventanas deslizantes de tamaño variable - Control de flujo

TCP permite variar el tamaño de las ventanas deslizantes. La manera en que lo hace es que cuando un programa envía un Acknowledgement, especifica cuántos octetos fueron recibidos, y además envía un **window advertisement**, donde el receptor especifica cuántos octetos más era capaz de recibir. Así el receptor puede incrementar o disminuir el tamaño de su propia ventana.

TCP sólo tendrá permitido cambiar el tamaño de su ventana como respuesta a un advertisement (exceptuando por la razón que veremos más adelante...).

Una ventaja de una ventana deslizante de tamaño variable es que **provee control de flujo**. Para evitar recibir más datos de los que puede guardar, el receptor puede progresivamente disminuir el tamaño de su ventana a medida que su buffer se llena, pudiendo en un extremo hacer un advertisement de una ventana de tamaño 0, y cuando se libere espacio en el buffer, se envía un tamaño de ventana mayor a 0.

### 7.3.7. Formato de los Segmentos TCP

Las unidades de transmisión del protocolo TCP son los **segmentos**. Se envían segmentos para establecer y cerrar conexiones, transferir datos, y enviar acknowledgements y window advertisements.

TCP permite enviar un acknowledgement dentro de un mismo segmento TCP con datos que se deseen enviar.

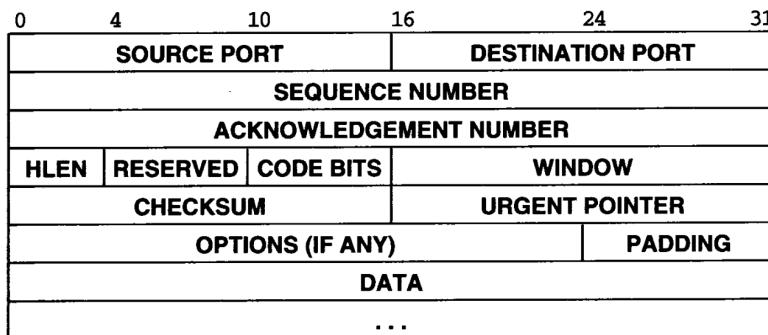


Figura 66: Formato de un mensaje TCP.<sup>2</sup>

El encabezado, conocido como encabezado TCP, contiene la información de control y la identificación esperada. Tiene los siguientes campos:

- 1) **Puerto de origen y destino.**
- 2) **Número de secuencia:** Indica la posición del segmento en el stream de datos del emisor.
- 3) **Número de Acknowledgement:** Indica el número del octeto que el emisor espera recibir luego.
- 4) **Header Length (HLEN).**
- 5) **Bits reservados para uso futuro.**
- 6) **Bits de Código/Banderas:**

<b>Bit (left to right)</b>	<b>Meaning if bit set to 1</b>
<b>URG</b>	<b>Urgent pointer field is valid</b>
<b>ACK</b>	<b>Acknowledgement field is valid</b>
<b>PSH</b>	<b>This segment requests a push</b>
<b>RST</b>	<b>Reset the connection</b>
<b>SYN</b>	<b>Synchronize sequence numbers</b>
<b>FIN</b>	<b>Sender has reached end of its byte stream</b>

Figura 67: Bits del campo CODE del encabezado TCP.<sup>2</sup>

- 7) **Ventana:** Especifica el tamaño de la ventana, es decir, el window advertisement se realiza en todos los segmentos.



- 8) **Checksum:** Se realiza de forma análoga al protocolo UDP, con el pseudo-header, con las direcciones IPs del origen y destino, y 0s para hacer de la longitud del header un múltiplo de 16 bits, etc... (Pero con el campo de UDP length cambiado por TCP length...)
- 9) **Urgent Pointer:** Si se prende la bandera de urgencia, este campo contiene el tamaño de la parte urgente del mensaje.
- 10) **Options:** Varía de tamaño dependiendo de cuántas opciones se desean enviar, y el tamaño del header entonces dependerá de este campo.

**Max segment size (MSS):** El emisor y receptor negocian el tamaño máximo de los segmentos TCP en la comunicación utilizando el campo de opciones.

#### 7.3.8. Datos fuera de banda

A pesar de que TCP está orientado al flujo, a veces puede ser importante enviar datos *fuera de banda*, sin tener que esperar a que el otro extremo consuma todos los octetos en el stream para leerlos.

Por esto, TCP permite especificar datos como **urgentes**, indicando que el receptor debe ser notificado de su arribo tan rápido como sea posible. El protocolo indica que, cuando hay datos urgentes, la aplicación asociada a la conexión debe entrar en "modo urgente".

Para indicar que hay datos urgentes en un paquete, se usa el bit **URG**, y el campo **Urgent Pointer**, que indica la posición donde terminan los datos urgentes en un segmento.

#### 7.3.9. Retransmisión

##### Acknowledgements y retransmisión

Cuando se retransmiten segmentos TCP, estos pueden ser más largos que los segmentos originales. Por lo tanto, los acknowledgements se asocian a una posición del stream en vez de un segmento, y cuando se hace un acknowledgement, se usa el número de secuencia del **último segmento contiguo recibido**.

El esquema de acknowledgements TCP se llama **acumulativo**, pues reporta cuánto del stream se ha acumulado.

Una ventaja es que elimina ambigüedades sin incrementar la complejidad, y que acknowledgements perdidos no fuerzan una retransmisión, pero una desventaja muy importante es que el transmisor no recibirá una confirmación de todas los segmentos que se hayan enviado exitosamente, porque puede perderse un segmento en el medio del stream y que hayan llegado 5 segmentos próximos a ese, y este sistema hace retransmitir todo.

##### Timeouts y retransmisión

Como todo protocolo seguro, TCP comienza un timer en el momento en que envía segmentos, y espera a una confirmación de recepción. Si el timer se acaba antes de recibirla, se asume que los datos



fueron perdidos y se retransmiten.

TCP debe tener un algoritmo de retransmisión que se adapte tanto a conexiones con un delay bajo como uno alto, pues el destino podría estar en la misma red LAN, o con decenas de nodos intermedios. Para esto, TCP utiliza un **algoritmo de retransmisión adaptativo**, donde monitorea el rendimiento de la conexión y deduce un tiempo razonable para los timeouts.

Este monitoreo consiste en registrar el tiempo en que se envía un paquete y el tiempo en que se recibe el acknowledgement que le corresponde, y con una fórmula matemática, obtiene el **RTT (Round Trip Time)**, tomando un promedio ponderado del tiempo que toma un acknowledgement en llegar desde el momento en que se envía un paquete. Es ponderado para darle un mayor peso al tiempo que tomó el último.

Finalmente, el **Timeout** será el RTT multiplicado por una constante  $\beta > 1$ .

### Medición del RTT - Algoritmo de Karn

Normalmente, medir el RTT sería trivial, simplemente se inicia el timer cuando se envía el paquete, y se termina cuando se recibe la confirmación de recepción. Sin embargo, TCP tiene un problema: si se termina el tiempo, se retransmite el paquete con exactamente la misma información, y cuando se reciba un acknowledgement, no se sabrá si se refiere al primero o segundo paquete enviado.

Este fenómeno se llama **ambigüedad de acknowledgements**, y se dice que TCP tiene **acknowledgements ambiguos**.

La manera en que se soluciona esto es utilizando el llamado **algoritmo de Karn**. Este consiste en simplemente ignorar el RTT para los segmentos retransmitidos, y no utilizarlo para calcular el Timeout.

Ahora bien, sigue habiendo el problema de que, si en un momento el delay de la transmisión incrementara significativamente, todos los paquetes se retransmitirían y nunca se actualizaría el Timeout. Para solucionarlo, se utiliza una estrategia que consiste en que cada vez que se retransmite un paquete, se incrementa el Timeout.

Entonces, el **algoritmo de Karn** combina estas dos ideas para calcular el timeout.

### Congestión y retransmisión

En TCP, si al terminar el Timeout no se recibió un acknowledgement, se reenvía el paquete. Sin embargo, a veces el paquete simplemente tenía cierto retraso. TCP no considera cuál es la razón del retraso y simplemente reenvía el paquete.

El problema radica en que, en muchos casos, los paquetes arriban tarde porque la red está **congestionada**. Esto es un problema, pues hacer retransmisiones congestionaría más a la red que ya está congestionada. De esta forma, se podría generar un círculo vicioso donde mayor congestión lleva a más retransmisiones, y estas a más congestión, hasta que la red se vuelva inutilizable, fenómeno llamado **colapso de congestión**.



### ***Slow Start y Multiplicative Decrease***

Para evitar este problema, una forma es que los routers envíen un mensaje ICMP a las fuentes de los mensajes para comunicarles el problema. Sin embargo, TCP puede reducir la congestión automáticamente reduciendo la velocidad de transmisión cuando ocurran delays.

Las técnicas que utiliza son **slow-start** y **multiplicative decrease**. Por otro lado, sabemos que en una conexión TCP, el emisor deberá conocer el tamaño de la ventana deslizante del receptor. Para controlar la congestión, TCP añade otro límite: **congestion window limit**, que limita el flujo de datos cuando ocurre congestión. La ventana TCP sigue la siguiente fórmula:

$$\text{Allowed\_window} = \min\{\text{receiver\_advertisement}, \text{congestion\_window}\}$$

De esta forma, si la congestión es baja, la ventana será la misma que la del receptor, y si es alta, se reducirá para acomodarse a las condiciones de la red.

