

# **Fundamentos de Redes de Comunicaciones**

Volumen 1

Luciana B. Falcon

3 de agosto de 2025

# Índice

<b>1. Redes de computadoras y Modelos de Comunicaciones</b>	<b>2</b>
1.1. Redes de Computadoras . . . . .	2
1.2. Performance . . . . .	7
1.3. Modelos de Comunicaciones . . . . .	8
1.4. Encapsulamiento . . . . .	9
1.5. Aplicaciones y Ejemplos . . . . .	10
<b>2. Nivel de Aplicación</b>	<b>14</b>
2.1. Protocolo HTTP . . . . .	16
2.2. Protocolo SMTP . . . . .	22
2.3. Protocolo DNS . . . . .	25
2.4. Aplicaciones y Ejemplos . . . . .	34
<b>3. Nivel de Transporte</b>	<b>56</b>
3.1. Protocolo UDP (User Datagram Protocol) . . . . .	58
3.2. Protocolo TCP (Transmission Control Protocol) . . . . .	61
3.3. TCP Performance . . . . .	70
3.4. Aplicaciones y Ejemplos . . . . .	82
<b>4. Arquitectura IP</b>	<b>90</b>
4.1. Protocolo IP (Internet Protocol) . . . . .	90
4.2. Configuración de direcciones - NAT - Fragmentación - ICMP . . . . .	97
4.3. Subnetting - Máscara Fija - Máscara Variable . . . . .	104
4.4. IPv6 . . . . .	107
4.5. Aplicaciones y Ejemplos . . . . .	116
<b>5. Bibliografía</b>	<b>130</b>
<b>6. Apéndice</b>	<b>131</b>

# 1. Redes de computadoras y Modelos de Comunicaciones

## 1.1. Redes de Computadoras

Es un conjunto de computadoras interconectadas con el objeto de intercambiar información. La interconexión puede realizarse mediante un simple cable trenzado o a través de redes más complejas, como por ejemplo Internet.

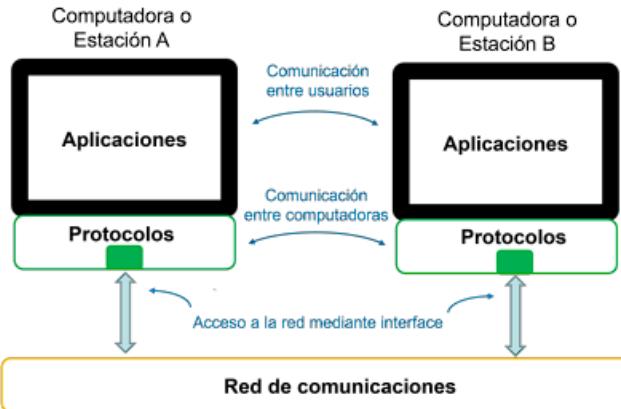


Figura 1: Interconexión entre computadoras.

### Componentes físicos y lógicos

Componentes físicos y lógicos de una red:

- Recursos físicos:
  - Medios: fibras ópticas, satélites, etc.
  - Interfaces de red.
  - Adaptadores.
  - Enlaces.
  - Recursos: ancho de banda.
- Equipos terminales o hosts:
  - Ejecutan las aplicaciones.
  - Se encuentran en el borde de la red.
- Switches y routers:
  - Despachan paquetes de datos.
  - Interconectan redes.
- Redes:
  - Interconexión de enlaces, switches y/o routers.

- Habilitan servicios y aplicaciones.
- Pueden ser propias y administradas por una organización.

Las redes, como infraestructura, habilitan servicios de consumo masivo, streaming, intercambio uno-a-uno o multiusuario, así como redes privadas virtuales.

## Espectro electromagnético

Es el conjunto de todos los tipos de radiación que se desplazan en ondas al conjunto de todas las ondas electromagnéticas. Las redes inalámbricas (Wi-Fi, celular, Bluetooth, satélite, etc.) transmiten datos usando una parte específica de ese espectro.

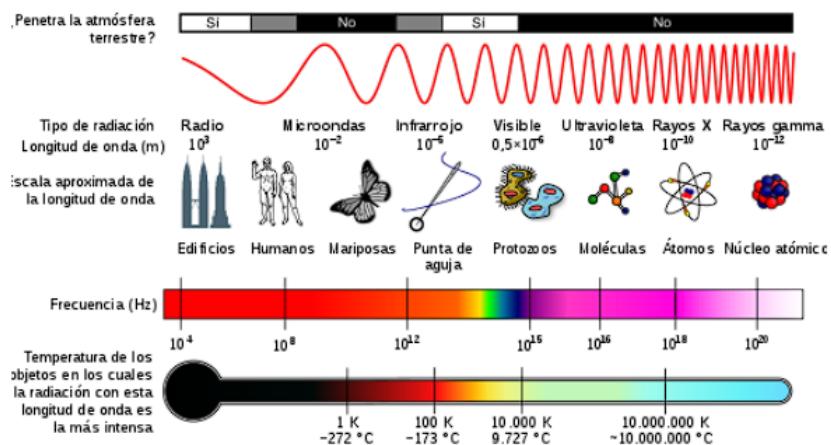


Figura 2: Espectro electromagnético.

## Definición de Protocolo

Es un conjunto de reglas o convenciones utilizadas para controlar la transferencia de datos en una red de computadoras, cuyos organismos de estandarización son:

- ITU: International Telecommunication Union. Ej: X.509 (certificados de clave digital).
- IEEE: Institute of Electrical and Electronics Engineers. Ej: 802.3, 802.11 (redes LAN).
- Internet Society IETF: Internet Engineering Task Force. Ej: RFC 790 (Protocolo IP).
- etc.

## Redes de Acceso

Las redes de acceso pueden ser de acceso residencial, institucional o móviles.

Consideraciones importantes:

1. **Velocidad de transmisión del acceso:** se refiere a cuán rápido podés enviar y recibir datos a través de la red.
2. **Servicio dedicado o compartido:** indica si la conexión es solo para uso personal o si es la compartida con otras personas.

Las consideraciones son aspectos clave para evaluar o comparar el tipo de conexión.

### **Formas de conexión (o medios de acceso a Internet)**

1. **Acceso Cablemódem:** Utiliza la red de cable coaxial (la misma que se usa para televisión por cable) para brindar acceso a Internet.
2. **xDSL:** Digital Subscriber Line: Usa las líneas telefónicas tradicionales de cobre para transmitir datos de alta velocidad.
3. **PON:** Passive Optical Network: Es una tecnología de fibra óptica para ofrecer conexiones de Internet de alta velocidad.
4. **Acceso por red celular 4G/5G:** Utiliza redes móviles para ofrecer acceso a Internet de antenas de compañías telefónicas.
5. **Acceso por red WiFi:** Proporciona acceso inalámbrico a Internet a través de redes locales (LAN inalámbricas) de un router o punto de acceso WiFi.
6. **SOHO:** No es un medio de conexión, sino un tipo de red pequeña para oficinas o negocios desde el hogar que puede usar uno o varios de los medios anteriores.

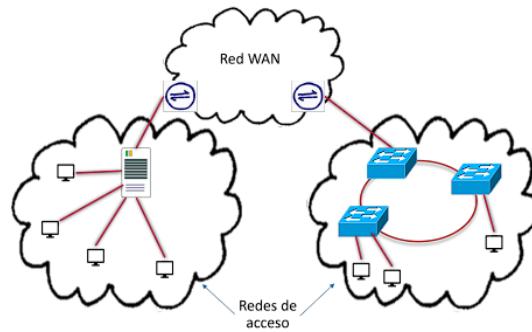


Figura 3: Redes de acceso.

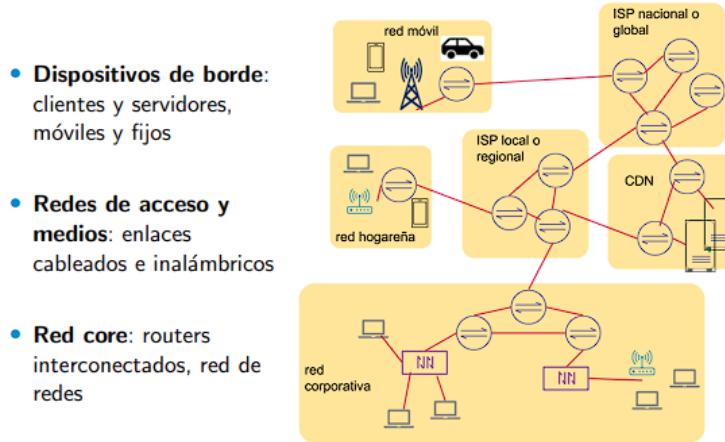
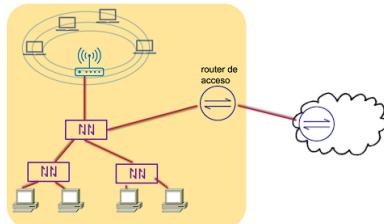


Figura 4: Estructura de Internet.



- La red LAN interna es una combinación de redes Ethernet
  - cableadas: 100, 1000 Mbps y más
  - y no cableadas (Wifi)
- Uno o más routers de borde proveen los accesos

Figura 5: Redes LAN corporativas u otras organizaciones.

- A su vez, Internet, como red de redes, también es una aglomeración de redes de acceso, de proveedores de servicios, de datacenters, de redes de contenido, etc.
- Pueden ser públicas o privadas.
- Sus conexiones se producen bajo acuerdos privados o bien en sitios de intercambio mutuo.
- *Continuará*

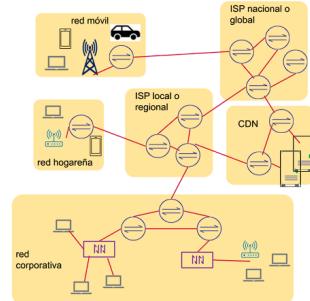


Figura 6: Internet como Red de redes.

Figura 7: Redes públicas vs redes privadas.

## Bits y Paquetes de Datos

La información que producen e intercambian las aplicaciones se fracciona en bloques de datos, genéricamente llamados paquetes que tienen una cantidad finita de bits,  $L$ . Dichos paquetes se insertan en la red de acceso a una tasa de transmisión  $R$ , en bits por segundo, lo que explica un retardo de transmisión o inserción al medio,

$$ret_{tr} = \frac{L(\text{bits})}{R(\text{bits/seg})} \quad (1)$$

## Medios

- **Enlace físico:** Medio a través del cual se transmiten los datos entre el transmisor y el receptor. Puede ser guiado o no guiado.
- **Medios guiados:** Propagan las señales sobre un medio sólido. Ejemplos: cable coaxial, UTP, fibra óptica.

- **Medios no guiados:** Propagan las señales por el espacio libre, requieren antenas. Ejemplos: microondas, ondas de radio.
- **Retardo de propagación (ret prop):** Tiempo que tarda una señal en recorrer el medio de transmisión desde el transmisor hasta el receptor. Depende de la velocidad de propagación en el medio y la distancia entre ambos.

### **Red Core**

La red core consiste en una malla de routers (cada router está conectado a varios otros routers) interconectados mediante enlaces físicos. En este entorno, los datos deben recorrer una secuencia específica de routers para ir desde el origen hasta el destino, lo que define una ruta o camino a través de la red.

En el core, es fundamental distinguir entre dos métodos de transmisión de datos: la **Commutación de circuitos** y la **comutación de paquetes**.

*¿Cuál es mejor?*

La comutación de circuitos es mejor para comunicaciones continuas y en tiempo real: llamadas telefónicas, videoconferencias.

Mientras que la comutación de paquetes es más eficiente para redes de datos: navegación web, correo electrónico, transmisión de archivos.

*¿Cuál puede servir a más cantidad de usuarios?*

La comutación de paquetes porque permite compartir los recursos de la red entre varios usuarios sin reservar un camino exclusivo para cada uno. Los paquetes pueden enviarse por diferentes rutas, aprovechando al máximo el ancho de banda disponible.

*Si se usa una red con comutación de paquetes, ¿cómo se logra un comportamiento tipo circuito?*

Mediante técnicas como reserva de recursos y Calidad de Servicio (QoS).

### **Analogías para cada tipo:**

- **Commutación de circuitos:** llamada telefónica.
- **Commutación de paquetes:** correo postal.

- Antes de iniciar el intercambio de los datos, se reservan los recursos necesarios que garantizan la comunicación.
- Los circuitos pueden ser reservados en forma permanente o bajo demanda.
- Los enlaces físicos entre los routers permiten un determinado número de circuitos, fracciones de la capacidad.
- **FDM** (Frequency Division Multiplexing): división del espacio electromagnético en bandas de frecuencia o en lambdas, ofreciendo a cada circuito la máxima velocidad que permite esa fracción.
- **TDM** (Time Division Multiplexing): división del tiempo en ranuras o slots. Todo el espacio electromagnético del medio se disponibiliza por una fracción del tiempo, periódicamente para cada circuito.
- Ideal para tráfico constante, con o sin intervalos de ráfagas.

Figura 8: Comutación de circuitos.

- Store-and-forward**
- Las estaciones fraccionan los datos en paquetes,
  - que se despachan de un router a otro a plena velocidad de cada uno de los enlaces.
  - El paquete completo debe llegar al router antes de poder ser enviado por otro enlace.
  - En el router se producen las funciones de:
    - Ruteo y de determinación de rutas
    - Forwarding o despacho de paquetes
  - Si la tasa de arribo de paquetes excede la tasa de transmisión, entonces:
    - Los paquetes se encolan, a la espera de ser enviado por el siguiente enlace, y se produce un **retardo de encolamiento**. O bien,
    - Los paquetes son descartados si la memoria del router (buffer) se llena, lo que significa una **pérdida de datos**.
  - Este tipo de conmutación es muy apta para aplicaciones que generan información en forma de ráfagas alternadas con silencios. Permite una operación simple.

Figura 9: Comutación de paquetes.

## 1.2. Performance

- Retardo. Componentes en cada nodo/router:
$$Ret_{nodo} = ret_{proc} + ret_{encol} + ret_{tr} + ret_{prop}$$
- Pérdida
    - Por errores del medio
    - Como función de  $R_{entr}$ ,  $R_{sal}$ , tasa.de.arribos.al.nodo, buffer
    - Throughput: tasa a la que se envían los bits de la fuente al destino
      - Instantáneo vs. promedio
      - El enlace de menor  $R$  impone el throughput extremo-a-extremo.

Figura 10: Resumen de los factores que influyen en el retardo y la pérdida en un nodo de red.



Figura 11: Tasa de arribos vs retardo.

### Medición aproximada del retardo

Ping: herramienta del sistema operativo para obtener una aproximación del retardo entre extremos.

```
silvia@silvia-ubuntu:~/Documents$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=19.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=14.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=11.2 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=18.8 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=14.7 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=116 time=16.1 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=116 time=12.4 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=116 time=14.0 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=116 time=16.4 ms
^C
--- 8.8.8.8 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8013ms
rtt min/avg/max/mdev = 11.186/15.293/19.512/2.571 ms
```

Figura 12: Ejemplo de salida de ping.

### 1.3. Modelos de Comunicaciones

En el contexto de redes de comunicación, un modelo es una representación simplificada y estructurada de cómo funciona un proceso complejo (como la comunicación de datos en una red). Se usa para analizar, diseñar y entender las redes, y generalmente organiza los elementos en capas para hacerlo más claro.

Los modelos tradicionales son OSI y TCP/IP:

- El modelo OSI (Open Systems Interconnection) nació como una estructura teórica para entender, enseñar y diseñar sistemas de red. Divide el proceso en 7 capas, cada una con funciones específicas.
- El modelo TCP/IP (Transmission Control Protocol / Internet Protocol) surgió de la necesidad práctica de comunicar computadoras. Tiene 4 capas principales.

<b>Aplicación</b>	File Transfer, Email, Web
<b>Presentación</b>	ASCII, Text, Sonido, Video
<b>Sesión</b>	Establecimiento de conexión
<b>Transporte</b>	Comunicación end-to-end, TCP*, UDP*
<b>Red</b>	Ruteo*, Direcciónamiento*, IP*
<b>Enlace</b>	Comunicación directa, HDLC, Ethernet*
<b>Físico</b>	Medios, Codificación, Modulación

\* Lo veremos.

Figura 13: Modelo OSI: modelo de referencia conceptual en redes.