### ***Tail-Drop Policy***

Los routers tienen una cola en su memoria con todos los paquetes que recibieron y tienen por dirigir, y utilizan una **Tail-Drop Policy**. Esto significa que si su memoria está llena por congestión, todo paquete recibido será descartado.

Al asumir esto, TCP reduce la ventana de congestión a la mitad cada vez que se pierde un paquete (multiplicative decrease). Cuando se libera la red y empiecen a recibirse los acknowledgements, la ventana de congestión se incrementará en un único segmento por cada acknowledgement recibido (slow start).

### ***Random Early Discard (RED)***

Ahora bien, Tail-Drop también genera ciertos problemas. Pues sabemos que los datagramas son multiplexados (por ejemplo, por puerto), y por lo tanto, se dropea un segmento con datagramas de N orígenes diferentes (en vez de dropear N datagramas de un mismo origen). Esto genera que todos los orígenes comiencen a hacer un slow start de forma sincronizada.

Para evitarlo, el router utiliza una técnica llamada **Random Early Discard**, que descarta datagramas de forma aleatoria cuando la cola está cerca de llenarse, evitando la sincronización global.

Este comienza a descartar paquetes con una probabilidad  $p$  que crece a medida que se llena la cola del router, llegando a  $p = 1$  cuando está completamente llena, momento en el cual funciona como Tail-Drop Policy.

#### **7.3.10. Estableciendo conexiones TCP**

Para establecer conexiones, TCP utiliza una **three-way handshake**.

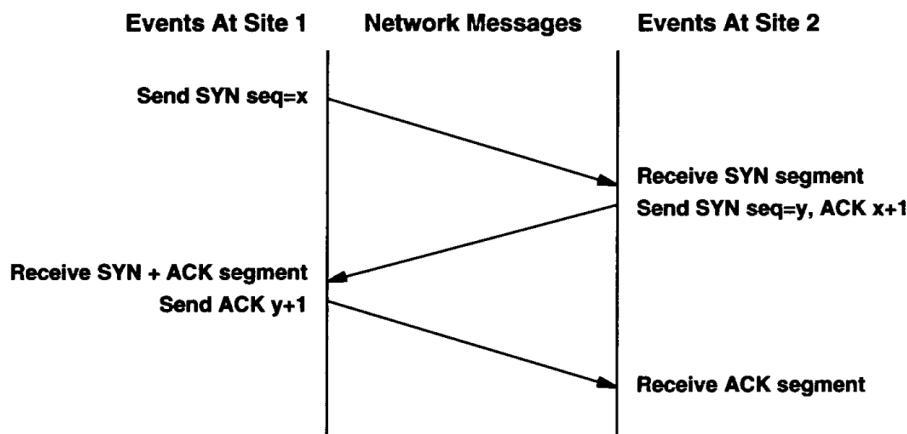


Figura 68: Three-way handshake.<sup>2</sup>

Donde el primer mensaje del handshake se puede identificar por tener la bandera SYN encendida, y su respuesta por tener tanto SYN como ACK encendidos. Finalmente, el tercer mensaje es simplemente para notificar que se ha establecido la conexión.

El software TCP suele esperar pasivamente por handshakes. Sin embargo, si dos máquinas intentan establecer una conexión simultáneamente, la conexión se realiza de la misma forma.

**El three-way handshake es necesario y suficiente para establecer una conexión.** Esto es porque con este formato de mensajes, ambos extremos de la conexión pueden calcular sus tiempos de timeout iniciales.

#### Initial sequence number

El three-way handshake garantiza que ambos lados están listos para transferir datos, pero además, les permite a ambos extremos establecer un **número de secuencia inicial**.

Cada máquina debe elegir un número de secuencia inicial de forma aleatoria.

La manera en que se establecen en una conexión en 3 mensajes es la siguiente:

- 1) A envía su número de secuencia inicial, x, en el segmento del primer SYN.
- 2) B envía su número de secuencia inicial, y, y en su acknowledgement especifica que espera el octeto x+1.
- 3) A responde que ha recibido todos los octetos hasta y, y espera al octeto y+1.

En todos los casos, los acknowledgements siguen la convención de usar el número del próximo octeto esperado.

#### 7.3.11. Cerrando conexiones TCP

Para cerrar conexiones, TCP utiliza un three-way handshake modificado.

El extremo que desea cerrar la conexión primero terminará de enviar todos los datos que debía, y luego enviará un segmento con la bandera FIN, indicando que se cerró la comunicación desde este lado. El otro extremo enviará un segmento acknowledgeando a este, y después de terminar de transmitir sus datos, enviará otro con la bandera FIN para también indicar que no hay más datos que enviar y la comunicación fue cerrada.

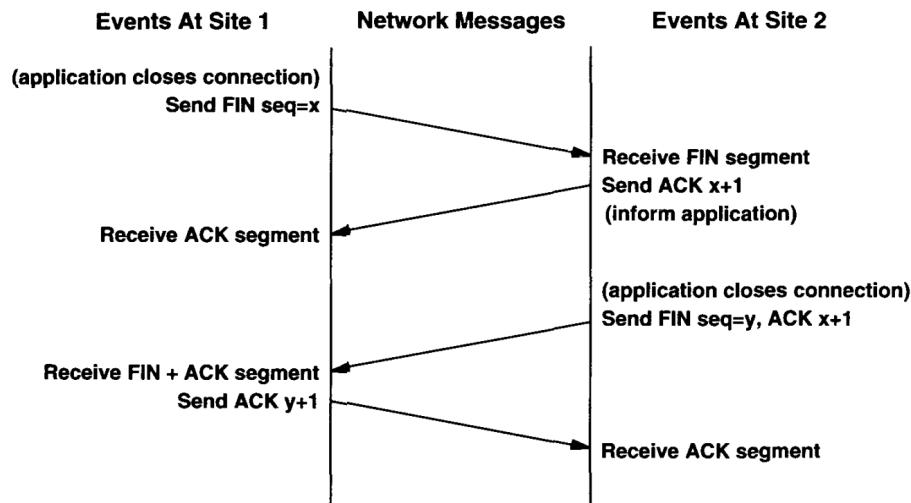


Figura 69: Three-way handshake modificado para el cierre de conexiones.<sup>2</sup>

La razón por la que hay un retraso entre el recibir el segmento de FIN y enviar el propio es que, cuando se recibe el segmento FIN, se informa a la aplicación, pero puede ocurrir que tome mucho tiempo entre que se informa a la aplicación del pedido y se cierre la conexión (por ejemplo, si involucra interacción humana). Por lo tanto, se envía el segmento ACK, para que el sitio 1 no comience a reenviar el mensaje.

### Reiniciando una Conexión TCP

Si ocurre algún problema y en algún momento un programa de aplicación necesita romper una conexión, este puede enviar un mensaje con la bandera RST. El otro extremo reacciona a este mensaje abortando inmediatamente la conexión, e informando a la aplicación que ocurrió un reinicio.

#### 7.3.12. Forzando el envío de datos

Vimos que TCP tiene libertad de dividir un flujo de datos en segmentos, sin importarle el tamaño de los mensajes que el programa de aplicación quiera enviar. TCP puede retrasar el envío de un datos mientras acumula octetos en su buffer por razones de eficiencia.

Sin embargo, en ciertas ocasiones, queremos que el envío se haga de forma inmediata y por lo tanto, TCP tiene una operación **push**, que le permite al programa de aplicación forzar el envío de todos los octetos que estén actualmente en el buffer, sin esperar a que este se llene. Esta función además prende la bandera PSH en el campo CODE del segmento enviado.



### 7.3.13. El síndrome de la ventana tonta

El **síndrome de la ventana tonta** (silly window syndrome) es un problema que ocurría en versiones antiguas de TCP que ocasionaba que los receptores anuncien poco espacio en la ventana, y los emisores envíen consecuentemente segmentos con poca cantidad de datos.

La manera en que ocurría era la siguiente:

- 1) Cuando se crea una conexión, el extremo receptor crea un buffer de K bytes y anuncia al otro extremo que su ventana tiene este tamaño.
- 2) El emisor envía rápidamente tanta información como cabía en el otro extremo.
- 3) Eventualmente, el receptor comunicará que ya no queda espacio en su ventana.

El problema comienza aquí:

- 4) Apenas quite un octeto de su buffer, este extremo comunicará que tiene espacio para recibir datos, y anunciará una ventana muy pequeña.
- 5) El emisor enviará pocos datos para llenar la pequeña ventana que se le anunció.
- 6) Se repiten los pasos 4 y 5, enviando múltiples segmentos pequeños, consumiendo banda ancha e introduciendo mayor computación de forma innecesaria en la red.

El síndrome de la ventana tonta se prevé añadiendo restricciones en el emisor y receptor, que no les permite enviar segmentos con pocos datos, o anunciar ventanas demasiado pequeñas, respectivamente.

Existen heurísticas donde sólo uno de los extremos poseen prevención de síndrome de la ventana tonta, pero suele ser conveniente implementarlo en ambos lados para no depender de la implementación del otro.

- **Del lado del emisor**

Se utiliza el **algoritmo de Nagle**. Lo que se busca es evitar enviar paquetes pequeños. Sin embargo, hay aplicaciones que pueden enviar bloques de datos arbitrariamente pequeños. Por esto, TCP recolecta los datos de varias llamadas a write de la aplicación y los envía a todos en un único segmento de tamaño razonable, una técnica llamada **clumping** (lo mismo que store buffering?).

Sin embargo, ahora surge el problema de que si se espera demasiado tiempo antes de enviar los datos, habría una cantidad importante de delay en la transmisión. TCP no sabe si debe esperar porque no sabe si llegarán más datos en el futuro. El algoritmo inteligentemente basa el delay en la performance de la red: Si el buffer no llena un segmento de tamaño razonable, esperar **hasta que llegue un acknowledgement**.

Esto ocurre incluso si se hace un *push*(!).



- **Del lado del receptor**

Es más sencillo: Luego de hacer un window advertisement nulo, esperar a que haya al menos  $\min\{MSS, Tamano\_buffer\_reception/2\}$  espacio disponible antes de actualizar el window advertisement. Además, se pueden retrasar los ACKs por hasta 500ms (y no más, para no complicar el cálculo del RTT).

## 8. Capa de Aplicación

### 8.1. DNS

Un **Servidor DNS** realiza dos tareas principales:

- **Forward mapping:** Encontrar la dirección IP física de un recurso web, conociendo su nombre.
- **Reverse mapping:** Encontrar el nombre de un recurso web, conociendo su dirección IP física.

Sin estos, sería imposible que cada host conociera la dirección IP física (que puede ser unique global o unique local) de todos los recursos web en Internet.

Con un servidor DNS, basta con que el host conozca el nombre del recurso web **nombrado** que desea acceder y la IP del servidor DNS. El host le preguntará (**querying**) al servidor cuál es la dirección del recurso.

Ahora bien, si fallara un servidor, el host perdería acceso total al Internet, por lo que existen servidores secundarios, terciarios, etc...

#### **Jerarquía en servidores DNS**

Existen muchas direcciones IP, por lo tanto, sería ineficiente que un sólo servidor DNS tuviera las IPs asociadas a todos los Nombres de Dominio.

Por lo tanto, existe una jerarquía de servidores que permite a cada servidor delegar una parte de la búsqueda a otros servidores.

#### **8.1.1. Implementación**

A la hora de implementar servidores de nombres, históricamente emergieron 3 necesidades:

- Necesidad de jerarquía de nombres.
- Necesidad de dividir el trabajo entre varios servidores.
- Necesidad de delegar la administración de los servidores.

#### **Dominios y delegación**

DNS usa una estructura de nombres en forma de árbol (jerárquica). En la cima se encuentra la raíz, seguida por los Top Level Domains (TLD), seguidos por tantos niveles inferiores como hayan

puntos en el nombre. Nota: La cima se representa con un punto ('.') silencioso. Los TLD se dividen en dos tipos:

- **gTLDs (genéricos):** .com, .net, .org, etc... Estos tienen una nueva subclase llamada **sTLDs (sponsored)**, donde una organización puede registrar cualquier TLD dado suficiente pago. Además, existen gTLDs reservados, como .gov o .edu.
- **ccTLDs (country-code TLDs):** .us, .ar, .uk, .tv, etc...

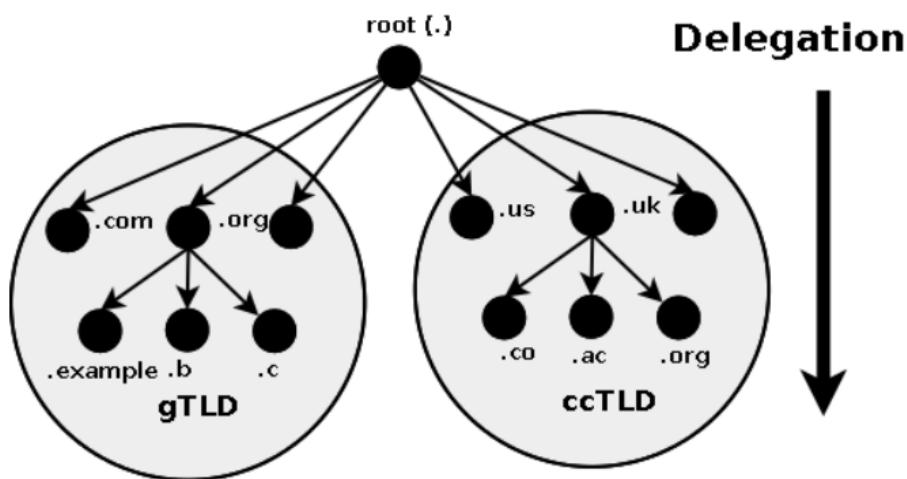


Figura 70: Estructura y delegación de dominios.<sup>3</sup>

Lo que llamamos un Domain Name es en realidad la combinación entre el nombre de dominio y un TLD. En los Domain Names, la jerarquía se lee de derecha a izquierda.

### Autoridad y delegación

La autoridad que se encarga de designar los gTLDs es la ICANN (Internet Corporation for Assigned Numbers and Names), y los de los ccTLDs se encargan sus respectivos países.

En la Figura 1-1 se ve como cada una autoridad puede delegar a niveles más bajos sobre los que tenga autoridad, y pueden haber delegaciones ilimitadas.

En www.example.com, example.com fue delegada de un gTLD por ICANN, y la parte de www fue escogida por el dueño de Domain Name example, ya que todo a la izquierda de example le pertenece a este, y puede delegar lo que quieran a la izquierda del Domain Name que se les asignó (si quiere poner pepe.example.com, nada se lo impide).

### Estructura y Organización en DNS

En DNS, para cada nivel en la jerarquía delegada existe un servidor DNS, y la responsabilidad de correr el Domain Name System le corresponde al control autoritativo del nivel.

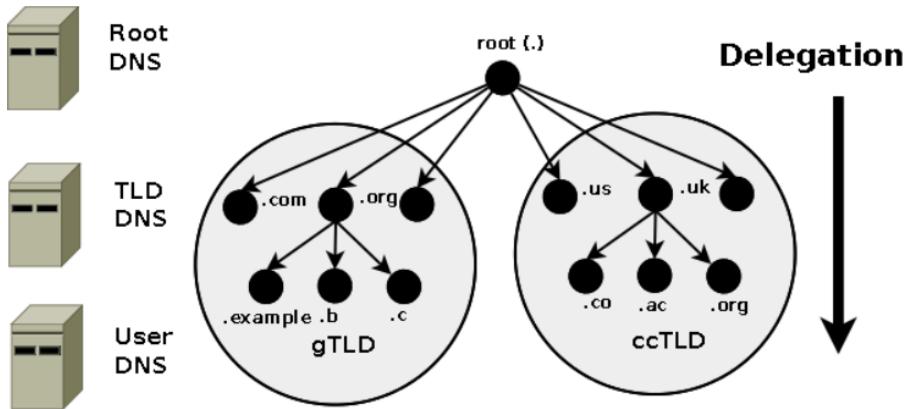


Figura 71: DNS mapeado a la delegación de dominios.<sup>3</sup>

Existen limitados servidores DNS raíces en el mundo, que son responsabilidad de ICANN. Sus direcciones IP son conocidas por todos los servidores DNS del mundo.

Si un servidor DNS no puede responder a un pedido (query) de un cliente, el query es pasado a un root-server, quien dirigirá el query al servidor DNS TLD, quien luego dirigirá el query al servidor Domain (User) DNS apropiado.

### Componentes DNS

Un **Sistema de Nombres de Dominios (DNS)** definido por RFC 1034 incluye 3 partes:

- Datos que describen al dominio.
- Uno o más programas de Servidores de Nombres.
- Un programa o librería resolver.

Un **servidor DNS** puede soportar varios dominios, y suelen hacer 3 tareas:

- Lee su archivo de configuración para conocer las zonas sobre las que tiene responsabilidad.
- Algunos servidores DNS permiten funcionalidades especiales como especificar si cachear o no.
- Responder preguntas (**queries**) de hosts locales y remotos.

#### 8.1.2. Zone files

Los **archivos de zona** contienen **Resource Records**, que describen un dominio o subdominio. Están compuestos por la siguiente información:

- 1) Información sobre cuál es la parte superior de la zona y sus propiedades (registro SOA)
- 2) Datos autoritativos para todos los nodos en la zona (Registros A (IPv4) o AAAA (IPv6)).



- 3) Información global de la zona (Registros MX para correo, o NS para servidores de nombres).
- 4) En el caso de la delegación de subdominios:
  - a) Los servidores de nombres responsables de los subdominios (uno o más Registros NS).
  - b) Uno o más **"glue records"** que permiten a un servidor de nombres llegar al subdominio (Registros A o AAAA) para los servidores de nombres del subdominio.

En los archivos de zona existen 3 tipos de entradas:

- Comments
- Directives
- Resource Records

### Directives

Los directives son definiciones de variables, y se definen con un **"\$"**. Hay 2 directives obligatorios:

- 1) **\$TTL**: Define un Time To Live predeterminado para todos los Resource Records.

Se puede poner un TTL a mano para los RR, pero si se deja en blanco, el directivo **"\$TTL"** va a ser el TTL de ese RR. El TTL es la cantidad de tiempo que será cacheado un RR. Si no se usa el RR en el tiempo especificado, es posible que se tenga que hacer otro lookup si se lo utiliza en el futuro.

- 2) **\$ORIGIN**: Permite establecer un punto de origen para las consultas de resolución de nombres dentro de una zona específica.