<b>Aplicación</b>	File transfer. Email, Web
<b>Transporte</b>	Comunicación end-to-end, TCP, UDP
<b>Internetwerk o Red</b>	Ruteo, Direcciónamiento, IP
<b>Físico (+Enlace)</b>	Comunicación directa, Ethernet

Figura 14: Modelo TCP/IP: usado en implementaciones reales.

## 1.4. Encapsulamiento

Es el proceso de envolver los datos con la información necesaria para que puedan viajar correctamente a través de la red.

A continuación, se muestra el encapsulado TCP/IP donde en cada capa del modelo se agrega un encabezado (y a veces una cola) al paquete de datos original.

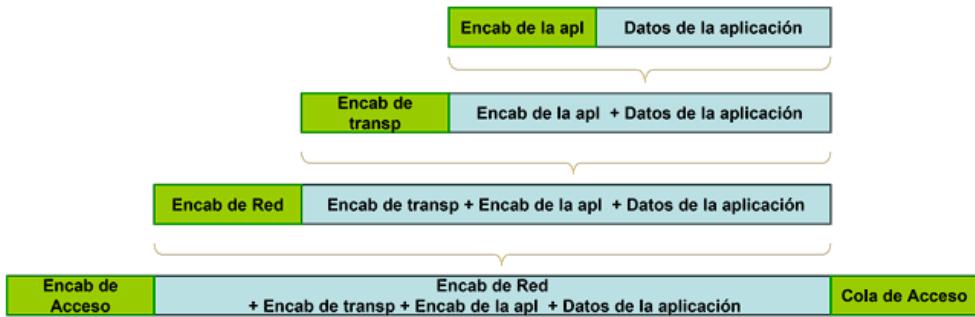


Figura 15: Encapsulamiento TCP/IP.

Idea clave: Cada capa envuelve a la anterior agregando información necesaria para que los datos puedan ser enviados, enrutados y entregados correctamente.

### Proceso del encapsulamiento

Un dispositivo genera datos de una aplicación. Dichos datos bajan capa por capa (Aplicación, transporte, Internetwerk o Red, Enlace), agregando encabezados en cada paso (lo que se ve como los bloques de colores que se van apilando).

Luego entran al medio físico (cable, WiFi, 4G, etc.) y son enviados. Los Intermediarios como routers o switches procesan solo las capas necesarias (por ejemplo, un router trabaja en la capa de red).

Finalmente llegan al dispositivo destino, donde se desempacan esas capas hasta llegar a la aplicación final.

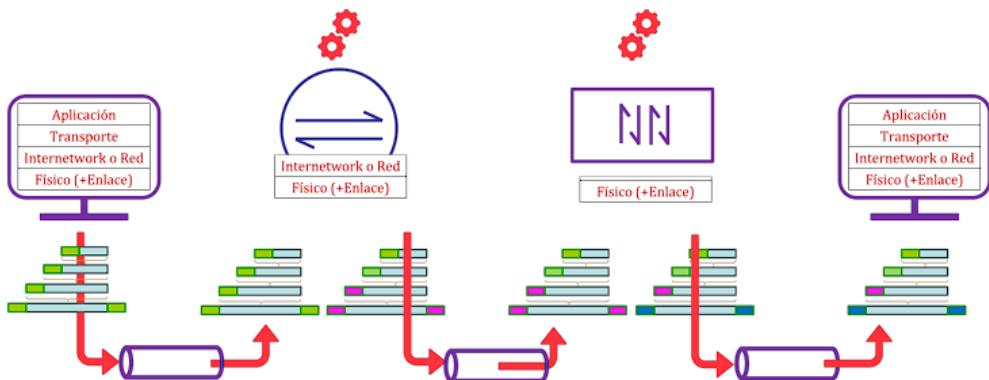


Figura 16: Representación del proceso de encapsulamiento.

## 1.5. Aplicaciones y Ejemplos

¿Cuáles son cinco tareas típicas que puede realizar una capa en un modelo de red? ¿Alguna de estas funciones podría ser implementada por más de una capa?

1. el control de errores.
2. el control de flujo.
3. la segmentación y reensamblado.
4. la multiplexación.
5. el establecimiento de conexión.

Estas tareas pueden duplicarse en diferentes capas. Por ejemplo, el control de errores suele proporcionarse en más de una capa.

*'Five generic tasks are error control, flow control, segmentation and reassembly, multiplexing, and connection setup. Yes, these tasks can be duplicated at different layers. For example, error control is often provided at more than one layer.'[1].*

¿Cuáles son las cinco capas del modelo de protocolos de Internet y cuáles son las principales funciones o responsabilidades de cada una?

1. Capa de Aplicación:  
Responsable de la interacción con el usuario.
  2. Capa de Transporte:  
Mover datagramas. La capa de transporte pasa un segmento de la capa de transporte y una dirección de destino, luego la capa de red le proporciona el segmento a la capa de transporte del destino.
  3. Capa de Red:  
Mover datagramas. La capa de transporte pasa un segmento de la capa de transporte y una dirección de destino, luego la capa de red le proporciona el segmento a la capa de transporte del destino.
  4. Capa de Enlace:  
Encamina el datagrama por una serie de routers entre el host y el destino. Son varios pasos entre cada nodo. Son tramas.
  5. Capa física:  
Mover de un nodo al siguiente los bits individuales que forman la trama.
3. ¿Qué es un mensaje de la capa de aplicación? ¿Y un segmento de la capa de transporte? ¿Y un datagrama de la capa de red? ¿Y una trama de la capa de enlace?

Un mensaje es la información que una app quiere transmitirle a su contraparte en otro host (por ej: Host 1 se conecta con Host 2 para mandarle un archivo. La app se encargaría de decirle a la capa de transporte qué archivo debe enviar. Entonces el mensaje sería el archivo).

El segmento (o datagrama UDP, según qué protocolo se esté usando) es el encabezado de la capa de transporte (que incluye checksum, longitud y etc.) + msj de la capa de app.

Un Datagrama es el encabezado de la capa de red + segmento de transporte + msj de la capa de app.

Una trama es el encabezado de la capa de enlace + datagrama de red + segmento de transporte + msj de la capa de app.

**4. ¿Qué capas de la pila de protocolos de Internet procesa un router? ¿Qué capas procesa un switch de la capa de enlace? ¿Qué capas procesa un host?**

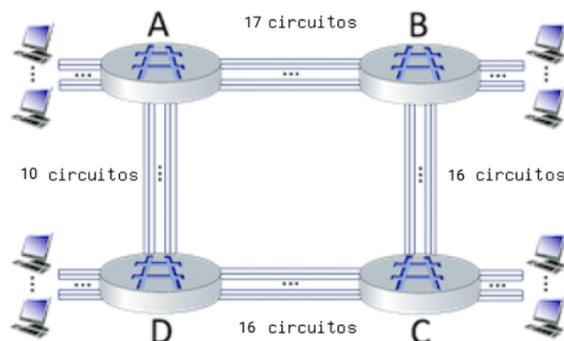
Ver figura 17.

**5. Indicar el nombre de las 15 capas por las que pasa el mensaje desde la fuente hasta el destino.**

1A, 2T, 3R, 4E, 5F, 6F, 7E, 8F, 9E, 10R, 11F, 12E, 13R, 14T, 15A.

Ver figura 18.

**6. Considere la red de conmutación de circuitos que se muestra en la siguiente figura, con los switches A, B, C y D. Suponga que hay 17 circuitos entre A y B, 16 circuitos entre B y C, 16 circuitos entre C y D y 10 circuitos entre D y A**



**¿Cuál es el número máximo de conexiones que pueden estar activas en la red al mismo tiempo?**

$$10 + 17 + 16 + 16 = 59.$$

Supongamos que todas estas conexiones máximas están en curso. ¿Qué pasa cuando llega otra solicitud de conexión de llamada a la red, será aceptada? Contesta sí o no.

No.

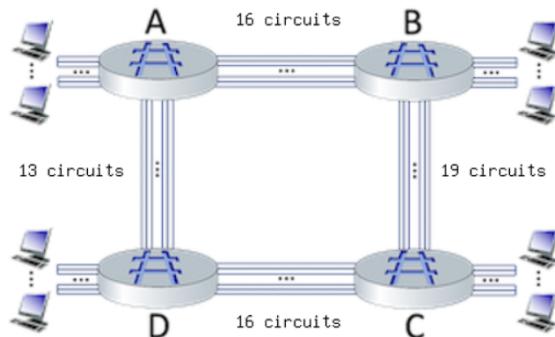
Suponga que cada conexión requiere 2 saltos consecutivos y las llamadas se conectan en el sentido de las agujas del reloj. Por ejemplo, una conexión puede ir de A a C, de B a D, de C a A y de D a B. Con estas restricciones, ¿cuál es el número máximo de conexiones que pueden estar activas en la red en cualquier momento? ¿una vez?

De A a C (y B a D): 16 conexiones, y de C a A (y D a B): 10 conexiones. El número máximo de conexiones que pueden estar activas en la red en cualquier momento: 10.

Supongamos que se necesitan 10 conexiones de A a C y 19 conexiones de B a D. ¿Podemos enrutar estas llamadas a través de los cuatro enlaces para acomodar las 29 conexiones? Contesta sí o no

No.

7. Considere la red de circuitos conmutados que se muestra en la siguiente figura, con los commutadores de circuitos A, B, C y D. Suponga que hay 16 circuitos entre A y B, 19 circuitos entre B y C, 16 circuitos entre C y D y 13 circuitos entre D y A



Supongamos que todas estas conexiones máximas están en curso. ¿Qué pasa cuando llega otra solicitud de conexión de llamada a la red, será aceptada? Contesta sí o no

NO.

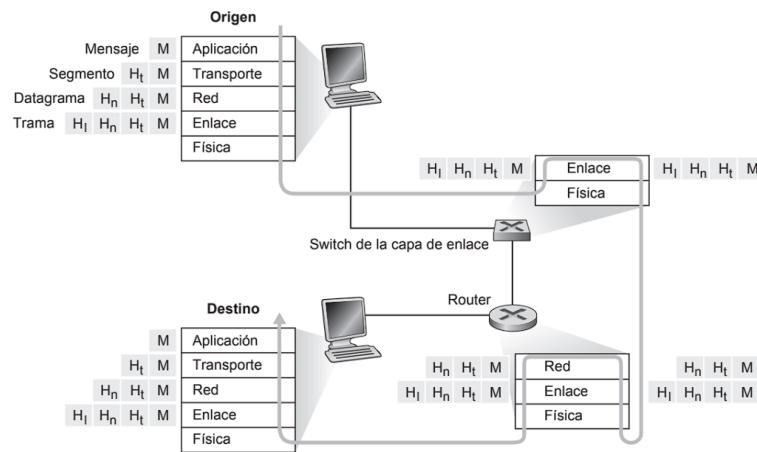
Supongamos que cada conexión requiere 2 saltos consecutivos y las llamadas se conectan en el sentido de las agujas del reloj. Por ejemplo, una conexión puede ir de A a C, de B a D, de C a A y de D a B. Con estas restricciones, ¿cuál es el número máximo de conexiones que pueden

estar activas en la red en cualquier momento? ¿una vez?

Número máximo de conexiones que pueden estar activas en la red en cualquier momento: 13.

**Supongamos que se necesitan 13 conexiones de A a C y 12 conexiones de B a D. ¿Podemos enrutar estas llamadas a través de los cuatro enlaces para acomodar las 25 conexiones? Contesta sí o no**

Sí.



**Figura 1.24** • Hosts, routers y switches de la capa de enlace. Cada uno de ellos contiene un conjunto distinto de capas, lo que refleja sus distintas funcionalidades.

Figura 17: Imagen extraída de Kurose y Ross (2017, p. 46).

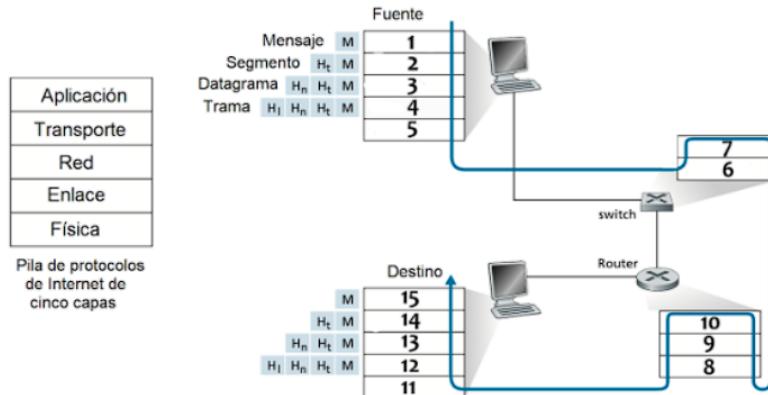


Figura 18: Capas de un mensaje de origen a destino.

## 2. Nivel de Aplicación

En el Nivel de Aplicación se encuentran los protocolos que brindan servicios directos al usuario final.

### Generalidades

Los protocolos de aplicación en redes permiten la comunicación entre dispositivos, y son utilizados por las aplicaciones para brindar servicios a los usuarios finales. Estos protocolos facilitan la interacción entre las partes involucradas, como en los modelos cliente-servidor o entre pares (peer-to-peer).

Se ejecutan en los extremos de la red, es decir, en los *hosts* o dispositivos finales, y funcionan sobre distintos sistemas operativos.

Ejemplos típicos incluyen: **HTTP** (navegación web), protocolos de correo electrónico (**SMTP**, **POP3**, **IMAP**), servicios de mensajería instantánea (chat), y plataformas de **streaming** de audio y video.

### Paradigma Cliente-Servidor

Involucra una comunicación entre un **cliente**, que solicita un servicio, y un **servidor**, que lo provee. Los roles están claramente diferenciados:

**Servidor:** está siempre disponible, a la espera de solicitudes provenientes de clientes.

**Cliente:** se conecta bajo demanda; los clientes no se comunican entre sí directamente.

Ejemplos de protocolos que utilizan este paradigma son: **HTTP**, **DNS**, **FTP**, **NTP**, **Telnet**, **SSH**, entre otros.

### Paradigma Peer-to-Peer (P2P)

A diferencia del modelo cliente-servidor, en el modelo **peer-to-peer (P2P)** no existe un servidor central siempre disponible. Los roles de cliente y servidor se diluyen o desaparecen: cada nodo puede actuar tanto como cliente como servidor.

Los dispositivos se comunican directamente entre sí, solicitando y ofreciendo servicios entre pares. Este tipo de arquitectura requiere una gestión más compleja, especialmente en términos de descubrimiento de nodos, sincronización y seguridad.

Ejemplos de aplicaciones basadas en este modelo incluyen sistemas de intercambio de archivos como **BitTorrent**, y mecanismos de descarga de software desde servidores distribuidos.

## Socket

Un socket es una interfaz de comunicación entre un proceso local (una aplicación en ejecución) y el protocolo de transporte que se encarga de enviar los mensajes por la red.

Es análogo a una puerta: permite que un proceso envíe o reciba información a través de la red. Un socket se compone de los siguientes elementos:

- Dirección IP del host.
- Protocolo de transporte (TCP o UDP).
- Número de puerto (para identificar la aplicación o servicio específico).

Existe un socket en cada extremo de la comunicación. Puede identificarse de forma similar a un número de proceso en el sistema operativo, y se representa como:

**<Protocolo de transporte, Dirección IP, Número de puerto>**

Ejemplo de una conexión:

- Lado servidor: TCP; 157.92.49.38; 80
- Lado cliente: TCP; 122.36.99.208; 9887

## Sesión

La sesión es el establecimiento de un enlace entre sockets en los extremos, ya sea en un modelo punto a punto o cliente-servidor. Define el ámbito de comunicación interactiva entre dos procesos.

Incluye el protocolo de aplicación que rige la comunicación (el “idioma” y las “reglas”) y la duración de la transacción. Dicha comunicación es bidireccional: ambos extremos pueden enviar y recibir información.

Se representa mediante una quíntupla que identifica todos los elementos clave de la comunicación en los niveles de transporte y red. Su notación típica es:

**⟨ Protocolo de transporte; IP\_origen; Puerto\_origen; IP\_destino; Puerto\_destino ⟩**

Por ejemplo: (UDP; 208.67.222.222; 23; 161.190.0.33; 23055).

## Requerimientos de las aplicaciones

Cada aplicación tiene sus requerimientos particulares, que reclamara al nivel de transporte. Estos son algunos, otros pueden ser la integridad, el jitter, etc.

Aplicación	pérdida de datos	Throughput	sensib t. de entrega
Transf. de archivo Correo electr. tráfico web	sin pérdida	dinámico	no
Audio t.real Video t.real	tolerante	audio: 5Kbps-1Mbps video:10Kbps-5Mbps	sí, < 10ms
streaming audio/video	tolerante	ídem anterior	sí, pocos seg.
juego online	tolerante	Kbps	sí, dec ms
texto, mensajería	sin pérdida	dinámico	sí/no

Figura 19: Requerimientos de las aplicaciones.

¿Qué define el nivel de aplicación?

- **Tipos de mensajes intercambiados:**  
Request, response.
- **Sintaxis de los mensajes:**  
Cuáles son los campos que componen los mensajes.
- **Semántica de los mensajes:**  
Qué significa el contenido de cada campo.
- **Reglas:**  
Cuándo y cómo los procesos envían y responden a los mensajes.
- **Protocolos abiertos:**
  - Definidos en RFCs de libre y abierta disponibilidad.
  - Permiten la interoperatividad entre sistemas.
  - Ejemplos: **SMTP, HTTP**.
- **Protocolos propietarios:**
  - Ejemplo: Skype.

## 2.1. Protocolo HTTP

HTTP (HyperText Transfer Protocol) es el protocolo de comunicación que permite la transferencia de información en la web, como páginas, imágenes, videos, etc., entre un cliente (normalmente un navegador) y un servidor web.

### WWW: World Wide Web

Sistema de información que permite el intercambio de contenidos sobre Internet. Los contenidos pueden ser documentos HTML u otros como imágenes o audio entre otros.

Los contenidos pueden ser de tipo archivo HTML, imagen JPEG, audio MP4, etc. Incluso admite MIME.

## URL: Universal Resource Locator

Tanto los servidores como los contenidos son identificados y ubicados mediante una URL, a modo de dirección. RFC 1738.



Figura 20: URL.

## HTTP: Hypertext Transfer Protocol

Protocolo de aplicación para solicitar y recibir documentos de hipertexto y otros recursos bajo demanda de los primeros.

- HTTP/1.0, 1996, RFC 1945.
- HTTP/1.1, 2014, RFC 7235.
- HTTP/2, 2015, RFC 9113.
- HTTP/3, 2022, RFC 9114.
- Cliente-Servidor.
- Utiliza TCP, puerto 80.
- Sin conservación de estado (stateless).

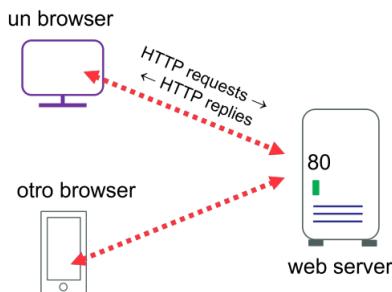


Figura 21: HTTP - Web server.

## HTTP/1.0 vs 1.1

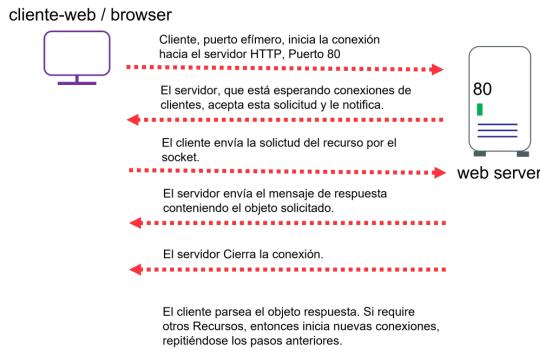


Figura 22: HTTP 1.0 - No persistente.

- El servidor deja abierta la sesión para permitir nuevas solicitudes de objetos.
- El cliente puede solicitar más objetos a medida que son referenciados.
- Cualquier extremo puede cerrar la sesión.
- Minimiza el retardo comparado con HTTP 1.0
- Incorpora mecanismos para recibir datos faltantes si la sesión se cierra anticipadamente.

Figura 23: HTTP 1.1 - Persistente.

## Mensajes

Muchos tipos de mensajes *request*: GET, POST, HEAD, ...  
 El formato estándar de los *requests* y *responses* incluye: **Start-line**, **HTTP-headers**, **empty-line** y **body**.

Algunos métodos Request:

- **GET**: Solicitud de un recurso.  
 Puede incluir parámetros específicos, por ejemplo: [www.sitios.com/lugares?canada](http://www.sitios.com/lugares?canada).
- **POST**: El cliente solicita que se acepten los datos incluidos en el cuerpo.  
 Estos datos habitualmente no son visibles para el usuario.
- **HEAD**: Similar a GET, pero sólo se requieren los encabezados.
- **PUT**: Solicitud para cargar o reemplazar un recurso en el servidor.

Respuestas HTTP:

- **1xx Informational**:  
 100 Continue, 101 Switching Protocols.
- **2xx Successful**:  
 200 OK, 202 Accepted.
- **3xx Redirection**:  
 300 Multiple Choices, 301 Moved Permanently.
- **4xx Client Error**:  
 403 Forbidden, 404 Not Found.
- **5xx Server Error**:  
 501 Not Implemented, 505 HTTP Version Not Supported.

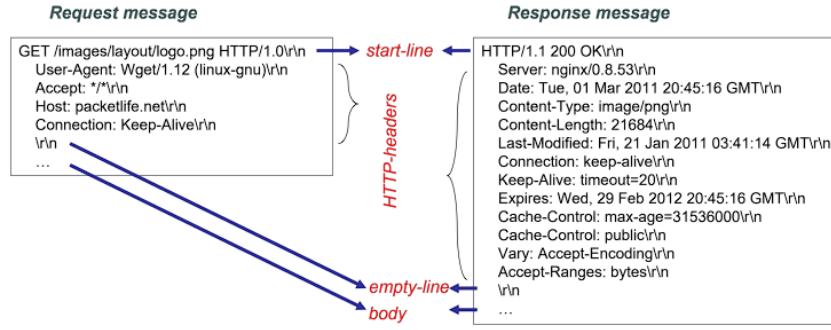


Figura 24: HTTP - mensajes.

### Conservar estado con cookies

Usos: autorizaciones, sesiones, carritos, acelerar entregas, etc.

Privacidad: conservación de información, cookies persistentes de terceros.

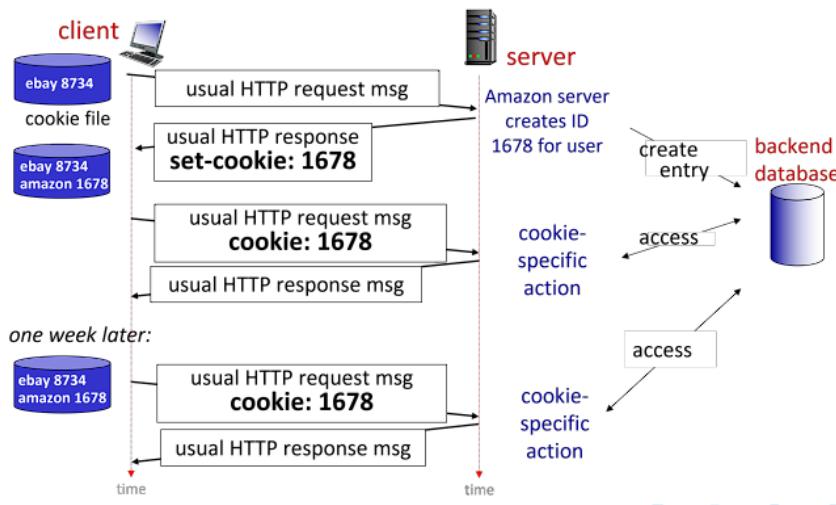


Figura 25: HTTP - cookies.

### Proxy server + web cache

Objetivo: Atender los *requests* sin involucrar al servidor.

Ventajas: Mejorar el tiempo de respuesta y reducir el consumo de ancho de banda.

Corolario: Usando GET condicional se maximiza la eficiencia de la operación.

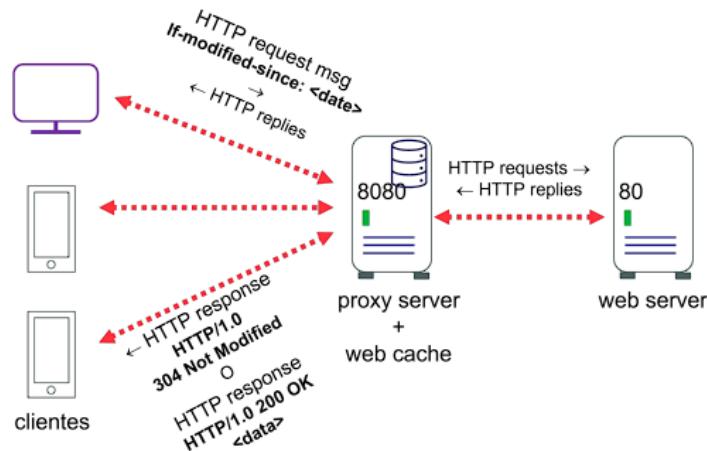


Figura 26: HTTP - Proxy & web cache.

## HTTP/2 - entregas flexibles

HTTP/2 introduce mejoras significativas respecto a HTTP/1.1, especialmente en el manejo de múltiples solicitudes y respuestas sobre una sola conexión. Algunas de sus características clave son:

- **Priorización de solicitudes:**

El cliente puede asignar prioridades a los recursos que solicita. Esto permite que los elementos más importantes (como el HTML principal) se entreguen primero, optimizando el tiempo de carga.

- **Server Push (empuje del servidor):**

El servidor puede enviar recursos no solicitados anticipadamente al cliente (por ejemplo, CSS o JS que sabe que el cliente va a necesitar). Esto reduce la latencia, ya que evita que el cliente tenga que descubrir y pedir esos recursos manualmente.

- **Transmisión basada en frames:**

Los datos de cada objeto se dividen en frames pequeños. HTTP/2 permite enviar estos frames de múltiples recursos de forma intercalada (*interleaved*), compartiendo la misma conexión. Esta técnica mitiga el problema de *Head-of-Line (HOL) blocking*, que en HTTP/1.1 afectaba negativamente al rendimiento cuando un recurso bloqueaba a los siguientes.

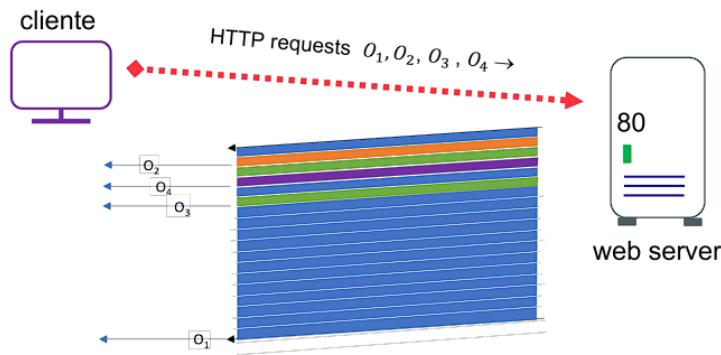


Figura 27: HTTP/2 - Web server.

## HTTP/3 - QUIC

HTTP/3 utiliza QUIC como protocolo de transporte, reemplazando a TCP por UDP para mejorar la eficiencia y reducir la latencia.

QUIC incorpora mecanismos integrados de seguridad basados en TLS 1.3, así como control de errores y gestión de congestión, permitiendo una conexión más rápida y robusta.

Además, QUIC implementa multiplexación de flujos sin sufrir bloqueo de línea de cabeza (*Head-of-Line blocking*), lo que mejora significativamente el rendimiento en redes con alta latencia o pérdida de paquetes.

HTTP/3 mejora la experiencia del usuario en conexiones móviles y redes inestables, al ofrecer una reconexión rápida y mantener las sesiones activas incluso ante cambios de red.

## REST API

Una REST API es un conjunto de reglas y restricciones que definen cómo diseñar una arquitectura de software para la comunicación entre sistemas, especialmente en aplicaciones web.

### Principios clave de REST

#### 1. Interfaz uniforme:

Permite que los diferentes sistemas interactúen de forma estandarizada, facilitando la interoperabilidad. Se basa en recursos identificados por URIs y manipulados mediante representaciones (por ejemplo, JSON o XML).

#### 2. Arquitectura Cliente-Servidor:

El cliente y el servidor están separados. El cliente solicita recursos y el servidor los proporciona. Esta separación permite desarrollar ambos lados de forma independiente.

#### 3. Stateless (Sin estado):

Cada solicitud del cliente al servidor debe contener toda la información nece-

saria para entenderla y procesarla. El servidor no guarda el estado de la sesión entre solicitudes.

4. **Cacheable (Con capacidad de caché):**

Las respuestas deben indicar si pueden ser almacenadas en caché o no, para mejorar el rendimiento y reducir la carga del servidor.

5. **Sistema en capas:**

Una arquitectura REST puede estar compuesta por capas (como balanceadores de carga, servidores de caché, etc.), sin que el cliente necesite conocer esta complejidad.

6. **Código bajo demanda (opcional):**

El servidor puede proporcionar código ejecutable al cliente (por ejemplo, scripts), aunque no es una restricción obligatoria.

## 2.2. Protocolo SMTP

SMTP es el protocolo encargado del envío de correos electrónicos desde el cliente hacia el servidor de correo, así como del intercambio de mensajes entre servidores. No se utiliza para la recepción de correos, función que corresponde a los protocolos **POP3** o **IMAP**. Este protocolo se encuentra definido en la **RFC 5321** y opera sobre el protocolo **TCP** utilizando el puerto 25.

Durante la comunicación, el servidor puede asumir tanto el rol de cliente como de servidor, dependiendo de la fase del proceso. SMTP funciona bajo una metodología de tipo *PUSH*, basada en el intercambio de comandos y respuestas, y establece una conexión persistente durante toda la sesión.

El proceso de envío de un correo electrónico consta de tres fases principales: el saludo inicial, la transferencia del mensaje y el cierre o despedida de la conexión. El mensaje transmitido tiene un formato estructurado en dos partes: el encabezado (*header*) y el cuerpo (*body*). La codificación estándar utilizada por SMTP es **ASCII de 7 bits**.

### Modelo operativo: Cliente-Servidor

Agente de Usuario (User Agent - UA):

- Es el cliente que interactúa directamente con el usuario.  
Ejemplos: Outlook, Apple Mail, Thunderbird, aplicaciones móviles como Gmail o iPhone Mail.
- Disponible bajo demanda, es decir, cuando el usuario lo abre.
- Gestiona la redacción y lectura de los mensajes.
- Envía y descarga mensajes desde el servidor cuando se requiere (a través de protocolos como **POP** o **IMAP**).

## Servidor de Correo

- Siempre activo, forma parte de la infraestructura permanente de la organización o proveedor de servicios (ejemplos: Sendmail, Postfix, servidores de Gmail).
- Recibe los mensajes de los usuarios y los encola para enviarlos a los servidores de destino.
- Envía los mensajes encolados a otros servidores.
- Recibe los mensajes destinados a los usuarios y los organiza en sus respectivas casillas.

## Separación de funciones Transfer y Delivery/Submission

Debido al alto uso de la mensajería y por cuestiones de seguridad, actualmente existen los **servidores de Transferencia** que operan entre dominios y son accesibles públicamente a través del puerto 25.

A su vez, estos servidores se conectan con los **servidores de Delivery**, que gestionan las casillas y reciben los mensajes salientes de los usuarios mediante **ESMTP** en el puerto 587.

Estos servidores de Delivery no están expuestos públicamente y se encuentran dentro de la red interna de la organización.

## Protocolos de acceso al correo

- SMTP: despacho y almacenamiento de mensajes.
- **Protocolos de acceso al correo electrónico:** utilizados para recuperar mensajes de la casilla.
- IMAP (Internet Mail Access Protocol): provee gestión de mensajes, incluyendo recuperación, eliminación, manejo de carpetas y búsqueda.
- HTTP: servicios como Gmail, Hotmail, etc., proveen una interfaz web que utiliza SMTP para enviar mensajes e IMAP (o POP3) para recuperarlos.