Una base de datos DNS puede tener **Fully Qualified Domain Name** (entero), o uno **relativo**.

Los FQDN tienen un **"."** al final. Un recurso con un FQDN puede ser accedido de forma global.

Ahora bien, DNS sólo lee FQDNs. Los DN relativos no tienen un punto al final, y se autocompletan con el valor de **\$ORIGIN**. Por ejemplo:

www IN A 192.168.29.200

mail IN A 192.168.29.201

En estos casos DNS entiende que estamos refiriéndonos a **www.\$ORIGIN**

Adicionalmente:

- Si se usa un **@** en el archivo, se reemplaza por el valor de **\$ORIGIN**.

Por otro lado, si se deja en blanco un DN en un RR, se reemplaza por el DN del último RR, pudiendo hacer múltiples líneas sin repetir el DN en cada una.

### Resource Records

Los principales tipos de RR son:



- **SOA** (Start of Authority): Define el servidor de nombres con autoridad en el Dominio, y sus características. No se puede definir más de un SOA por zone file.
  - **NS** (Name Server): Los RR de tipo NS guardan el nombre de un DNS autoritativo en un Dominio. Así, cuando se pida acceder a ese Dominio, sabremos a qué servidor hablar.
- Cuando se hace delegación de un subdominio, estos RR siempre deben estar acompañados de **Glue Records** (Registros A o AAAA), para que se pueda llegar a estos servidores.
- **MX** (Mail Exchange): Para correo.
  - **A** (IPv4) y **AAAA** (IPv6): *Host Record* - Datos autoritativos para todos los nodos en la zona.
  - **PTR**: (Para reverse mapping - falta explicarlo)

Sintaxis del registro SOA (y explicación de la base de todos):

```
domain-name TTL class-type record-type name-server e-mail-address serial-number refresh-time retry-time expiry-time negative-cache-TTL
```

Ejemplo:

```

1 @ IN SOA ns1.example.com. hostmaster.example.com. (
2     2023092500 ; serial number
3         12h ; refresh time
4         15m ; retry time
5         3w ; expire time
6         2h ; negative cache TTL
7 )
```

- 1) **domain-name**: En este ejemplo, ns1.example será el DN Server con autoridad sobre el DN "@".
- 2) **TTL**: Como dejamos el TTL vacío, se reemplaza por \$TTL. Esta entrada describe cuánto tiempo se guardará en el caché el Record.
- 3) **class-type**: Hoy en día, sólo se usa la clase IN. Existen CH y HS pero eran para testeos en MIT.
- 4) **record-type**: SOA
- 5) **name-server**: Notemos que ns1.example.com no es una dirección IP, y tendrá que ser linkeado a una IP en los record de tipo .A. adelante
- 6) **e-mail-address**: Define el email de administrador del Dominio. Se suele usar el email hostmaster, porque describe el uso. Notemos que el email no tiene @, esto es pq si no, se reemplazaría por \$ORIGIN. Entonces, se usa un "...en su lugar.
- 7) **serial-number**: El numero de serial es la fecha del último cambio que tuvo el servidor Master DNS. Los últimos dos dígitos es para el caso en que el servidor haya hecho cambios en su zone file más de una vez en un día.



- 8) **times** (todos): Los slave servers no crean los zone files de Dominios sobre los que no tienen autoridad. En cambio, le piden a los servidores master que les provean con el zone file. Lo hacen con frecuencia **refresh-time**. Si no recibe una respuesta del master, reintenta cada **retry-time**. El serial number marca la fecha de la versión del zone file, de forma tal que si hubo algún cambio, se haga un zone transfer, y asegura que si no lo hubo, no se envíe todo el zone file. Una vez expira el **expire-time**, el Slave no va a resolver más queries con este dominio. En este caso, se considera que la zona está "dead".

**negative-cache-TTL:** Supongamos que a un servidor DNS le piden la ip de "xyz.example.com.". Si este (de forma iterativa o recursiva) encuentra que este dominio no existe, entonces devuelve error. El negative-cache-TTL es el tiempo por el que se cachean las queries negativas. Esto hace que, en este caso por 2h, el slave no envíe más queries por xyz.example.com. al master.

### 8.1.3. Queries

La tarea principal de un servidor DNS es la de responder a **queries** de un resolver, o de un servidor DNS actuando en nombre de un resolver.

Un query se puede pensar como "Cuál es la dirección IP de www.example.com?".

Un servidor DNS puede recibir un query de **cualquier IP**. Para algunas IPs, tendrán autoridad o serán esclavos, y para otras deberán reenviar las queries, u otra combinación. En la mayoría de casos, un servidor recibe queries para dominios que no conoce, esto es, para los cuales no poseen zone files. Existen dos tipos de queries: Las queries recursivas, y las iterativas.

#### ▪ Queries recursivas

Una **query recursiva** es una query donde el servidor se encargará de que la consulta sea respondida completamente (o dará un error).

Una query recursiva tendrá una de 3 respuestas:

- 1) La respuesta al query, y posiblemente algún CNAME record (aliases) que puedan ser útiles, indicando si la información era autoritativa o cacheada.
- 2) Un error indicando que el dominio/host no existe.
- 3) Algún otro error de tipo temporal, como que no se pudo acceder al servidor DNS por errores de red.

En una query recursiva, el servidor, en nombre del cliente (sub-resolver), seguirá el camino del sistema DNS a través del universo para encontrar su respuesta.

## Recursive and Iterative Queries

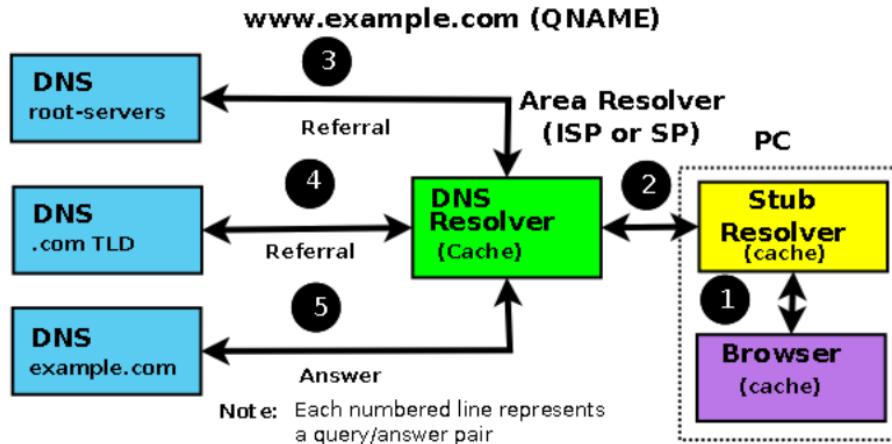


Figura 72: Query recursiva.<sup>3</sup>

La query en este ejemplo sigue los siguientes pasos:

- 1) El usuario escribe **www.example.com** en su navegador, que ejecuta una función llamando al **stub-resolver** local. (1)
- 2) El stub-resolver envía la consulta (query) '¿cuál es la dirección IP de **www.example.com**?' al **Resolver DNS** configurado localmente, pidiéndole, si está disponible, que sea recursiva. (2)
- 3) El Resolver DNS busca la dirección de **www.example.com** en su caché, si la encuentra, la devuelve directamente al stub-resolver, si no, sigue con el siguiente paso.
- 4) El Resolver DNS envía una consulta al root-server para la IP **www.example.com**. (3)
- 5) El root-server no sabe nada sobre **example.com**, y mucho menos sobre la parte **www**, pero sí sabe sobre el siguiente nivel en la jerarquía, **.com**, por lo que responde con una referencia apuntando a los servidores TLD para **.com**. (3)
- 6) El Resolver DNS envía la consulta a uno de los servidores de TLD **.com**. (4)
- 7) El servidor TLD sabe sobre **example.com**, pero no sobre **www**, por lo que nuevamente da una referencia, esta vez para el servidor DNS de **example.com**. (4)
- 8) El Resolver DNS envía nuevamente '¿cuál es la dirección IP de **www.example.com**?' a un servidor DNS de **example.com**.
- 9) El zonefile de **example.com** tiene un registro A (dirección IPv4), por lo que el servidor autoritativo para **example.com** devuelve el registro A de **www.example.com**. (5)



- 10) El Resolver DNS envía la respuesta “www.example.com=x.x.x.x” al stub-resolver y cachea la información. (2)
- 11) El stub-resolver cachea la información “www.example.com=x.x.x.x” y responde a la función llamada en 1. con esto. (1)
- 12) El navegador recibe la respuesta, coloca la información en su caché e inicia una sesión HTTP en la dirección x.x.x.x.

Los servidores root y TLD no soportan queries recursivas.

## ▪ Queries iterativas

Una **query iterativa** es una query donde el servidor DNS puede dar una respuesta parcial (es decir, un referral), o un error. Todos los servidores DNS soportan queries iterativas. Son simplemente queries DNS que no piden servicios recursivos, por lo que a veces se llaman no-recursivas.

Hay 4 posibles respuestas a una query iterativa:

- 1) La respuesta al query, y posiblemente algún CNAME record (aliases) que puedan ser útiles, indicando si la información era autoritativa o cacheada.
- 2) Un error indicando que el dominio/host no existe.
- 3) Algún otro error de tipo temporal, como que no se pudo acceder al servidor DNS por errores de red.
- 4) Una **referencia** (referral): Si el servidor no tiene la información pedida, devuelve el nombre y dirección IP de uno o más servidores DNS que estén más cerca del dominio consultado (en todos los casos en el nivel inmediatamente inferior en la jerarquía DNS). Esta referencia puede, como no, ser hacia un servidor autoritativo sobre el dominio objetivo.