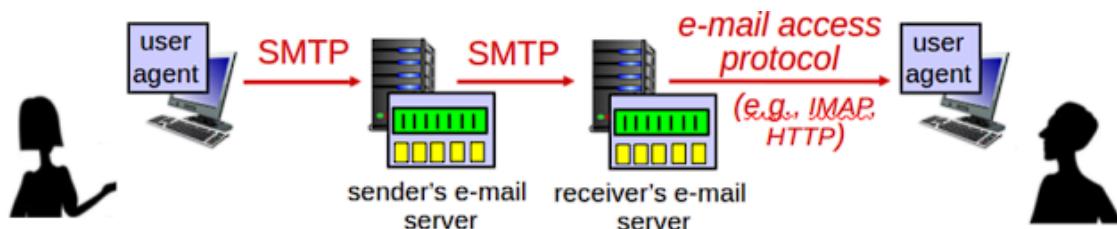


Figura 28: Protocolos de acceso al correo.

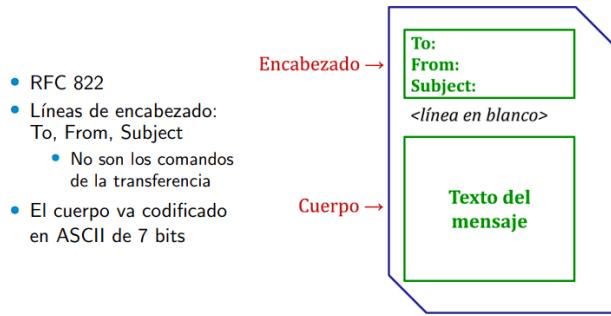


Figura 29: SMTP - formato mensaje.

#### telnet servername 25

- esperar la respuesta con código 220 del servidor
- utilizar los comandos HELO, MAIL FROM:, RCPT TO:, DATA, QUIT para enviar un mensaje sin usar el agente.

*Notar que esto funcionará si las conexiones telnet están permitidas, ya que por motivos de seguridad es habitual que esté deshabilitado*

Figura 30: Probando Telnet.

### IMAP: Internet Mail Access Protocol

RFC 9051. El servidor IMAP opera sobre TCP puerto 143, y sobre TCP puerto 993 cuando se utiliza IMAP con SSL/TLS.

Permite la administración de la casilla de correo del usuario, permitiendo que los mensajes permanezcan almacenados en el servidor.

La conexión puede mantenerse abierta mientras la interfaz de usuario esté activa.

El estado de la casilla se muestra dinámicamente, admitiendo conexiones concurrentes desde múltiples clientes.

La extensión **IMAP IDLE** habilita al servidor para notificar a los clientes la llegada de nuevos mensajes mediante *PUSH Notification*.

Cuando se utiliza **MIME**, IMAP permite realizar descargas parciales de los mensajes.

### MIME: Multipurpose Internet Mail Extensions

RFC 1521. MIME es opcional, pero requiere que ambos extremos lo implementen para funcionar correctamente.

Permite adjuntar múltiples partes en un mensaje, añadiendo líneas específicas en el encabezado que comunican la estructura del mensaje. Entre estas líneas se incluyen:

- `Mime-Version:`
- `Content-Type:`
- `Content-Transfer-Encoding:`
- `Content-ID:`
- `Content-Description:`

Ejemplo de encabezado MIME:

```
Mime-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII
```

### 2.3. Protocolo DNS

El Protocolo DNS (Domain Name System Protocol) es una aplicación auxiliar fundamental de Internet que permite traducir nombres de dominio legibles por humanos a direcciones IP numéricas.

Ejemplo: `www.fi.uba.ar` → 157.92.49.38.

Está definido en las RFCs: **1034** (Conceptos y requerimientos), **1035** (Implementación), y **2085** (Seguridad - extensiones).

Es una aplicación crítica en Internet.

Funciona bajo un modelo cliente-servidor, utilizando tanto **UDP** como **TCP** en el puerto 53.

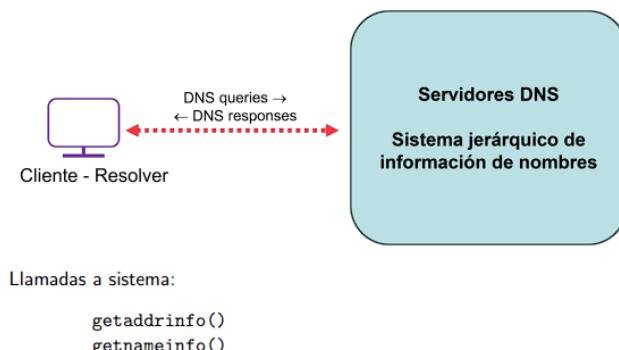


Figura 31: Modelo de consulta.

### Dominios

Un **dominio** es un grupo de hosts bajo un único control administrativo, como una compañía, agencia de gobierno, u otra organización.

Los dominios se organizan jerárquicamente: un dominio puede contener dominios subordinados.

Los nombres asignados a los dominios reflejan esta estructura jerárquica. En la parte más baja se encuentran las hojas, que corresponden a los hosts individuales.

Cada host posee una dirección IP única, y cada dominio tiene un representante o servidor autorizado que gestiona la información del dominio.

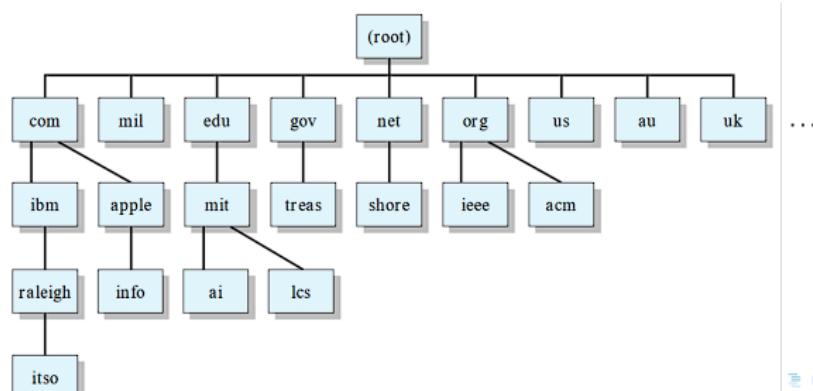


Figura 32: DNS - Dominios.

### TLD: Top Level Domain

En la raíz de la jerarquía DNS existen 13 **root name servers** (sin nombre individual), nombrados desde `a.root-servers.net` hasta `m.root-servers.net`.

Estos servidores proveen redundancia y estabilidad al sistema.

Gestionan cientos de millones de consultas diarias.

Desde la raíz se delega a 308 **ccTLDs** (Country Code Top-Level Domains), como `.arpa`, `.com`, `.net`, `.org`, `.edu`, `.ar`, `.it`, `.zw`, entre otros.

Dentro del dominio `.ar` existen 13 **SLDs** (Second Level Domains), como `.com`, `.org`, `.mil`, `.gov`, etc.

### Resource Record

Un RR es la unidad básica de información que se almacena cualquier tipo de dato DNS. Cada dominio tiene uno o varios resource records, que contienen datos específicos asociados a ese nombre de dominio.

### Uso en cmd

```
nslookup -type=TIPO dominio
```

Ejemplos:

- nslookup -type=A www.uba.ar → Obtiene la IP versión 4 (IPv4).
- nslookup -type=AAAA www.uba.ar → Obtiene la IP versión 6 (IPv6).
- nslookup -type=MX uba.ar → Obtiene los servidores de correo.
- nslookup -type=NS uba.ar → Obtiene los name servers.
- nslookup -type=CNAME www.youtube.com → Obtiene el alias del nombre.

A	Una dirección IP de host. Un RR por cada una.
AAAA	Una dirección IPv6.
CNAME	Mapea un alias con un nombre canónico o real.
MX	Mail Exchange.
NS	Servidor autoritativo para un dado dominio.
PTR	Puntero a otra parte del dominio.
SOA	Inicio de una zona autoritativa.

Figura 33: DNS - Algunos RR.

```
C:\Users\wallo>nslookup -type=CNAME www.youtube.com
Server: 196-140-30-181.fibertel.com.ar
Address: 181.30.140.196

Non-authoritative answer:
www.youtube.com canonical name = youtube-ui.l.google.com
```

Figura 34: Ejemplo CNAME en cmd.

### FQDN: Fully Qualified Domain Name

El **Nombre de Dominio Completo** (FQDN, *Fully Qualified Domain Name*) incluye el nombre de la computadora y el nombre de dominio asociado a ese equipo. Las etiquetas o *labels* están separadas por puntos “.”.

Ejemplo: `www.uba.ar`.

El FQDN siempre termina con un punto “.” que indica que el nombre está completo; sin embargo, este punto final puede omitirse en muchas aplicaciones.

Es importante distinguir que el FQDN es un dato específico, mientras que el DNS es el sistema que gestiona y resuelve estos nombres.

### Direcciones IP

IPv4 y IPv6 son los dos protocolos de direccionamiento IP usados en redes:

- **Direcciones IPv4:**  
4 bytes expresados en decimal separados por “.”.  
Ejemplo: 104.18.7.141
- **Direcciones IPv6:**  
8 bloques de 16 bits en notación hexadecimal de cuatro dígitos por bloque.  
Los bloques están separados por “:”.  
Ejemplo: 2606:4700:0000:0b22:0000:0000:6812:68d
- En las *unicast* se reconoce una parte de **red** y una parte de **host**.

### **Respuesta autoritativa y no autoritativa**

Cuando un cliente (usuario) quiere entrar a una página web, el dispositivo realiza una consulta DNS automáticamente (por ejemplo, para saber la IP de example.com). Hay dos tipos principales de respuestas:

- **Respuesta autoritativa:** proviene directamente del servidor DNS que tiene autoridad sobre el dominio consultado (por ejemplo, los servidores NS de example.com).
- **Respuesta no autoritativa:** proviene de un servidor recursivo (como los de Google o el ISP), que responde utilizando información almacenada en su caché.

Si ya se accedió recientemente al mismo dominio (por ejemplo, google.com), es posible que la IP se encuentre en la caché del sistema operativo, del navegador o del servidor DNS recursivo. En ese caso, la consulta DNS se resuelve más rápido, pero la respuesta será no autoritativa, ya que no proviene directamente del servidor con autoridad sobre el dominio.

Si la información no está en caché, o si el Time To Live (TTL) expiró, se realiza una consulta DNS completa. Esta puede dar como resultado:

- una respuesta autoritativa, si se contacta directamente con el servidor DNS autoritativo del dominio;
- o una respuesta no autoritativa, si un servidor recursivo obtiene la respuesta desde otro servidor y la guarda en caché para futuras consultas.

## Operación sin cache

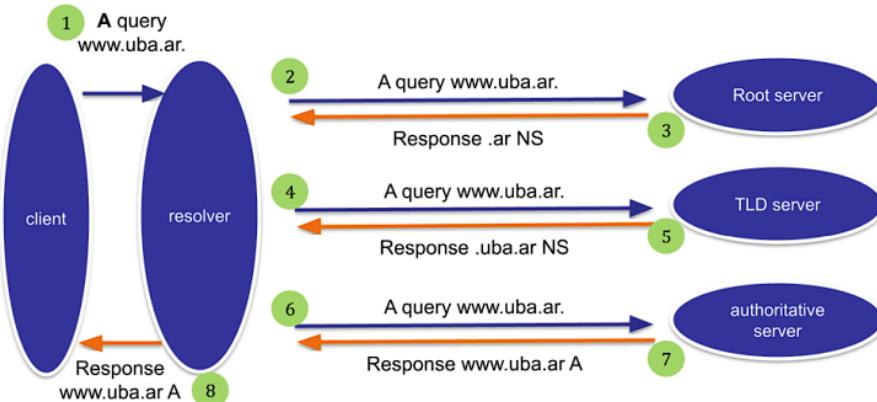


Figura 35: Operación sin cache.

1. El cliente (por ejemplo, tu navegador) quiere acceder a **www.uba.ar**. Le pide al **resolver** (servidor DNS recursivo) que resuelva ese nombre.
2. El **resolver** envía una consulta tipo A por **www.uba.ar** al **servidor raíz (root server)**.
3. El **root server** no sabe la IP, pero responde con los servidores autoritativos del TLD **.ar**. (Ejemplo: “preguntá en los servidores que manejan **.ar**”)
4. El **resolver** ahora pregunta a uno de esos servidores del TLD **.ar** por **www.uba.ar**.
5. El TLD responde con los servidores autoritativos del dominio **uba.ar**. (Ejemplo: “lo maneja tal servidor, preguntale a él”.)
6. El **resolver** pregunta a ese servidor autoritativo por **www.uba.ar**.
7. El servidor autoritativo devuelve la respuesta final: Un registro tipo A con la IP correspondiente a **www.uba.ar**.
8. El **resolver** finalmente le responde al cliente con esa IP.

Sin cache porque ninguno de los pasos intermedios están almacenados. Si lo estuvieran, algunos pasos se saltarían, haciendo el proceso más rápido.

## TCP vs UDP

UDP, puerto 53.

Encabezado corto, ágil procesamiento.

Sin garantía de entrega.

Mensajes hasta 512 bytes.

TCP, puerto 53.

Procesamiento complejo.

Con garantía.

Mensajes largos.

Zone transfers (actualización de servers de dominio primarios a secundarios).

## Método recursivo o iterativo

1. El cliente consulta internamente. Si no se resuelve internamente...
2. Consulta a un servidor DNS local. Si no se obtiene respuesta...
  - este consulta a otro servidor y luego envía la respuesta al servidor local.  
Método **recursivo**.
  - este devuelve al servidor local la dirección del siguiente servidor al cual este tendrá que consultar.  
Método **iterativo**.
3. Se repiten las consultas hasta obtenerse respuesta.
4. Se entrega la respuesta al cliente.

Localmente, el server usa el método recursivo y es **caching-only**, no se configura información sobre ninguna zona.

## Cache y TTL

- Los servidores guardan un **cache** de la información (excepto los servidores TLD).
- Esa información es **autoritativa** si proviene de una réplica o **zone transfer** de un servidor primario o **master**.
- Esta se conserva hasta un **TTL** límite.
- Esto permite reducir el número de consultas a los servidores primarios.
- En las respuestas no autoritativas se suele incluir el registro **NS** para la zona.
- **Caching** también se aplica para resoluciones no exitosas (**Negative caching**).

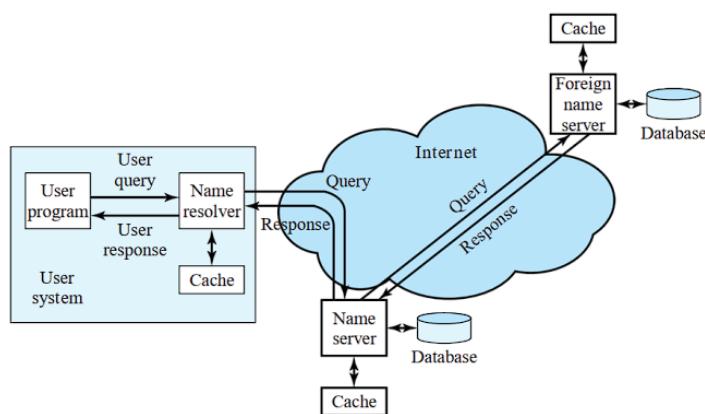


Figura 36: DNS - Operación.

## Mensaje

El DNS sirve para buscar páginas, pero internamente funciona enviando y recibiendo mensajes. Todos los mensajes DNS, ya sean consultas o respuestas, tienen una estructura común: un único formato utilizado para todas las operaciones, incluyendo **queries**, **responses**, **zone transfers**, **notifications**, y **dynamic updates**.

Cada mensaje consiste en un encabezado de 12 bytes, seguido de cuatro secciones de longitud variable: questions (o queries), answers, authority records, y otros registros.

Es importante destacar que sólo los servidores **autoritativos** pueden devolver un error de nombre, indicando que el nombre solicitado en el query no existe.

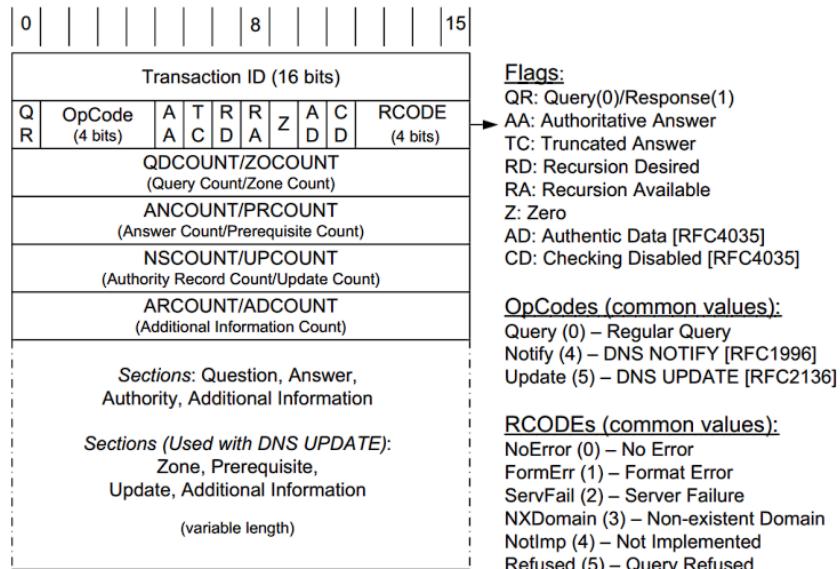


Figura 37: DNS - Formato mensaje.

## NsLookUp

Es una herramienta de línea de comandos que permite consultar información DNS sobre un dominio, como la dirección IP asociada, los servidores de correo (MX), o los servidores autoritativos (NS) que gestionan ese dominio.

**Server:** es el servidor DNS usado, el proveedor de internet.

**Address:** es la dirección IP de ese servidor DNS.

**Addresses:** es la respuesta principal.

```

[ca] Command Prompt
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\wallo>nslookup google.com
Server: 196-140-30-181.fibertel.com.ar
Address: 181.30.140.196

Non-authoritative answer:
Name: google.com
Addresses: 2800:3f0:4002:80f::200e
          142.251.129.174

```

Figura 38: Ejemplo nslookup en la consola.

The screenshot shows the NsLookup.io website interface. At the top, there is a search bar with the query "fi.uba.ar" and a "Find DNS records" button. Below the search bar, the title "DNS records for **fi.uba.ar**" is displayed. Underneath, there are tabs for "Cloudflare", "Google DNS", "Authoritative" (which is selected), "Control D", and "Local DNS". A note below the tabs states: "The authoritative DNS server responded with these DNS records. It has indicated that records should be cached for the period specified by the time to live (TTL). When the configuration of the authoritative DNS server changes, it will immediately serve the updated DNS records." The "A records" section shows one entry: "IPv4 address" followed by "186.33.219.219" and "Revalidate in 1h". The "AAAA records" section says "No AAAA records found.". The "CNAME record" section is partially visible at the bottom.

Figura 39: NsLookup.io: página web para encontrar registros DNS.

## LookUp Inverso

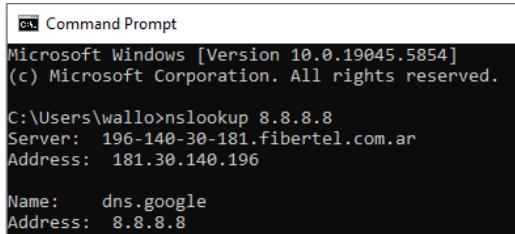
El LookUp inverso (o *reverse DNS lookup*) es justo lo contrario de lo que se hace normalmente con el DNS, es decir, a partir de una dirección IP se obtiene a qué nombre de dominio está asociado.

### *¿Cómo funciona?*

Utiliza un registro especial de tipo PTR (*Pointer Record*) en el DNS. La búsqueda se realiza en un dominio especial llamado `in-addr.arpa` (para IPv4) o `ip6.arpa` (para IPv6).

### *¿Para qué se usa?*

- Verificación de servidores de correo (para evitar *spam*).
- Herramientas de red como `nslookup` o `dig`.
- Auditorías, seguridad, y trazas de red.



```
Command Prompt
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\wallo>nslookup 8.8.8.8
Server: 196-140-30-181.fibertel.com.ar
Address: 181.30.140.196

Name: dns.google
Address: 8.8.8.8
```

Figura 40: Ejemplo nslookup inverso en la consola.

## Seguridad del servicio DNS

Medidas y mecanismos existentes para proteger el sistema DNS de ataques o manipulaciones que puedan hacer que un usuario termine en un sitio falso, o que la información sea alterada:

- **Ataques DoS: queries excesivas:**

Un atacante envía muchas consultas DNS en poco tiempo a un servidor DNS para saturarlo y que deje de responder (Denegación de Servicio). Es como “bombardear” al servidor con preguntas.

- **Amplificación:**

Este ataque se aprovecha de que las respuestas DNS suelen ser mucho más grandes que las consultas. Por ejemplo, una consulta puede ser de 50 bytes, pero la respuesta puede ser de 1000 bytes, es decir, 20 veces más grande (relación 20:1).

Entonces, un atacante envía consultas pequeñas pero con la dirección IP de la víctima (suplantada) como origen. El servidor DNS responde con una respuesta grande que envía a la víctima, saturándola con tráfico.

- **Impersonificación de servidores y alteración de los registros:**

Un atacante finge ser un servidor DNS legítimo. Por ejemplo, vos hacés una consulta DNS y, en lugar de responderte el servidor verdadero, te responde uno falso controlado por el atacante. Resultado: el atacante puede darte una IP equivocada, como la de un sitio falso que él controla (phishing, robo de contraseñas, etc.).

Alteración de registros DNS: Incluso si el servidor parece legítimo, los datos (los registros) que devuelve pueden haber sido modificados. Por ejemplo, que el registro A para www.banco.com apunte a la IP de un servidor del atacante.

- **Cache poisoning:**

Sin seguridad, alguien podría hacer “cache poisoning” y engañar a tu navegador para que vaya a una página falsa, aunque escribas bien la dirección.

## DNSSEC: Domain Name System Security Extensions:

- Añade firmas criptográficas a los registros.
- Suma nuevos registros para la validación de las firmas.
- RFC 2535, 4033, 4034.

## 2.4. Aplicaciones y Ejemplos

### 1. Para una sesión de comunicación entre un par de procesos, ¿qué proceso es el cliente y cuál es el servidor?

En una arquitectura cliente-servidor existe un host siempre activo, denominado servidor, que da servicio a las solicitudes de muchos otros hosts, que son los clientes.

El **browser** (navegador) es el cliente. Y el servidor web puede ser **apache** (Apache HTTP Server, servidor web). Los clientes son los inicializadores de la comunicación y el servidor está en la espera 'escuchando'.

*"En la Web, un navegador es un proceso cliente y el servidor web es un proceso servidor. En la compartición de archivos P2P, el par que descarga el archivo se designa como el cliente y el host que está cargando el archivo se designa como el servidor" (Kurose Ross, 2017, p. 74).*

### 2. Luego de leer “2.1.1 Arquitecturas de las aplicaciones de red” en Computer Networking (Kurose), responder si para una aplicación de intercambio de archivos P2P\*, ¿está de acuerdo con la afirmación: “No existe la noción de los lados cliente y servidor de una sesión de comunicación”? ¿Por qué o por qué no?

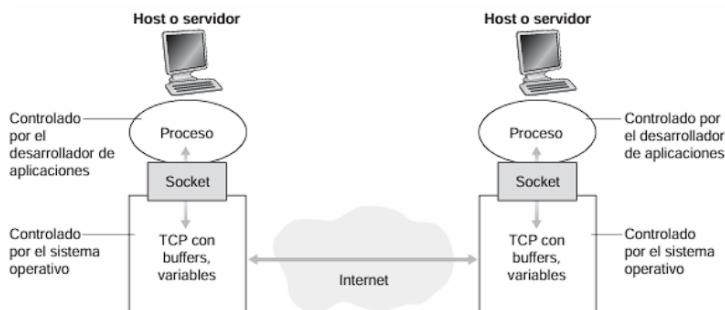
Sí existe la noción de cliente (el que descarga) y servidor (el que carga el archivo).

*"En la compartición de archivos P2P, el par que descarga el archivo se designa como el cliente y el host que está cargando el archivo se designa como el servidor." (Kurose Ross, 2017, p. 73).*

### 3. ¿Qué es un “socket”?

Es una interfaz software que permite la comunicación entre procesos a través de una red. Representa el punto de conexión entre la capa de aplicación y la capa de transporte del modelo de red. A través del socket, una aplicación puede enviar y recibir datos utilizando los servicios del protocolo de transporte (por ejemplo, TCP o UDP).

Análogamente, un socket puede verse como un enchufe: el enchufe (socket) conecta la aplicación (como si fuera un electrodoméstico) a la red eléctrica (la red de comunicaciones).



**Figura 2.3** • Procesos de aplicación, sockets y protocolo de transporte subyacente.

Figura 41: Imagen tomada de Kurose y Ross (2017, p. 74).

#### 4. Mencione una aplicación que requiera que no haya pérdida de datos y que también sea extremadamente sensible al tiempo.

Una app que no sea tolerante a pérdidas y que tenga alta sensibilidad a tiempos de entrega puede ser **telecirugía**. El cirujano recibe imágenes y respuestas hápticas de lo que está haciendo a distancia. No se pueden perder muchos datos porque debe comprenderse exactamente lo que se ve del otro lado y lo que se está haciendo. Y obviamente no pueden tardar mucho los paquetes porque se perdería precisión.

#### 5. ¿Cuáles son algunas diferencias entre TCP y UDP?

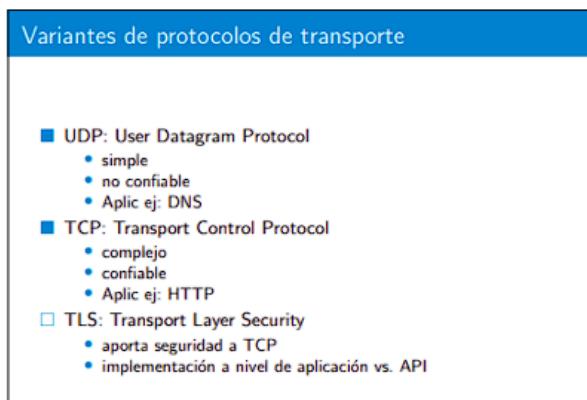


Figura 42: UDP vs TCP.

#### 6. ¿Por qué TCP y UDP no tienen mecanismos de cifrado?

Por diseño: ninguno de los dos especifica mecanismos de cifrado propios en los protocolos, y delegan (implícitamente) estas funciones a la capa de aplicación.

Los mecanismos de cifrado llegaron luego de la creación de los protocolos. Además los mecanismos de cifrado se actualizan continuamente y van cambiando. Y, finalmente, para la época en la que se inventaron los protocolos de transporte los procesadores eran muy lentos como para resolver todo en tiempo razonable.

**7. ¿Por qué HTTP, SMTP e IMAP se ejecutan sobre TCP en lugar de UDP?**

Seguridad de envío y no importa el tiempo.

**8. ¿Cuál es la diferencia entre una conexión HTTP persistente y una conexión no persistente?**

*NO persistente (o no keep-alive):* HTTP/1.0 lo usa por defecto.

Cuando se recibe el recurso, la conexión se cierra (lo hace el servidor). Esto implica más latencia y sobrecarga en el servidor y la red.

*Persistente (o keep-alive):* HTTP/1.1 lo usa por defecto.

Se mantiene abierta una sola conexión TCP para permitir nuevas solicitudes de objetos. Cualquier extremo puede cerrar la sesión. Además es más eficiente que el no persistente debido a que minimiza el retardo. Finalmente incorpora mecanismos para recibir datos faltantes si la sesión se cierra anticipadamente.

**9. Describa cómo el almacenamiento en caché web puede reducir el retraso en la recepción de un objeto solicitado. ¿El almacenamiento en caché web reducirá la demora para todos los objetos solicitados por un usuario o solo para algunos de los objetos? ¿Por qué? ¿En qué casos un almacenamiento en caché web no mejora el tiempo de respuesta?**

El proxy gestiona y redirige el tráfico de red, actuando como intermediario.

Una caché web, también denominada servidor proxy, es una entidad de red que satisface solicitudes HTTP en nombre de un servidor web de origen.

Cuando la caché web recibe el objeto, almacena una copia en su dispositivo de almacenamiento local y envía una copia, dentro de un mensaje de respuesta HTTP, al navegador del cliente (a través de la conexión TCP existente entre el navegador del cliente y la caché web).

¿Reducción de retraso? ¿para todos los objetos? → Al devolver recursos desde caché local/intermedia, no se espera al servidor original, por lo que reduce el retardo en la recepción de un objeto solicitado, pero solo para los que están cacheados y válidos.

¿Cuándo no mejora el tiempo de rta? → Primera visita, recursos no cacheables, contenido dinámico o cache expirado.

¿Da respuesta no autoritativa? → Sí, cuando la caché entrega el recurso sin consultar el servidor original.

**10. La siguiente cadena de caracteres ASCII ha sido capturada por Wireshark cuando el navegador enviaba un mensaje GET HTTP (es decir, este es el contenido real de un mensaje GET HTTP). Los caracteres**

< cr >< lf > representan el retorno de carro y el salto de línea (es decir, la cadena de caracteres en cursiva < cr > del texto que sigue a este párrafo representa el carácter de retorno de carro contenido en dicho punto de la cabecera HTTP). Responda a las siguientes cuestiones, indicando en qué parte del siguiente mensaje GET HTTP se encuentra la respuesta.

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai  
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax) <cr><lf>Accept: text/xml, application/xml, application/xhtml+xml, text/html;q=0.9, text/plain;q=0.8, image/png,*/*;q=0.5<cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept-Encoding: zip,deflate<cr><lf>Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr><lf>Connection:keep-alive<cr><lf><cr><lf>
```

Con < cr > (retorno de carro) y < lf > (salto de línea) la cadena es:

```
GET /cs453/index.html HTTP/1.1  
Host: gaia.cs.umass.edu  
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax)  
Accept: text/xml, application/xml, application/xhtml+xml, text/html;q=0.9, text/plain;q=0.8, image/png, */*;q=0.5  
Accept-Language: en-us,en;q=0.5  
Accept-Encoding: gzip, deflate  
Accept-Charset: ISO-8859-1, utf-8;q=0.7, *;q=0.7  
Keep-Alive: 300  
Connection: keep-alive
```

a. ¿Cuál es la URL del documento solicitado por el navegador?

gaia.cs.umass.edu/cs453/index.html

”... el nombre de host del servidor que alberga al objeto y el nombre de la ruta al objeto. Por ejemplo, en el URL: http://www.unaEscuela.edu/unDepartamento/imagen.gif, www.unaEscuela.edu corresponde a un nombre de host y /unDepartamento/imagen.gif es el nombre de una ruta”(Kurose Ross, 2017, p. 82).

b. ¿Qué versión de HTTP se está ejecutando en el navegador?

HTTP/1.1 → 1.1

c. ¿Qué tipo de conexión solicita el navegador, persistente o no persistente?

Connection:keep-alive → Persistente

d. ¿Cuál es la dirección IP del host en el que se está ejecutando el navegador?

No se puede saber esa IP solo a partir del mensaje GET porque este mensaje fue capturado del lado del cliente, y HTTP no incluye la IP del remitente en el cuerpo del mensaje.

e. ¿Qué tipo de navegador inicia este mensaje? ¿Por qué es necesario indicar el tipo de navegador en un mensaje de solicitud HTTP?

*User-Agent: Mozilla/5.0* → Mozilla. Para dar más precisión a la solicitud. Mejora la compatibilidad, para que no te manden cosas que no se adapten a tu navegador.

11. El siguiente texto muestra la respuesta devuelta por el servidor al mensaje de solicitud GET HTTP del problema anterior. Responda a las siguientes cuestiones, indicando en qué parte del siguiente mensaje se encuentran las respuestas.

```
HTTP/1.1 200 OK<lf>Date: Tue, 07 Mar 2008  
12:39:45GMT<cr><lf>Server: Apache/2.0.52 (Fedora)  
<cr><lf>Last-Modified: Sat, 10 Dec 2005 18:27:46 GMT<cr><lf>ETag:  
"526c3-f22-a88a4c80"<cr><lf>Accept-  
Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf>  
Keep-Alive: timeout=max=100<cr><lf>Connection:  
Keep-Alive<cr><lf>Content-Type: text/html; charset=  
ISO-8859-1<cr><lf><cr><lf><!doctype html public "-//w3c//dtd html 4.0 transitional//en"><lf><html><lf> <head><lf>  
<meta http-equiv="Content-Type"  
content="text/html; charset=iso-8859-1"><lf> <meta  
name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT  
5.0; U) Netscape]"><lf> <title>CMPSCI 453 / 591 /  
NTU-ST550ASpring 2005 homepage</title><lf></head><lf>  
<aquí continúa el texto del documento (no mostrado)>
```

Con *<cr>* (retorno de carro) y *<lf>* (salto de línea) la cadena es:

```
HTTP/1.1 200 OK  
Date: Tue, 07 Mar 2008 12:39:45 GMT  
Server: Apache/2.0.52 (Fedora)  
Last-Modified: Sat, 10 Dec 2005 18:27:46 GMT  
ETag: "526c3-f22-a88a4c80"  
Accept-Ranges: bytes  
Content-Length: 3874  
Keep-Alive: timeout=max=100  
Connection: Keep-Alive  
Content-Type: text/html; charset=ISO-8859-1  
  
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

```
<meta name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT 5.0; U)
Netscape]"><title>CMPSCI 453 / 591 / NTU-ST550A Spring 2005 homepage</title>
</head>
```

- a. ¿Ha podido el servidor encontrar el documento? ¿En qué momento se suministró la respuesta con el documento?