## ▪ Queries inversas

Las queries inversas asocian un Resource Record con un dominio. Un ejemplo es “Cuál es el RR de este MX record”

Estas **no** sirven para encontrar un nombre desde una IP. Esto se llama **Reverse mapping/lookup**, y usan queries iterativas y recursivas con el Nombre de Dominio IN-ADDR.ARPA.

Las queries inversas no son obligatorias, y de hecho quedaron obsoletas.

### 8.1.4. Actualizaciones de zona

Cuando una zona tiene alguna actualización, el servidor maestro debe propagar esta información a todos los servidores DNS esclavos.



Debe considerarse que las **zone updates** son un posible vector de ataque, pues un esclavo puede recibir una actualización maliciosa.

Existen varias formas en que se realizaron actualizaciones de zona a lo largo del tiempo:

- **Full Zone Update (AXFR):** Originalmente, los esclavos le enviaban periódicamente (períodos determinados en el SOA RR) al master un query pidiendo su número Resource Record SOA. Si el número de serial del master es mayor al del esclavo, se pide un zone transfer (AXFR).
- **Incremental Zone Update (IXFR):** Como enviar todo un zonefile es costoso, se implementaron los zone updates incrementales. Estos permiten al esclavo y maestro enviarse únicamente los RR que hayan cambiado.

En este caso, cuando el serial sea diferente, si ambos lados pueden utilizar IXFR, se hace el zone update incremental. Este se hace en el puerto TCP 53, mientras que el resto de operaciones DNS se hacen en el puerto UDP 53.

- **Notify (NOTIFY):** El estándar que vimos hasta ahora recomienda que haga refresh intervals de 12 horas. Esto significa que un servidor esclavo puede tener información desactualizada por hasta 12 horas. En entornos dinámicos esto puede ser inaceptable.

Notify consiste en que el servidor maestro notifique a los esclavos que un cambio puede haber ocurrido en los records de dominio. Esto ocasiona que el esclavo pida el RR SOA del maestro, y si el número de serial del maestro es mayor, incia un AXFR o un IXFR.

### Actualizaciones dinámicas

Normalmente, cuando se necesita actualizar algún RR, se edita el archivo de zona manualmente y se reinicia el servidor DNS. Sin embargo, cuando los archivos de zona son muy grandes o se necesitan realizar demasiados cambios, esto resulta imposible de mantener.

Por lo tanto, se utilizan **actualizaciones dinámicas**, que permiten al servidor mantener sus operaciones de forma continua mientras se realizan los cambios.

*Acá salteo una pequeña explicación de cómo se hacen*

### 8.1.5. Seguridad - Amenazas y clasificación

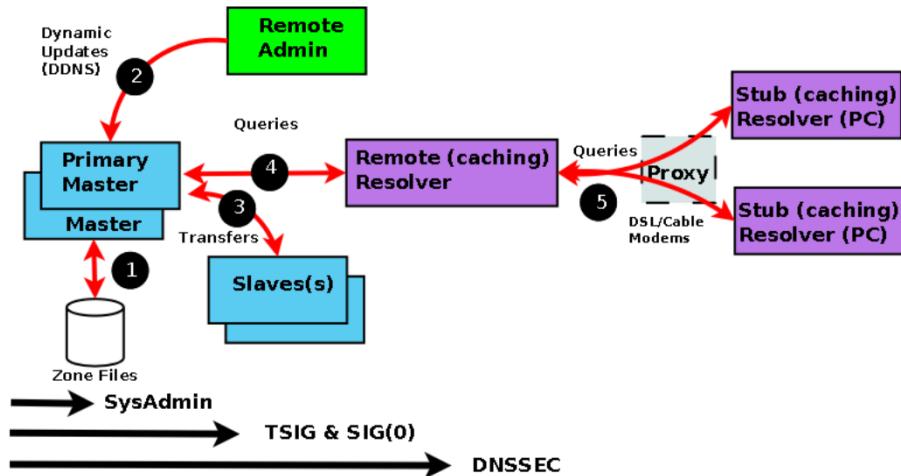


Figura 73: Vectores de ataque en DNS.<sup>3</sup>

#	Área	Amenaza	Tipo
1	Zone Files	Corrupción de archivos	Local
2	Actualiz. dinámicas	Updates no autorizadas, spoofing	Server-Server
3	Zone Transfers	Spoofing	Server-Cliente
4	Queries Remotas	Spoofing, eavesdropping, NS comprometido	Server-Cliente
5	Queries de Resolutor	Eavesdropping, cache envenenado, spoofing	Cliente-Cliente

Mientras más se aleja del master, más complicada es la solución.

Tenemos tres tipos de amenazas que nos interesan: **Locales**; las amenazas **Servidor-Servidor**; y las amenazas **Servidor-Cliente**.

Protegiéndose contra amenazas:

- **Locales**: Para evitarlas, debe asegurarse que los archivos de zona (y named.conf) estén guardados de forma segura, y que se tengan copias de seguridad de todos ellos.
- **Servidor-Servidor**: BIND provee Access Control Lists (ACLs), que provee protección básica de direcciones IP. Este puede ser burlado utilizando IP spoofing, pero es mejor que nada. En el caso de los zone transfers, también se puede solucionar teniendo únicamente masters, pero tendrían que sincronizarse todos sus zone files.
- **Servidor-Cliente**: En estos casos, DNSSEC utiliza un sistema de llaves públicas y privadas para la autenticación.

### 8.1.6. Reverse mapping

Una query DNS normal tiene la forma “Cuál es la IP del host www en el dominio example.com”. Sin embargo, hay veces que queremos saber “Cuál es el nombre del host con la ip x.x.x.x”. Esto es

lo que se conoce como **reverse mapping**. El reverse mapping no es obligatorio en un servidor DNS, y para llevarlo a cabo, se utiliza un nombre de dominio reservado **IN-ADDR.ARPA**, para IPv4, y **IP6.ARPA**, para IPv6.

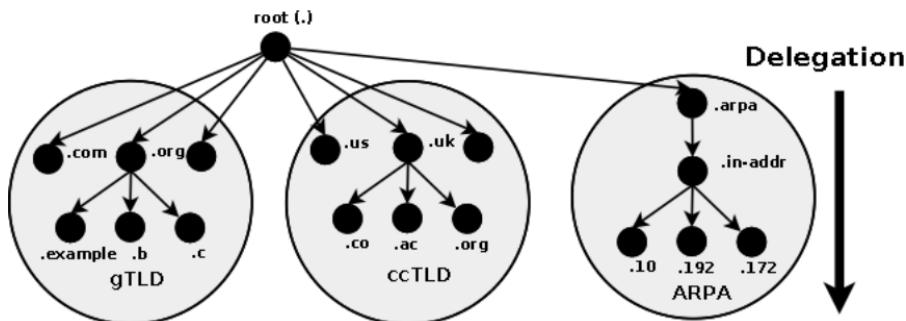


Figura 74: Reverse mapping IN-ADDR.<sup>3</sup>

## Ejemplo de Reverse Mapping

Si tenemos un host de nombre bill.example.com, con ip 192.168.23.17, primero debemos notar que es una IP de clase C. Por lo tanto, la IP de la red es 192.168.23. Luego, la invertimos y agregamos el dominio reservado y tenemos que la IP del dominio IN-ADDR.ARPA será: 23.168.192.IN-ADDR.ARPA. El zone file se verá de la siguiente forma

```
1 $TTL 2d ; 172800 seconds
2 $ORIGIN (24/)23.168.192.IN-ADDR.ARPA.
3 @ IN SOA ns1.example.com. hostmaster.example.com. (
4 ; serial number
5 ; refresh
6 ; update retry
7 ; expiry
8 ; nx = nxdomain ttl
9 )
10 IN NS ns1.example.com.
11 1 IN PTR www.example.com.
12 .....
13 17 IN PTR bill.example.com.
14 .....
```

En el caso de las direcciones IPv6, se revertirá cada dígito y se separará con un “:”:

2001:0db8::1 ⇒ 1.0.0.0, 0.0.0.0, 0.0.0.0, 0.0.0.0, 0.0.0.0, 0.0.0.0, 8.b.d.0, 1.0.0.2, IP6.ARPA

### 8.1.7. Vistas

Las vistas permiten mostrar a la máquinas una visión distinta de la jerarquía de nombres DNS de la que se ve desde el exterior. Por ejemplo le permite revelar todos los host a los usuarios internos pero

restringir la vista externa a unos pocos servidores de confianza. O podría ofrecer los mismos hosts en ambas vistas pero proporcionar registros diferentes a los usuarios internos.

#### 8.1.8. Configuraciones DNS

Sabemos que los servidores DNS pueden, para unas zonas, esclavos, maestros, o pueden dar punteros a servidores con autoridad sobre estas. El tipo que tiene un servidor para cada zona está determinado en el archivo named.conf, y está en el parámetro de la zona “type”, tomando los valores master, slave, stub, forward y hint.

En esta sección, explicaremos el funcionamiento de cada tipo de servidores.

##### 1) Servidores Maestros

Los **servidores maestros** definen zone files para los cuales su DNS es **autoritativo** (type ‘master’). La zona ha sido delegada a este DNS (mediante un RR).

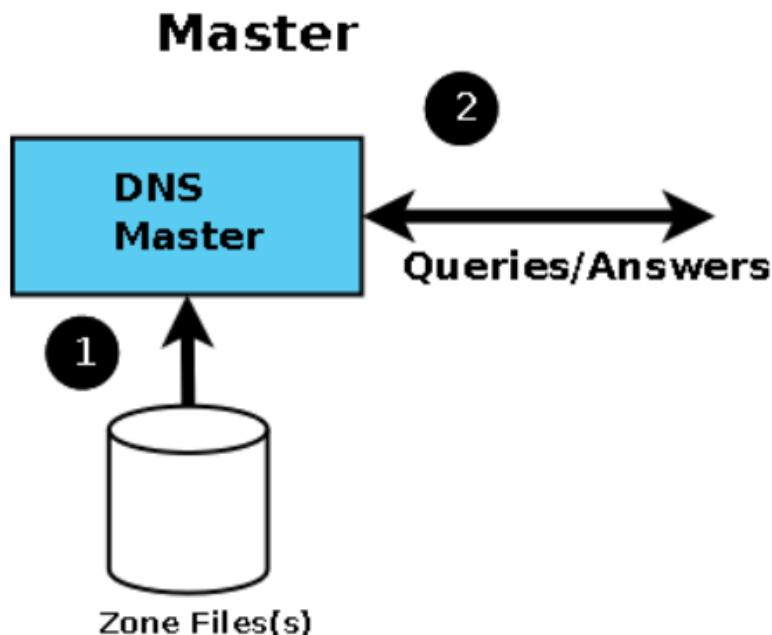


Figura 75: Servidor maestro.<sup>3</sup>

El estado de Maestro se define en BIND incluyendo la línea “type master” en la declaración de la zona en el archivo named.conf:

```

1 zone "example.com" in {
2     type master;
3     file "/etc/bind/db.example.com";
4 }
  
```

Se puede además añadir la opción de NOTIFY, para notificar las actualizaciones de zona, con la línea “also-notify 192.168.0.2; . . . ;”, o apagar todas las operaciones NOTIFY con “notify no;”.

## 2) Servidores Esclavos

Un Esclavo DNS obtiene su información de la zona a través de un zone transfer, y va a responder como un servidor autoritativo para las zonas en las que esté definido como esclavo. No es posible distinguir que una query fue respondida por un maestro o esclavo.

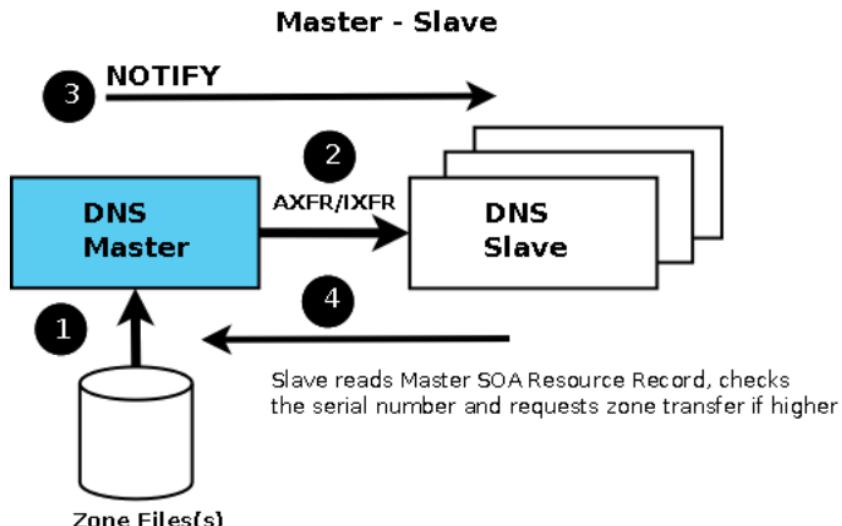


Figura 76: Los servidores esclavos reciben sus zone-files a partir de un zone-transfer.<sup>3</sup>

El archivo named.conf en este caso se verá así:

```

1 zone "example.com" in {
2     type slave;
3     masters { 192.168.23.17; ...;};
4     file "/etc/bind/db.example.com";
5 };

```

Un esclavo puede recibir sus archivos de zona de otro esclavo. Esto puede ser útil en caso que se desee ocultar el servidor maestro, lo cual se llama una **configuración stealth**. En este caso, se debe colocar la IP del servidor esclavo del cual obtiene sus archivos de zona, y este lo verá como un “maestro”. Además, por defecto, este último enviará mensajes NOTIFY cada vez que reciba un zone file del verdadero master.

## 3) Resolvers o Servidores de Cacheo

Un **Resolver DNS** es un servidor que recibe información de otros servidores en respuesta a queries que hagan los hosts, y luego la guarda (cachea) localmente.

Cuando el Resolver responde a una query con datos que tenía cacheados, la respuesta que dará será de tipo ‘no-autoritativa’, si los obtiene de un servidor master, será ‘autoritativa’.

Un servidor que provee servicios de cacheo debe también proveer queries recursivas, y a su vez, para proveer queries recursivas se necesita acceso a los root servers, el cual es proveído utilizando la línea “type hint” en named.conf.

#### 4) Servidores de Forwarding/Proxy

Un **Servidor de Forwarding** es uno que se limita a simplemente redirigir queries a otro DNS y cachear los resultados.

Su utilidad radica en el hecho que pueden ayudar a evitar utilizar redes externas cuando estas son lentas o caras. Permiten reducir la latencia y la saturación de las redes,

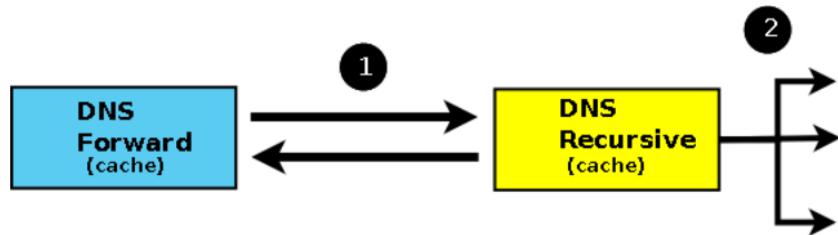


Figura 77: Servidor Proxy.<sup>3</sup>

En BIND, se implementan en la sección de opciones con las líneas:

```

1 options {
2     directory "/var/named";
3     forwarders {10.0.0.1; 10.0.0.2; ...};
4     forward only;
5 };

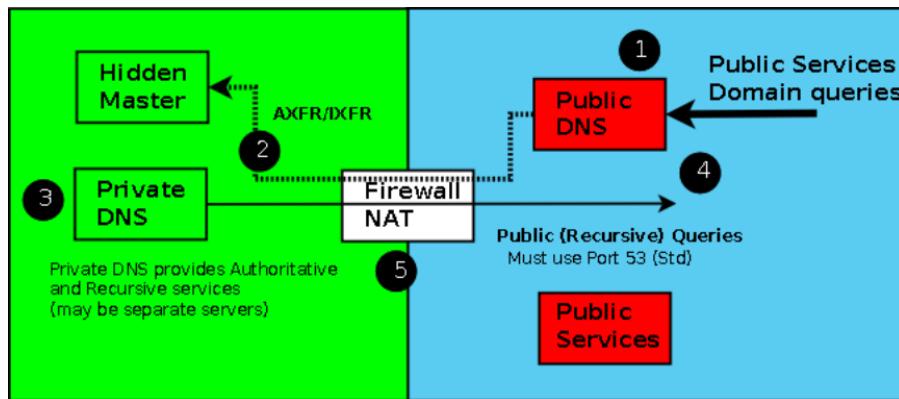
```

Donde forward toma los parámetros first y only, que indican si sólo van a enviar queries a los forwarders, y si no hay respuesta entonces no responden, o si, en caso de que no haya respuesta, intentarán responder el query.

#### 5) Servidores Stealth (/DMZ/Maestro oculto)

Un **Servidor Stealth** es un servidor tal que *no es visible en ningún NS Record público*. Se define por las siguientes características:

- Se necesita un DNS Público para acceder a los servicios públicos (web, mail, etc...)
- Se desea ocultar los hosts internos en una red ante queries o en caso que el DNS público sea comprometido.

Figura 78: Configuración de los servidores stealth.<sup>3</sup>

Los servidores externos sólo proveen respuestas autoritativas, y no tienen caching (no aceptan queries recursivas). El archivo de zona del DNS Público tendría únicamente los RR que se desea que sean visibles públicamente. Estos servidores **no deben** aceptar ni enviar transferencias del Servidor Stealth.

Archivos de zona minimalistas para cada servidor pueden ser los siguientes:

- DNS Público

```

1 example.com. IN SOA ns.example.com. root.example.com. (
2           2003080800
3             3h
4           15m
5             3w
6             3h
7           )
8   IN NS ns1.example.com.
9   IN NS ns2.example.com.
10  IN MX 10 mail.example.com.
11 ns1   IN A 192.168.254.1
12 ns2   IN A 192.168.254.2
13 mail  IN A 192.168.254.3
14 www   IN A 192.168.254.4

```

- Stealth Server

```

1 ; private zone master file used by stealth server(s)
2 ; provides public and private services and hosts
3 example.com. IN SOA ns.example.com. root.example.com. (
4           2003080800
5             3h
6           15m

```



```

7      3w
8      3h
9
10     IN  NS   ns1.example.com.
11     IN  NS   ns2.example.com.
12     IN  MX   10   mail.example.com.
13 ; public hosts
14 ns1      IN  A    192.168.254.1
15 ns2      IN  A    192.168.254.2
16 mail     IN  A    192.168.254.3
17 www      IN  A    192.168.254.4
18 ; private hosts
19 joe      IN  A    192.168.254.5
20 ....
21 accounting IN  A    192.168.254.28
22 payroll   IN  A    192.168.254.29

```

Adicionalmente, utilizando **views** en bind, es posible disminuir más la visibilidad del Servidor Stealth, redirigiendo todos los queries DNS internos al servidor externo.