*200 OK* → Sí, ha podido.

Fecha: *Tue, 07 Mar 2008 12:39:45 GMT*

- b. ¿Cuándo fue modificado por última vez el documento?

*Last-Modified: Sat, 10 Dec 2005 18:27:46 GMT* → 10/12/2005 18:27:46 GMT

- c. ¿Cuántos bytes contiene el documento devuelto?

*Content-Length: 3874* → 3874 (Content-Length mide payload, sin header.)

- d. ¿Cuáles son los primeros cinco bytes del documento que se está devolviendo?

¡!doc

- e. ¿Ha acordado el servidor emplear una conexión persistente?

*Connection: Keep-Alive* → Persistente.

- 12.** Un cliente HTTP desea recuperar un documento web que se encuentra en un URL dado. Inicialmente, la dirección IP del servidor HTTP es desconocida. ¿Qué protocolos de la capa de aplicación y de la capa de transporte además de HTTP son necesarios en este escenario?

Capa de aplicación:

- DNS : para resolver el nombre de dominio del URL a una dirección IP.

Capa de Trasporte:

- UDP: para enviar la consulta DNS al servidor DNS.

- TCP: para establecer una conexión confiable con el servidor web una vez obtenida la IP (recordemos que HTTP (versión 1.0 o 1.1) se basa en TCP para transmitir datos).

- 13.** TEXT es uno de los Content-Type que admite MIME para la transferencia de contenidos vía mail o web. ¿Qué otros tipos hay? ¿Qué subtipos?

MIME (Multipurpose Internet Mail Extensions):

Es un sistema (**formato**) que le dice al receptor qué tipo de archivo está recibiendo y

cómo manejarlo, es decir describe el contenido del mensaje (formato, tipo, adjuntos).

Antes de mime, los correos electrónicos usaban el protocolo SMTP, que estaba diseñado en los años 70s/80s y solo permitía enviar texto en formato ASCII, es decir no se podían mandar ni siquiera caracteres especiales como emojis, tildes, PDFs o audios. Era muy limitado. Y, las imágenes eran representadas en puro texto.

## Working of MIME Protocol

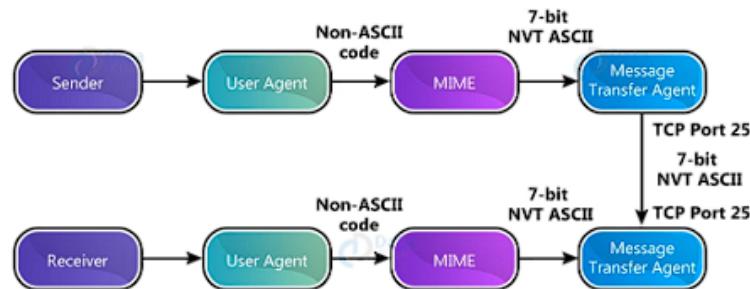


Figura 43: Secuencia MIME.

Tipo MIME	Subtipo	Descripción
text	plain html css csv	Texto plano HTML Hojas de estilo Datos separados por comas
image	jpeg png gif svg+xml	Imagen JPEG Imagen PNG Imagen GIF Imagen vectorial SVG
audio	mpeg ogg wav	Audio MP3 Audio OGG Audio WAV
video	mp4 ogg webm	Video MP4 Video OGG Video WebM
application	json pdf zip octet-stream	Datos en JSON Documento PDF Archivo ZIP Binario genérico
multipart	form-data mixed alternative	Formulario con archivos Contenido combinado Varias versiones del contenido

Cuadro 1: Ejemplos de tipos y subtipos MIME.

### 14. ¿Cuáles códigos de respuesta existen y qué significan?

Respuestas:

- 1xx Informational.  
100 Continue - 101 Switching Protocols.
- 2xx Successful.  
200 OK - 202 Accepted.
- 3xx Redirection.  
300 Multiple Choices - 301 Moved Permanently.
- 4xx Bad request.  
403 Forbidden - 404 Not Found.
- 5xx Server Error.  
501 Not Implemented - 505 HTTP Version Not Supported.

**15. A partir de alguno de los mensajes que haya recibido, explore su encabezado y contenidos.**

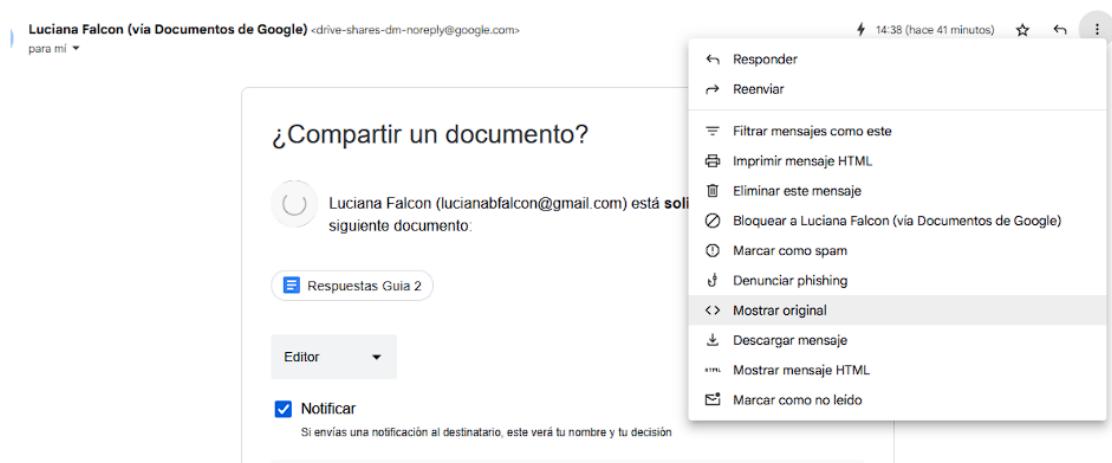


Figura 44: Como explorar un mensaje.

```
MIME-Version: 1.0
From: Luciana Falcon <lucianabfalcon@gmail.com>
Date: Wed, 18 Jun 2025 22:15:28 -0300
X-Gm-Features: AX0GCFshDjW5LZVkfMwiyKH1HpVpZuMFd3FWCTIaG4on2gQen7p7yHw61hvPIc0
Message-ID: <CA+EztL16wz5RDwf0eQb+te+epkO=XjEqOooYQ00u6h8Y4PSFJQ@mail.gmail.com>
Subject: mail prueba
To: lfalcon@fi.uba.ar
Content-Type: multipart/alternative; boundary="000000000000da5cfa0637e27bb2"

--000000000000da5cfa0637e27bb2 Espacio y Separador
Content-Type: text/plain; charset="UTF-8" Cuerpo
mensaje de prueba
--000000000000da5cfa0637e27bb2 Espacio y Separador Cuerpo HTML
Content-Type: text/html; charset="UTF-8"
<div dir="ltr"><div class="gmail_default" style="font-family:georgia,serif">mensaje de prueba</div></div>
--000000000000da5cfa0637e27bb2-- Espacio y Separador
```

Figura 45: Mensaje original: contenido.

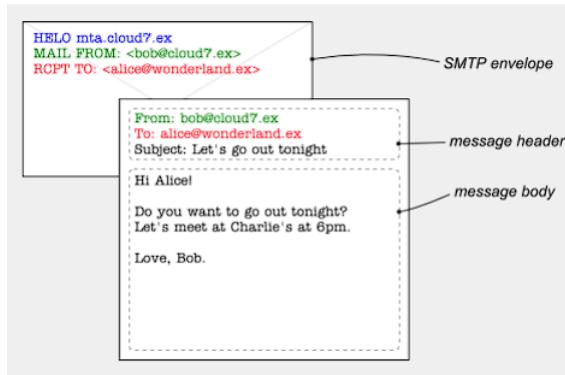


Figura 46: Contenido de un mail.

### 16. Mencione 3 motivos por los cuales un servidor de DNS no puede ser centralizado.

3 motivos por los cuales un servidor de DNS no puede ser centralizado Veamos qué dice Kurose en su página 136: No es apropiada para el Internet actual, con su gran (y creciente) número de hosts. Los problemas con un diseño centralizado incluyen:

- Un único punto de falla.  
Si el servidor DNS falla, ¡todo Internet también falla!.
- Volumen de tráfico.  
Un solo servidor DNS tendría que manejar todas las consultas DNS (para todas las solicitudes HTTP y mensajes de correo electrónico generados por cientos de millones de hosts).
- Base de datos centralizada y distante.  
Un único servidor DNS no puede estar “cerca” de todos los clientes que consultan. Si colocamos el servidor DNS único en Nueva York, entonces todas las consultas desde Australia deben viajar al otro lado del mundo, quizás a través de enlaces lentos y congestionados. Esto puede provocar retrasos significativos.
- Mantenimiento.  
El único servidor DNS tendría que mantener registros de todos los hosts de Internet. Esta base de datos centralizada no solo sería enorme, sino que también tendría que actualizarse frecuentemente para tener en cuenta cada nuevo host.

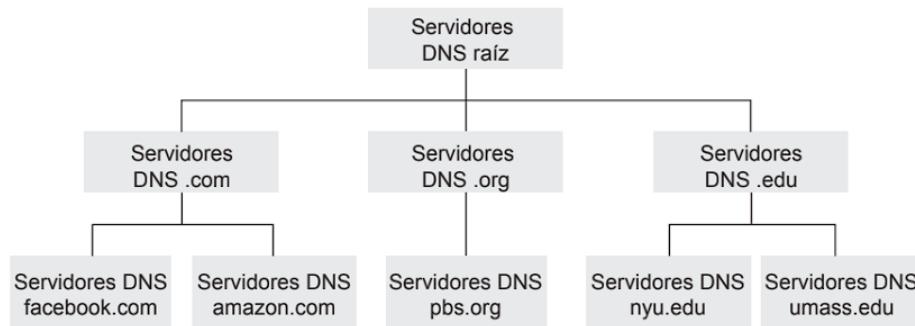
Ampliación con base en la RFC 1034:

1. Escalabilidad - Antes del DNS, existía un archivo llamado HOSTS.TXT, mantenido por el NIC. Se actualizaba manualmente y se distribuía vía FTP a todos los hosts. Problema: A medida que crecía el número de nodos, el ancho de banda necesario crecía cuadráticamente. Imposible de sostener. RFC 1034 (1987): “The sheer size of the database and frequency of updates suggest that it must be maintained in a distributed manner”.

2. Tolerancia a fallos y descentralización natural - Internet, desde sus inicios, fue pensada como una red resiliente, sin puntos únicos de falla. Centralizar los nombres en un único servidor (o archivo) haría vulnerable a toda la red. El diseño distribuido y jerárquico del DNS permite que, si un nodo falla, otros puedan responder o asumir su rol temporalmente.
3. Consistencia eventual y caching - El sistema DNS sacrifica consistencia fuerte en favor de escalabilidad y rendimiento. Las respuestas se cachearán localmente (resolver, browser, sistema operativo) y se actualizan periódicamente con un TTL (Time To Live). Esto reduce la carga de consultas y permite performance aceptable globalmente.
4. Desacoplamiento de autoridad - .com, .org, .edu, etc., cada uno gestionado por organizaciones distintas. Los dominios como google.com o mit.edu administran sus propios nombres internos. Esto es clave para mantener la flexibilidad y autonomía de cada actor en internet.

En conclusión: La centralización no solo era técnicamente inviable por cuestiones de escalabilidad y rendimiento, sino que ideológicamente iba contra los principios de la arquitectura de internet. DNS es una solución distribuida, jerárquica y tolerante a fallos, diseñada exactamente para resolver ese problema.

**17. Dada la jerarquía de servidores DNS (Servidores DNS raíz, Servidores de dominio de nivel superior y servidores autoritativos) se pide:**



**Figura 2.17** ♦ Parte de la jerarquía de los servidores DNS.

Figura 47: Imagen tomada de Kurose y Ross (2017, p. 108).



Figura 48: Consulta DNS, por la IP de Amazon.com, ejemplo.

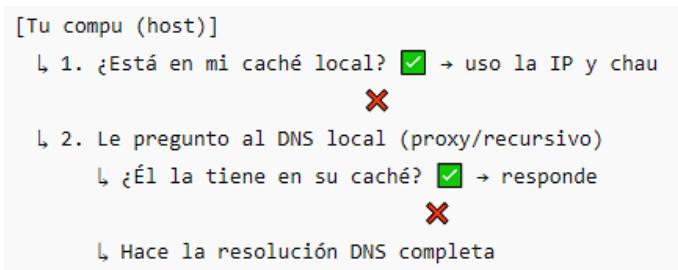
a. ¿Qué direcciones IP proporcionan cada uno?

1. Servidor Raíz → Devuelve las IP de los TLD.  
Los servidores de nombres raíz proporcionan las direcciones IP de los servidores TLD.
  2. Servidores TLD\* → Devuelve la IP de los servidores autoritativos.
  3. Servidor Autoritativo → Devuelve la IP del servidor con el que se quiere comunicar.

\*TLD significa Top-Level Domain, ejemplo: **com** es el TLD, **amazon** es el Second-Level Domain (SLD) y **www** es un subdominio (opcional).

b. Cuando un host realiza una consulta DNS, ¿a qué tipo de servidor de DNS, que actúa como proxy llega?

Al DNS local.



c. Llamamos R al Servidor DNS raíz, T al Servidor TLD, A al servidor autoritativo, L al servidor local y H al host que realiza la consulta. Supongamos que el servidor TLD conoce el servidor DNS autoritativo correspondiente al nombre de host. Indique el trayecto del mensaje de

consulta desde que el host lo inicia hasta que obtiene la dirección IP del host consultado mediante la letra correspondiente, un guión, la letra siguiente y así sucesivamente.

R al Servidor DNS raíz

T al Servidor TLD

A al servidor autoritativo

L al servidor local

H al host que realiza la consulta

Trayecto iterativo:

$H \rightarrow L \rightarrow R \rightarrow L \rightarrow T \rightarrow L \rightarrow A \rightarrow L \rightarrow H$ .

Trayecto recursivo:

$H \rightarrow L \rightarrow R \rightarrow T \rightarrow A \rightarrow T \rightarrow R \rightarrow L \rightarrow H$ .

d. ¿Cuántos mensajes DNS se necesitan enviar para obtener la dirección correspondiente a un nombre de host si no se encuentra en el proxy del servidor local y servidor TLD conoce el servidor DNS autoritativo correspondiente al nombre del host consultado?

8 consultas.

18. ¿Cuál es la diferencia entre consultas de DNS iterativas y consultas recursivas?

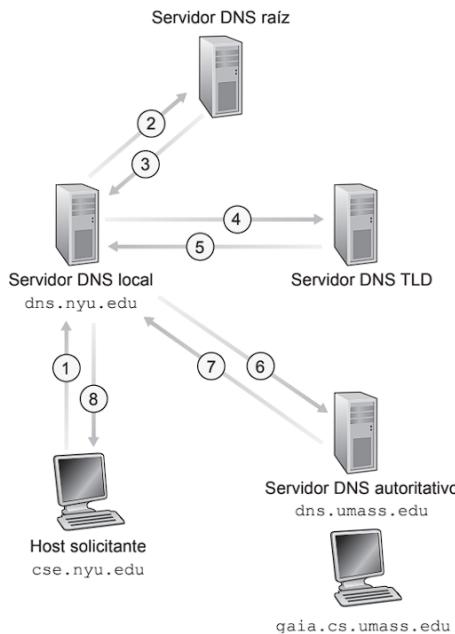


Figura 49: (a) Consulta Iterativa

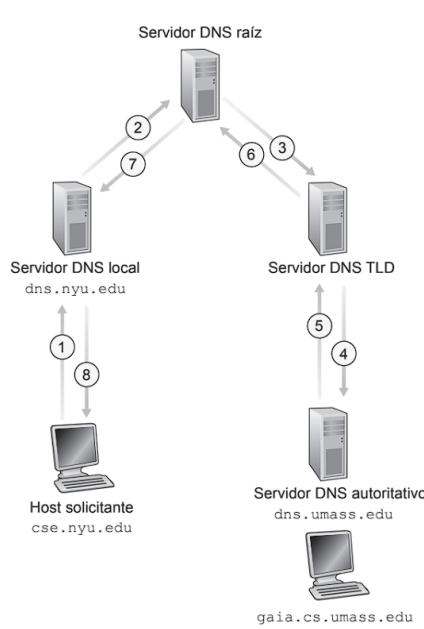


Figura 50: (b) Consulta Recursiva

Figura 51: Imágenes tomadas de Kurose y Ross (2017, p. 110-111).