## 6) Servidores Sólo Autoritativos

Los **Servidores Sólo Autoritativos** son servidores que dan respuestas autoritativas (son maestros o esclavos para alguna(s) zona(s)), pero no cachean.

En general, estos sólo se usan como el servidor público en un sistema Stealth DNS, para proveer seguridad, o en servidores DNS de alta performance.

Para implementarlo, en el apartado options se agrega la línea “recursion no;”

## 7) Servidores Split Horizon

Los **Servidores Split Horizon** son servidores DNS que dan respuestas diferentes según la dirección IP de origen de la query, u otras características. Se suele utilizar para crear divisiones geográficas o balancear la carga computacional.

Los Servidores Split Horizon se implementan utilizando views.

*Son salteadas las unidades 5 (Instalación de BIND), 6 (Ejemplos de uso de BIND - Archivos de zona) y 7 (Estructura de los archivos named.conf).*

## 8.2. Firewall

La seguridad resulta cada vez más importante tanto para individuos como compañías, pues el internet, con todas sus utilidades, trae varios riesgos.



### 8.2.1. Métodos de ataque

1) **Acceso no autorizado:** Cuando alguien que no debería tener acceso a algún servicio se conecta y lo usa. Es posible protegerse especificando cuidadosamente quienes tienen acceso a estos servicios.

2) **Explotación de debilidades en programas:** Hay programas que no fueron diseñados con su seguridad en mente. Para protegerse, no deben utilizarse programas no seguros. En general, instalar y utilizar únicamente los servicios estrictamente necesarios es una buena práctica.

Se puede utilizar netstat para ver qué puertos están siendo utilizados por los programas para conocer todos los posibles vectores de ataque, y en general, se desea minimizar la cantidad.

3) **Denial of service (DDoS):** Un tipo de ataque que causa que un servicio o programa deje de funcionar, o imposibilita que otros usuarios lo utilicen. Puede lograrse desde el nivel de red enviando paquetes maliciosos que causan que la conexión falle, o desde el nivel de aplicación, enviando comandos que detengan el funcionamiento del programa.

Se puede evitar impidiendo que tráfico sospechoso llegue a los hosts, o que comandos sospechosos sean enviados al programa.

4) **Spoofing:** Consiste en que un host o aplicación intentan hacerse pasar por otro.

Se puede evitar con métodos de verificación de la autenticidad de paquetes, o filtrando los paquetes que tengan IPs de origen inválidas. También puede ayudar a evitar este tipo de ataques cuando un SO utiliza mecanismos de control impredecibles, como el número de secuencia de paquetes TCP en Linux.

5) **Eavesdropping (escucha no autorizada):** Un tipo de ataque donde un host simplemente escucha todos los paquetes que circulan por la red, capturando datos que no le pertenecen. Estos datos pueden ser información sensible como contraseñas.

Una forma de evitarlo es evitando el uso de redes de broadcast.

### 8.2.2. Firewalls

Finalmente, definamos qué es un Firewall.

Un **Firewall** es un host confiable y seguro que actúa como un nodo que conecta dos redes. Todo el tráfico entre estas redes deberá entonces pasar por el firewall, y este tendrá un conjunto de reglas para determinar si el tráfico será permitido (**allowed**), bloqueado (**blocked**, sin dar una respuesta), o rechazado (**rejected**, notificando al origen).

Los firewalls, sin embargo, pueden tener diferentes formas. Los más complejos incluyen una serie de hosts separados, llamados “perimeter network” o “Demilitarized Zone (DMZ)”. Dos hosts actúan como un “filtro” que sólo deja pasar ciertos tipos de tráfico, y entre estos existen servidores de la red como el de email o un proxy de www.

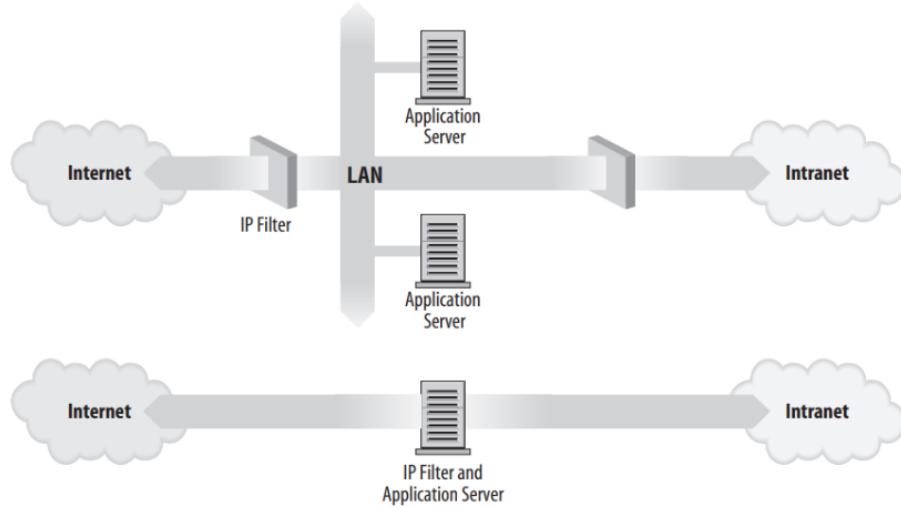


Figura 79: Las dos principales clases de diseños de firewalls.<sup>4</sup>

El kernel de Linux provee funcionalidades que le permiten actuar como firewall IP, haciendo procesamiento de paquetes IP que permite utilizar ciertos filtros.

Las 3 principales clases de procesamiento de paquetes son:

- **Filtering:** Consiste en decidir si se permite que un paquete pase por el firewall o no.
- **Mangling:** Consiste en modificar los paquetes que pasan.
- **Network Address Translation (NAT):** Consiste en modificar las IPs de origen o destino de los paquetes que pasan.

El firewall que provee Linux se llama además **stateful**, que quiere decir que este puede seguir el estado de conexiones activas en la red, así pudiendo monitorear si algún paquete que pase por él fuera de una conexión inválida (INVALID), o aceptando un paquete en base al hecho que pertenece a una conexión establecida (ESTABLISHED), por ejemplo.

### 8.2.3. IP Filtering

El IP Filtering es un mecanismo que decide qué tipos de paquetes IP serán procesados normalmente, y cuáles droppeados (eliminados e ignorados) o rechazados. Esto se puede hacer utilizando múltiples criterios:

- 1) Tipo de protocolo - UDP, TCP, ICMP, etc...
- 2) Número de puerto (Para TCP y UDP).
- 3) Tipo de paquete - SYN/ACK, data, ICMP Echo Request, etc...
- 4) Dirección IP de origen.



## 5) Dirección IP de destino.

En general, el IP Filtering combina varias de estas técnicas para sólo permitir los datos que se desean.

Cabe destacar que el IP filtering se hace al nivel de capa de red. Por lo tanto, no sabe nada acerca de las aplicaciones conectadas, sólo sabe sobre la conexión.

### 8.2.4. IP Tables

#### Tablas

La arquitectura IP tables agrupa las reglas del procesamiento de los paquetes en 3 tablas, cada una con sus propias cadenas:

- **filter:** Se utiliza sólo para filtrar paquetes, con ACCEPT, REJECT (rechazar, notificando al origen), DROP (desechar), o LOG (registrar el suceso). Tiene 3 cadenas, INPUT, OUTPUT, y FORWARD.
- **nat:** Se usa para traducir direcciones de red: los paquetes que pasan por esta tabla ven sus direcciones modificadas. Se usa en las cadenas PREROUTING, POSTROUTING y OUTPUT.
- **mangle:** Se usa para modificar paquetes (excepto sus direcciones). Puede utilizarse en todas las 5 cadenas.

#### Reglas

Las **reglas** consisten en un **match**, que determina a qué paquetes afecta la regla, y un **objetivo**, que especifica qué se hará con estos paquetes. En el archivo de IP tables, una regla tendrá la siguiente forma:

```
$F [-t table] comando [+chain] [match] [objetivo/salto]
(siendo F = /sbin/iptables el archivo donde se guarda esta información)
```

#### Puntos de enganche

En iptables se definen 5 **puntos de enganche**: PREROUTING, INPUT, FORWARD, OUTPUT y POSTROUTING.

- **FORWARD:** Permite procesar paquetes que están en medio del paso de la interfaz de entrada y la de salida.
- **INPUT:** Procesa paquetes con destino local, es decir, dirigidos al firewall.
- **OUTPUT:** Procesa paquetes que salen del firewall.
- **PREROUTING:** Procesa paquetes apenas llegan a la interfaz.
- **POSTROUTING:** Procesa paquetes antes de que salgan por la interfaz.



## Comandos

Los comandos especifican qué se hace con esta y las demás reglas de la IP table. Los más comunes son:

- 1) **-A** (**--append**) agrega una la regla a la chain. (`$F -A FORWARD -i eth0 -o eth1 -j REJECT`)
- 2) **-D** (**--delete**).
- 3) **-P** (**--policy**) determina una política por defecto de una cadena. Es normal por ejemplo utilizarla si tenemos un firewall que queremos que rechace todos los paquetes que no sean para un puerto específico, o si sólo queremos aceptar paquetes de un conjunto de IPs confiables.
- 4) **-I** (**--insert**) inserta la regla en una posición específica, cosa que importa ya que las reglas se realizan en orden (puede afectar si por ejemplo se hace un NAT antes de un filtering).
- 5) **-R** (**--replace**) se puede hacer utilizando el índice o especificando la regla a reemplazar.

Y hay muchas más, como **-list**, que permite listar las reglas, o **-flush**, que elimina todas las reglas, ambas pudiendo especificar si a todas o a sólo una cadena.

## Matches

En los matches, el **!** marca la negación de la condición, y va entre el tipo de match y el argumento. Existen 5 tipos:

- 1) **Matches genéricos:** Una comparación que siempre está disponible.
  - a) **-p** (**--protocol**) Matchea a un tipo de protocolo (tcp, udp, icmp, etc...).
  - b) **-s** / **-d** (**-src** / **-dst**) Filtra según dirección de origen/destino, y puede utilizarse [/mask] para filtrar un conjunto de direcciones.
  - c) **-i** (**--in-interface**).
  - d) **-o** (**--out-interface**).
  - e) **-f** (**--fragment**) chequea una parte de un paquete fragmentado.
- 2) **Matches TCP:** Está sólo disponible cuando se trata de un paquete TCP/UDP.
  - a) **--sport** / **--dport** Filtra por un puerto de origen/destino.
  - b) **--tcp-flags** Filtra paquetes que llevan una bandera tcp prendida: ACK, SYN, FIN, URG, PSH, FIN, RST.
  - c) Otros...
- 3) **Matches UDP.**
  - a) **--sport** / **--dport** Filtra por un puerto de origen/destino.



#### 4) Matches ICMP.

a) `--icmp-type` Matchea el tipo del mensaje, con su nombre o número.

#### 5) Matches especiales.

- a) `--limit` Pone un límite a la cantidad de veces que se aceptan mensajes que pasan cierto filtro en un período de tiempo.
- b) `-m multiport` Permite agregar múltiples puertos en los comandos `-s/dports`.
- c) `--mac` Filtra según dirección mac de origen.
- d) Otros...

Ejemplo de combinación de matches

```
-m multiport -o eth2 -d 181.16.1.18 -p tcp -dports 80,443
```

#### Targets/saltos

- 1) **Target ACCEPT/REJECT/DROP:** Aceptan, rechazan o descartan un paquete.
- 2) **Target LOG:** Guarda un registro de algún tipo de información deseada (tiene parámetros para especificar qué información registrar, como `--log-prefix`, `--log-tcp-options`, etc...)
- 3) **Target SNAT/DNAT:** Modifica direcciones ip de los paquetes de origen y destino respectivamente. Estas pueden únicamente usarse en la tabla NAT y las cadenas POSTROUTING, para SNAT, y PREROUTING y OUTPUT, para DNAT. Tienen el argumento `--to-source` y `--to-destination`, y puede ponerse tanto una dirección IP como puerto.
- 4) **Target MASQUERADE:** Hace lo mismo que SNAT, pero sólo debe usarse para direcciones IP asignadas dinámicamente, en las que, antes de conectarnos con el ISP, no sabremos qué dirección se tendrá. Se cambia la dirección de origen de un paquete según su NIC (tarjeta de red).
- 5) **Target REDIRECT:** Redirige paquetes hacia el firewall. Con `--to-ports` se especifica a qué puertos enviar los paquetes.
- 6) **Target TOS:** Cambia el ToS de un paquete.

## 9. Conceptos adicionales

Esta sección corresponde a conceptos adicionales que formaron parte del cursado, y no añadí a ninguna de las secciones anteriores para no perder el orden de los temas dados en los libros, que lo hace más dinámico y fácil de seguir.



### 9.0.1. Hardware de red

- **Repetidor:** Dispositivo de capa 1 del modelo OSI, que amplifica las señales físicas de la transmisión.
- **Hub:** Dispositivo de capa 1, es un repetidor con múltiples puertos. Se limita a propagar la señal que ingresa por uno de sus puertos a todos los demás. Simplemente interconecta interfaces de red, físicamente es como si estuvieran “soldados” las diferentes conexiones a sus puertos.
- **Switch:** Es un dispositivo de capa de enlace. Permite la interconexión de hosts (tal como el hub) pero sin dividir el ancho de banda del canal, ya que tiene un panel trasero de alta velocidad que direcciona las tramas. Además, las tramas recibidas por una interfaz no se replican en todas las demás, sino que se envían por la interfaz adecuada para arribar al destino (si se conoce). De esta forma, los dominios de colisión son independientes entre puertos y se puede lograr una mayor eficiencia de la capacidad del canal.

Los switches, además, **proveen seguridad** de varias formas: el direccionamiento logra que los mensajes con destino en un único nodo no sean vistos por todos los demás en la LAN. Pero también es capaz de proteger datos en el caso de broadcasts, con el uso de VLANs (si es un switch que las admite), que permite dividir una red LAN en varias LANs virtuales, donde los broadcasts se enviarán de forma local para cada una. Finalmente, hay switches que tienen **ACLs (Access Control Lists)**, que, similar a un firewall simple, permiten filtrar paquetes entrantes en base a criterios como dirección IP de origen o destino, puerto, entre otros.

- **Bridge/Puente:** Dispositivo de capa 2, permite conectar redes LAN heterogéneas, adaptando las tramas enviadas por las mismas, y tiene las funcionalidades de un switch. Por ejemplo, podría conectarse un enlace de fibra con uno UTP.

### Conectando dos redes

Los switches (y bridges) permiten conectar redes con diferentes tecnologías (10BaseT/100BaseT). Sin embargo, esto sólo vale si se trata de redes con el mismo MTU. Si dos redes tienen MTUs diferentes, se necesita utilizar un router, que son los dispositivos que se encargarán de fragmentar o enviar el mensaje ICMP MTU Exceeded al origen para su fragmentación. Los switches no tienen esta funcionalidad.

### UTP vs Fibra óptica

La fibra óptica es, hoy en día, más económica, resistente a la interferencia, y tiene un mayor rango y ancho de banda. Sin embargo, UTP sigue siendo la opción más común para interconectar redes LAN. Esto se debe a varios factores. Entre ellos, que UTP es una tecnología más familiar y menos sensible a roturas que la fibra óptica. Además, el costo de instalación y la tecnología que se necesita para instalar una red con cables de cobre es más económica, y en general, en la mayoría de redes de hogares, no se aprovecha la totalidad del ancho de banda que en teoría haría a la fibra óptica una opción más atractiva.



## UTP vs Coaxial

Como ya vimos, los cables coaxiales son más resistentes, tanto a la interferencia como roturas, y tienen un mayor ancho de banda y rango. Sin embargo, el cable coaxial tiene un coste significativamente superior al UTP.

### 9.1. VLANs

En un principio, todas las computadoras de una oficina o la mayoría de ellas pertenecían a una misma LAN.

Las **VLANs** permiten a las empresas separar cada host de la red por sector de organización, es decir, contabilidad, administración, investigación, etc. Esto incrementa la seguridad, al ocultarse los mensajes de una VLAN de las demás, y la eficiencia, pues los mensajes de difusión serían enviados sólo a la VLAN a la que pertenece un host.

Las VLANs serán manejadas por el administrador de red, configurando un switch para definir a qué VLAN pertenece cada puerto.

### 9.2. BOOTP

**BOOTP** es un protocolo de resolución inversa de direcciones (encontrar IPs a partir de MACs). La resolución inversa de direcciones permite una gestión centralizada de las configuraciones, y se apoya en un servidor (escuchando y respondiendo pedidos de este tipo) que almacena las correspondencias MAC-IP.

#### Funcionamiento

- 1) Cuando el cliente arranca envía un “*BOOTP Request*” a la dirección broadcast poniendo como dirección de origen 0.0.0.0 (aún no sabe cuál es su ip).
- 2) El servidor recibe el mensaje, busca en su tabla la MAC del solicitante y si la encuentra se prepara para responder con un “*BOOTP Reply*” con la correspondiente IP.
- 3) Como el servidor no puede enviar la respuesta por unicast porque su ARP cache no tiene una entrada con la IP del cliente (mandar un ARP Request no sirve porque el cliente no sabe cuál es su IP), manda el “*BOOTP Reply*” con la respuesta en broadcast también.

Hay otra alternativa donde el servidor tiene privilegios de completar la ARP cache para introducir una nueva entrada y responder directamente al cliente.

Los mensajes BOOTP Request y Reply se envían dentro de paquetes IP/UDP (por lo tanto pueden atravesar routers, entonces servidor y cliente pueden estar en distintas redes).



## A. Bibliografía

1. Redes de computadoras. Andrew S. Tanenbaum, David J. Wetherall. 5ta Edición.
2. Comer D.E. - Internetworking with TCP\_IP\_ Principles, Protocols, and Architecture. Volume 1
3. [DNS for Rocket Scientists](#)
4. Linux Network Administrators Guide. Olaf Kirch