**19. Dado que la correspondencia entre el nombre de un host y su dirección IP puede cambiar, ¿cuál es el comportamiento de un servidor de DNS respecto de la información almacenada en su caché DNS para prevenir esto?**

La información en la caché se borra en el tiempo que indique el parámetro **TTL** (Time to live).

El servidor DNS almacena las respuestas en su caché sólo durante el tiempo que indica el TTL. Cuando el TTL vence, la información se considera no confiable (por ejemplo, porque un servidor podría haber cambiado de dirección o un sitio web podría haberse movido a otro hosting), por lo que se vuelve a consultar al servidor original para prevenir inconsistencias.

**20. Un cliente se conecta a una aplicación de home banking basada en la web mediante protocolo HTTP, el cual no tiene memoria del estado de la conexión. Una vez que inició sesión, ¿cómo identifica el servidor al cliente?**

El servidor utiliza cookies para identificar al cliente después del inicio de sesión. Por ejemplo:

`Set-cookie: 1678`

El servidor responde con una cabecera HTTP **Set-Cookie**, que le indica al navegador guardar una cookie:

Set-Cookie: session\_id=abc123xyz; HttpOnly; Secure; SameSite=Strict

En cada nueva petición que haga el cliente, el navegador automáticamente envía esa cookie:

Cookie: session\_id=abc123xyz

**21. Supongamos que estás realizando un análisis de tráfico de red y te encontrás con la siguiente captura de dos paquetes DNS relacionados con la resolución de nombres de dominio para "wireshark.org".**

No.	Time	Source	Destination	Protocol	Length	Info
1 0.000000		192.168.0.114	205.152.37.23	DNS	73	Standard query 0x180f A wireshark.org
2 0.091164		205.152.37.23	192.168.0.114	DNS	89	Standard query response 0x180f A wireshark.org A 128.121.50.122

```

> Frame 1: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface unknown, id 0
> Ethernet II, Src: HonHaiPrecis_6e:8b:24 (00:16:ce:6e:8b:24), Dst: DLink_21:99:4c (00:05:5d:21:99:4c)
> Internet Protocol Version 4, Src: 192.168.0.114, Dst: 205.152.37.23
> User Datagram Protocol, Src Port: 1060, Dst Port: 53
> Domain Name System (query)

> Frame 2: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface unknown, id 0
> Ethernet II, Src: DLink_21:99:4c (00:05:5d:21:99:4c), Dst: HonHaiPrecis_6e:8b:24 (00:16:ce:6e:8b:24)
> Internet Protocol Version 4, Src: 205.152.37.23, Dst: 192.168.0.114
> User Datagram Protocol, Src Port: 53, Dst Port: 1060
> Domain Name System (response)

```

a. ¿Qué tipo de consulta DNS se envía en el primer paquete y quién la realiza?

La máquina con IP 192.168.0.114 (tu cliente, probablemente una PC) está haciendo una consulta estándar (Standard query) de tipo A (quiere saber la dirección IPv4) asociada al dominio wireshark.org.

b. ¿Cuál es la dirección IP del servidor DNS al que se envía la consulta DNS en el primer paquete?

El destino es un servidor DNS (205.152.37.23).

c. ¿Qué tipo de respuesta se recibe en el segundo paquete y cuál es la dirección IP asociada al nombre de dominio "wireshark.org"?

Tipo de respuesta: standard query response A.

IP: 128.121.50.122

d. ¿Cuáles son las direcciones MAC de origen y destino en ambos paquetes Ethernet?

MAC Origen (compu): 00:16:ce:6e:8b:24

MAC Destino (server dns): 00:05:5d:21:99:4c

Una dirección MAC (Media Access Control) es un identificador único. Tiene 48 bits, y se expresa generalmente en 6 pares hexadecimales, por ejemplo: 00:1A:2B:3C:4D:5E. Está asignada por el fabricante de la tarjeta de red, y suele ser única para cada dispositivo, identifica el hardware (la pc tiene una, el router, etc.).

**e. ¿Puedes explicar por qué el puerto de origen y destino cambia entre la consulta DNS en el primer paquete y la respuesta DNS en el segundo paquete?**

Porque se escribe teniendo de referencia al que envía el mensaje.

El primero (frame 1) el host, el segundo (frame 2) el server.

**22. Descargar, analizar mediante Wireshark y contestar las preguntas sobre la transmisión de datos HTTP relacionada con la descarga de un archivo de imagen desde un servidor web a partir de la captura de paquetes de HTTP.cap del siguiente enlace:**

<https://packetlife.net/captures/category/web/>

**a. ¿Qué recurso se está solicitando en el primer paquete HTTP y quién realiza la solicitud?**

**b. ¿Cuál es el código de estado de la respuesta del servidor en el segundo paquete y qué significa este código?**

**c. ¿Qué tipo de archivo se está transfiriendo según la información proporcionada en la captura?**

**d. ¿Cuál es el tamaño del contenido (en bytes) de la imagen que se está transfiriendo según la respuesta del servidor?**

**e. ¿Cuál es la longitud de la respuesta HTTP (en bytes) en el segundo paquete?**

**f. ¿Cuál es la fecha y hora en que se envió la respuesta del servidor al cliente según los datos proporcionados en la captura?**

**23. Enviar un mensaje de consulta DNS directamente desde el host en el que está trabajando para resolver la IP del servidor www.uba.ar mediante nslookup.**

**a. ¿Qué información devuelve este comando?**

```
root@comdatos:~# nslookup google.com.ar
Server:      10.0.2.3
Address:     10.0.2.3#53

Non-authoritative answer:
Name:   google.com.ar
Address: 142.250.79.131
Name:   google.com.ar
Address: 2800:3f0:4002:810::2003
```

Devuelve la IPv4 y la IPv6 de google.com.ar.

b. ¿Cuál es la dirección IP del servidor web de google.com.ar?

IPv4 142.250.79.131  
 IPv6 2800:3f0:4002:810::2003

c. ¿Cuál es la dirección IP del servidor DNS que proporcionó la respuesta al comando nslookup?

10.0.2.3 puerto 53.

Non-authoritative answer: El DNS que respondió no es el autoritativo, simplemente te da una respuesta en caché ( si el TTL no expiró, la rta es del caché local) o que obtuvo de otro (root → TLD → autoritativos).

d. La respuesta de este comando proporciona dos datos, ¿Que representa cada uno?

Brinda los datos del servidor DNS local y los datos del servidor de google.

El comando nslookup proporciona:

1. Los datos del **servidor DNS configurado localmente**, que es quien realiza la resolución del nombre de dominio (por ejemplo, Server: 10.0.2.3).
2. La **dirección IP (o direcciones) del dominio consultado**, es decir, la respuesta a la consulta DNS (por ejemplo, Address: 142.250.79.131 para google.com.ar).

e. Existen tres clases de servidores DNS: los servidores DNS raíz, los servidores DNS de dominio de nivel superior (TLD, Top-Level Domain) y los servidores DNS autoritativos, organizados en una jerarquía:

- Los servidores de nombres raíz proporcionan las direcciones IP de los servidores TLD.
- Los servidores TLD proporcionan las direcciones IP para los servidores DNS autoritativos.
- Los servidores autoritativos contienen las direcciones IP y sus nombres correspondientes de cada página web.

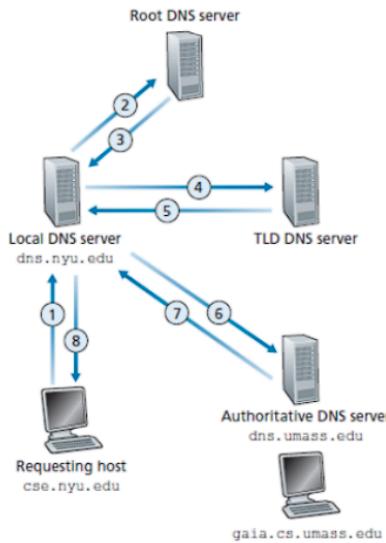


Figure 2.19 • Interaction of the various DNS servers

**¿Qué servidor, según el esquema de la figura anterior, devuelve esta información?**

El servidor autoritativo.

24) Para el comando `nslookup -type=NS fi.uba.ar` se pide:

a. ¿Qué información devuelve el comando?

El comando devuelve, en primer lugar, el nombre y la dirección IP del **el servidor DNS que está haciendo la consulta en tu máquina** que realiza la consulta.

Luego, proporciona los **nombres de los servidores de nombres (NS)** responsables de gestionar el dominio `fi.uba.ar`.

El comando `nslookup -type=NS fi.uba.ar` devuelve una lista de los servidores de nombres (NS) responsables de manejar las consultas DNS para el dominio `fi.uba.ar`.

DNS posee diferentes tipos de registros: A, AAAA, CNAME, MX, PTR, NS, ANY, entre otros. El parámetro `-type` del comando `nslookup` permite especificar el tipo de registro deseado.

Al ejecutar la siguiente consulta, se observa que el servidor que responde es el servidor DNS local (loopback):

```
Server: 127.0.0.53
Address: 127.0.0.53#53
```

```
$ nslookup -type=NS fi.uba.ar
Non-authoritative answer:
fi.uba.ar nameserver = ns4.fi.uba.ar.
```

```
fi.uba.ar nameserver = ns1.uba.ar.
```

Authoritative answers can be found from:

Como curiosidad, al repetir la consulta, se observa que ahora se obtiene una respuesta más completa desde el servidor autoritativo:

```
Server: 127.0.0.53  
Address: 127.0.0.53#53
```

Non-authoritative answer:

```
fi.uba.ar nameserver = ns1.fi.uba.ar.  
fi.uba.ar nameserver = ns1.uba.ar.  
fi.uba.ar nameserver = ns2.fi.uba.ar.  
fi.uba.ar nameserver = ns4.fi.uba.ar.
```

Authoritative answers can be found from:

```
ns1.fi.uba.ar internet address = 157.92.49.2  
ns2.fi.uba.ar internet address = 157.92.49.3  
ns4.fi.uba.ar internet address = 190.2.38.186
```

**b. ¿La respuesta al comando nslookup provino de un servidor autorizado o no autorizado?**

La primera consulta no recibe una respuesta de un servidor autoritativo, pero al consultar una segunda vez, si recibe una respuesta de un servidor autoritativo.

**c. ¿Qué significa “Respuesta no autoritativa” en la respuesta?**

Respuesta no autoritativa indica que la respuesta fue obtenida de un servidor que no oficial para ese dominio, posiblemente desde la caché del servidor local o reenviada desde otro servidor.

**d. ¿De qué tipo es el archivo de recursos que contiene la información devuelta?**

Primera respuesta (primera consulta) → NS.

Respuesta extendida (segunda consulta) → NS y A.

**e. ¿Por qué hay dos tipos diferentes de direcciones IP?**

En la salida del comando pueden observarse dos direcciones IP distintas: una corresponde al **servidor DNS local** que realiza la consulta (por ejemplo, 127.0.0.53), y las otras a los **servidores de nombres (NS)** del dominio consultado (por ejemplo, 157.92.49.2, correspondiente a ns1.fi.uba.ar).

Sin embargo, cuando se habla de *tipos* de direcciones IP desde un punto de vista técnico, se hace referencia al **formato del protocolo IP**:

- **IPv4:** direcciones de 32 bits, como 157.92.49.2.
- **IPv6:** direcciones de 128 bits, como 2001:4860:4860::8888.

Por lo tanto, el hecho de que aparezcan distintas direcciones IP en la salida del comando no implica necesariamente que se estén utilizando distintos *tipos* de IP en términos de protocolo, sino que se muestran direcciones de servidores distintos: el local y los autoritativos. En este caso particular, todas las direcciones mostradas son de tipo IPv4.

**25. ¿Qué respuesta aparece en la pantalla con el comando nslookup 157.92.1.1?**

```
C:\Users\wallo>nslookup 157.92.1.1
Server:  197-140-30-181.fibertel.com.ar
Address: 181.30.140.197

Name:    ns1.uba.ar
Address: 157.92.1.1
```

Figura 52: Comando nslookup inverso.

Realiza una consulta inversa (reverse DNS lookup), o sea, obtiene el nombre de dominio asociado a esa dirección IP.

Si en la salida no dice “Non-authoritative answer”, o dice explícitamente “Authoritative answer”, entonces sí proviene de un servidor autoritativo.

**26. DNS usa UDP en vez de TCP. Si se pierde un paquete DNS, no hay recuperación automática. ¿Provoca esto un problema y, de ser así, cómo se resuelve?**

Para contrarrestar la falta de recuperación automática, DNS emplea las siguientes estrategias:

1. Retransmisión en caso de tiempo de espera: Si el cliente que realiza la consulta DNS no recibe una respuesta en un tiempo determinado (llamado timeout), simplemente retransmite la consulta. El cliente DNS local o el servidor DNS esperará una cantidad de tiempo (generalmente entre 1 y 5 segundos) y, si no recibe respuesta, intentará nuevamente. Estos intentos se hacen varias veces antes de dar la consulta por fallida.

2. Uso de múltiples servidores DNS: Los sistemas suelen estar configurados para consultar a varios servidores DNS en caso de falla. Si un servidor DNS no responde (posiblemente debido a pérdida de paquetes), el cliente intentará hacer la misma consulta a otro servidor DNS configurado. Por ejemplo, si tu computadora está configurada con dos servidores DNS (primario y secundario), si el primario no responde a tiempo, la consulta se envía al secundario.

**27. Suponga que en UDPCliente.py, después de crear el socket, añadimos esta línea: clientSocket.bind((" ", 5432))**

**a. ¿Será necesario modificar el programa UDPServidor.py?**

No, no es necesario modificar UDPServidor.py.

Al cliente se le asigna normalmente un puerto efímero aleatorio como puerto de origen, pero si se fuerza el puerto (como en bind((" ", 5432))), eso no cambia el funcionamiento del servidor.

El servidor simplemente recibirá el mensaje desde el puerto 5432 en vez de uno aleatorio.

**b. ¿Cuáles son los números de puerto para los sockets en UDPCliente y UDPServidor luego del cambio?**

UDPCliente → puerto 5432 (porque se fuerza con .bind()).

UDPServidor → puerto fijo (no está especificado en el enunciado) que debe haber sido configurado con .bind() del lado servidor.

serverSocket.bind(('0.0.0.0', 12000)) → Escucha en todas las interfaces, puerto 12000.

clientSocket.bind((" ", 5432)) → Fuerza al cliente a usar el puerto 5432.

**c. ¿Cuáles eran antes de realizar este cambio?**

Antes de añadir la línea `clientSocket.bind((" ", 5432))`, el cliente no especificaba un puerto particular, por lo tanto el sistema operativo le asignaba automáticamente un **puerto efímero, >1023**.

UDPCliente → puerto efímero (asignado por el SO).

UDPServidor → puerto fijo (especificado en el .bind() del servidor).

**28. a) Describa cómo es el proceso y cuáles son los protocolos involucrados en cada transacción para que alice@contoso.com pueda enviarle un mensaje de correo electrónico a bob@acme.com. Realice un diagrama de entidades y protocolos.**

- ① Ana usa su UA para escribirle un mensaje e-mail para Beto, beto@barracas.edu
- ② El UA de Ana envía el mensaje a su servidor de correo; el cual es encolado para salida
- ③ Oficiando como cliente, el servidor inicia una conexión con el servidor de correo de Beto
- ④ El servidor de correo de Ana envía el mensaje de ella por la conexión
- ⑤ El servidor de Beto ubica el mensaje en la casilla de mensajes de Beto
- ⑥ Beto invoca a su UA para recuperar el mensaje y leerlo

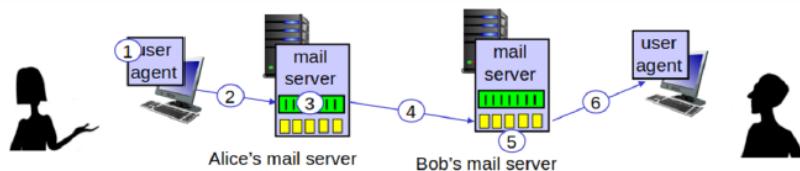


Figura 53: Descripción del proceso de envío de mail.

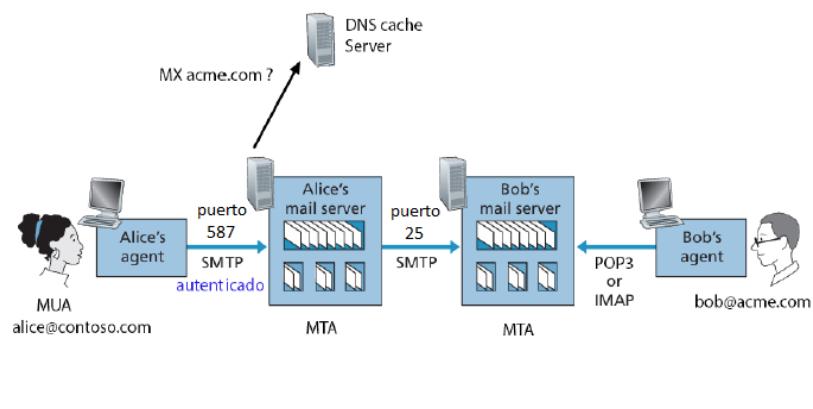


Figura 54: Diagrama de protocolos y entidades del proceso de envío de mail.

b) ¿Cómo es el diálogo entre dos SMTP servers?

```

S: 220 barracas.edu          Envelope
C: HELO almagro.edu.ar
S: 250 Hello almagro.edu.ar, pleased to meet you
C: MAIL FROM: <ana@almagro.edu.ar>
S: 250 ana@almagro.edu.ar... Sender ok
C: RCPT TO: <beto@barracas.edu>
S: 250 beto@barracas.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Salimos a cenar?
C: Puede ser pizza?          Content
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 barracas.edu closing connection

```

Figura 55: Captura del intercambio.

El protocolo SMTP está definido originalmente en la RFC 821 (1982) SIMPLE MAIL TRANSFER PROTOCOL. Que luego fue reemplazada por la RFC 2821 (2001) y luego por RFC 5321 (2008).

<https://datatracker.ietf.org/doc/html/rfc821>

Luego, el diálogo para el envío del mail se lo puede reducir a 3 pasos:

1. MAIL FROM
2. RCPT TO
3. DATA

En resumen en cada mail hay:

- un SMTP envelope
- un SMTP content y éste tiene
  - Message header
  - Message body

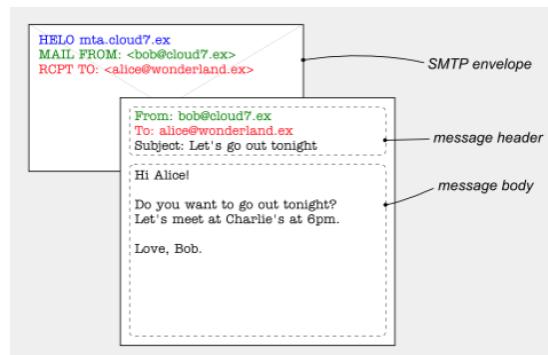


Figura 56: Contenido del mail.

### 3. Nivel de Transporte

En el Nivel de Transporte se presentan los protocolos que permiten el envío fiable o rápido de datos entre aplicaciones que están en diferentes dispositivos.

#### Características

- Implementados en los extremos (es decir en las estaciones terminales de la comunicación).
- Interfase con la aplicación.
- Interfase con el sistema operativo (tiene una implementación en el SO, no es dependiente de él sino que funciona entre SO).
- Multiplexación: colecta múltiples procesos (nos obliga a tener 'identificación' particular para cada conversación que se esté dando).  
Demultiplexación: entrega segmentos a los procesos.

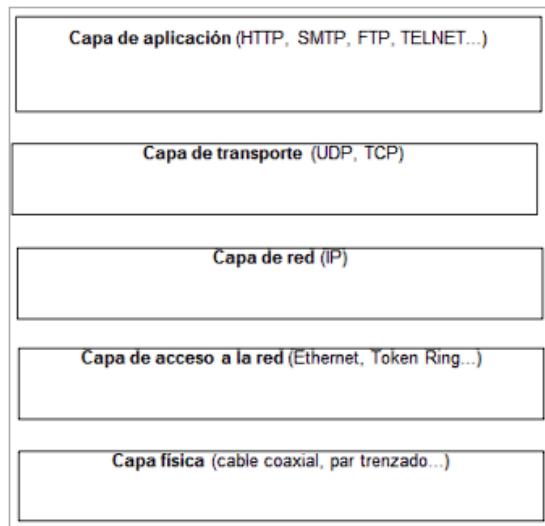


Figura 57: Modelo TCP/IP.

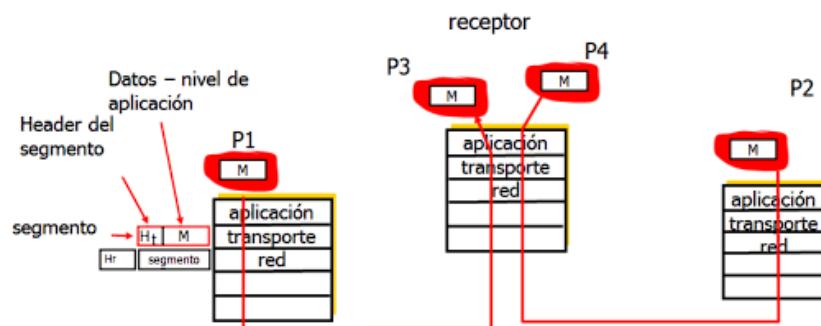


Figura 58: Funcionamiento del protocolo de transporte.

## Implementación

- Identificación:
  - Imposibilidad de extender las direcciones IP.
  - No quedan bits disponibles (en los 128 bits de una dirección IP).
  - No se pueden utilizar parámetros dependientes del sistema operativo (cada SO identifica sus procesos de forma diferente; por ejemplo, uno puede usar 16 bits y otro no).
  - Debe funcionar en todos los sistemas.
- Implementación:
  - Se crea una nueva abstracción: *protocol port number* (lo crea el SO de forma estandarizada como lo exige la RFC).
  - Es independiente del sistema operativo pero interactúa con él.
  - Utilizado en protocolos TCP/IP.
  - Reside en el SO y es compartido con diferentes aplicaciones que requieren servicios de red.

## Proceso

Se crea y se destruye (cuando no se necesita mas) como un proceso.

## Ports

Son los identificadores de procesos. Tienen una definición de 16 bits.

- De 0 a 1023: llamados *Well-Known Ports*.  
Están reservados para usos particulares asignados por la IANA (Internet Assigned Number Authority).
- Mayor a 1024: llamados *puertos efímeros*.  
Son proporcionados por el host a los procesos. Quedan reservados para la libre elección del lado cliente.

Se llama cliente siempre al que inicia la conversación en una aplicación, sin importar el modelo (cliente-servidor o pair-to-pair).

## Mensajes

La interfaz de red recibe los datos mediante mecanismos de interrupción, la cual es una señal que detiene momentáneamente lo que está haciendo el procesador. En este caso, cuando la red recibe un mensaje, se produce una interrupción y entonces:

- El sistema operativo ejecuta el código del *device driver*.
- La interrupción está asociada a una operación de I/O (entrada/salida), por ejemplo:

- `write(device, buff, len)`
- `read(...)` es similar.

## Dirección de Socket

Una dirección de socket identifica un extremo de comunicación y está compuesta por:

- Protocolo de transporte (TCP o UDP)
- Puerto local
- Dirección IP local

Ejemplo: (TCP, 2456, 193.44.234.3)

*Nota:* Puertos mayores a 1024 suelen ser efímeros (asignados temporalmente por el sistema operativo).

## Sesión (Quíntupla)

Una sesión de comunicación se describe mediante una quíntupla que incluye:

- Protocolo de transporte (TCP o UDP)
- Puerto local y dirección IP local
- Puerto remoto y dirección IP remota

Ejemplo: (TCP, 1500, 193.44.234.3, 21, 193.44.234.5)

*Notas:*

- Puerto local > 1024 → puerto efímero, normalmente usado por clientes.
- Puerto remoto 21 → puerto reservado para el servicio FTP.

### 3.1. Protocolo UDP (User Datagram Protocol)

User Datagram Protocol (RFC 768 - STD 6 - Agosto 1980).

Resumen: • simple • no confiable • Aplic ej: DNS.

#### Características

- Proporciona un procedimiento para que los programas de aplicación envíen mensajes a otros programas con un mecanismo de protocolo mínimo.
- "Best effort", significa que no garantiza nada. Los segmentos pueden perderse (osea los mensajes pueden no llegar). Por lo que la aplicación puede no importarle o tiene que saber qué hacer algo cuando esto ocurre. Los segmentos pueden llegar desordenados.

- Sin conexión, los mensajes UDP se mandan pero puede no llegar (como una carta manual). No hay garantías que la red ni el otro extremo estén disponibles para aceptar los datos. (al contrario de TCP que antes de enviar el primer mensaje se fija si esta disponible el otro extremo para recibir los datos).
  - No hay “handshake”.
  - Los segmentos son independientes.
- A cada mensaje de la aplicación le corresponde un segmento, osea un UDP. Es decir el volumen de datos que genera la aplicación se va a encapsular en un mensaje de UDP y el tamaño es limitado por tener un máximo (en cambio en TCP podría ser de tamaño infinito, podría estar generando datos permanentemente la aplicación).

### **Segmento vs Datagrama**

Un segmento es cualquier tipo de mensaje (bloque de datos) del nivel de transporte.

Segmento TCP, trabaja con segmentos que pueden fragmentarse y reordenarse. Segmento UDP, en la práctica se habla más de datagrama UDP.

En cambio un datagrama (o simplemente ”paquetes”) es un mensaje autocontenido, es decir, contiene todo el volumen de datos (no hay que esperar a otros datos mas adelante).

Datagrama UDP. Datagrama IP → capa de red.

En resumen:

- Aplicación → Datos.
- Transporte → Segmento TCP (con esos datos).
- Red → Datagrama IP (con el segmento TCP adentro).
- Enlace → Trama (con el datagrama IP adentro).
- Física → Bits en el cable o aire.

### **Estructura**

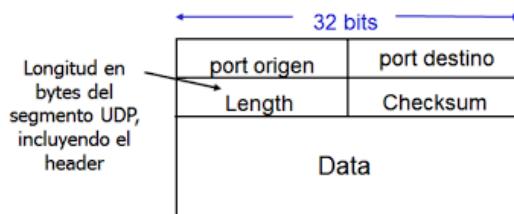
*¿Por que no se puede encapsular la aplicación directamente sobre el nivel de red, y saltar (quitar) el encapsulado con el protocolo de transporte?*

Porque a pesar que el protocolo de transporte UDP en si no aporta demasiado, las aplicaciones lo usan porque provee el nro de puerto que es lo que define la sesión.

### **Ejemplos de aplicaciones**

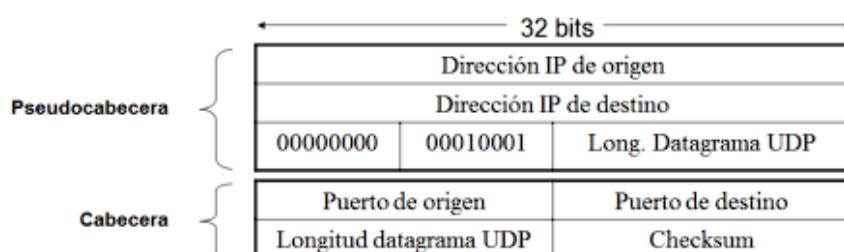
- DNS (Domain Name System) :  
Es como la guía telefónica de Internet: traduce los nombres de dominio en direcciones IP.

- **SNMP (para monitoreo de redes):**  
Permite monitorear y administrar dispositivos de red como routers, switches, servidores, impresoras y más. Es un protocolo que vos usás para obtener información o configurar remotamente esos dispositivos.
- **TFTP (Trivial File Transfer Protocol):**  
Protocolo de red muy simple que permite la transferencia de archivos entre dispositivos en una red, sin necesidad de autenticación ni navegación por carpetas.
- **RPC (Remote Procedure Call):**  
Es una técnica que permite que un programa llame a una función que se ejecuta en otra máquina, como si fuera local.
- **VoIP (Voice over Internet Protocol):**  
Es una tecnología que permite hacer llamadas telefónicas usando Internet en lugar de líneas telefónicas tradicionales.



- ✓ Checksum
- ✓ Abarca todo el segmento

Figura 59: Estructura UDP.



- La pseudocabecera se añade al principio del datagrama para el cálculo del checksum, pero no se envía.  
Permite a UDP comprobar que IP no se ha equivocado (ni le ha engañado) en la entrega del datagrama  
El valor  $10001_2 = 17_{10}$  indica que el protocolo de transporte es UDP

Figura 60: Cabecera UDP.

## Encapsulamiento

1. Lo que viene como datos se encapsula en el UDP Hdr. Cuando el mensaje esta todo armado pasa al proceso IP.
2. IP Identifica que es UDP porque reconoce el nro 17 en el IP Header. Y a partir de eso sabe como interpretar los datos con el desglosador (parser) de UDP (como una plantilla).
3. Finalmente todos los contenidos del encabezado IP (el IP Header) se van a colocar en la trama Frame que va a tener su propio encabezado (el Frame Header).

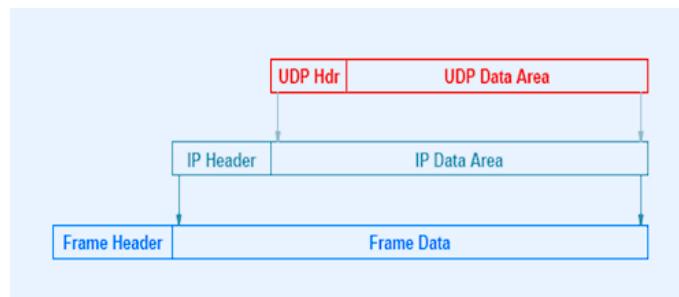


Figura 61: Encapsulado UDP.

*NOTA:* En gral los protocolos saben que llevan y transmiten es UDP lo informan con código 17 - Si es IP lo informan con código 06.

## Sesión

Si bien no hay un inicio ni cierre de sesión, la misma se da por lo que vamos a definirla porque tenemos ambos sockets conectados a una sesión porque unívocamente vamos a reconocer una aplicación lado cliente, una aplicación lado servidor conectándose e intercambiando información.

### 3.2. Protocolo TCP (Transmission Control Protocol)

Resumen: • complejo • confiable • Aplic ej: HTTP.

#### Propiedades

- **RFC 793 Std 7 - Septiembre 1981:**  
Este protocolo al ser complejo ha tenido muchas modificaciones a lo largo del tiempo con lo cual se hizo un cuerpo con todo ordenado del mismo y por eso se le dio la identificación STD-7, correspondiente a la ultima versión contiene las ultimas actualizaciones de 793.
- **draft-ietf-tcpm-rfc793bis-28:**  
Documento formal de la lectura.

- Orientado a conexión (analogía a la llamada telefónica).
 

Quiere decir que si no hay certeza que si el otro extremo de la comunicación de la sesión no está activo y disponible para recibir la conexión del otro extremo entonces no va a haber transferencia de datos.
- Confiable:
 

Porque se envía un acuse de recibo (ACK), una notificación que el receptor le hace al emisor para validar que recibió correctamente los datos. En TCP, cada segmento enviado debe ser confirmado por el receptor, de lo contrario, se retransmite.
- Ordenado:
 

Quiere decir que en destino la información procesada, se hace el acuse de recibo pero no pasa a la aplicación a menos que este en el orden que estaban presentados en los datos originales (si llegan desordenados se quedan en el buffer de recepción sin interrumpir a la aplicación).
- Ventana:
 

Son buffers de transmisión y de recepción. Nos permiten tener una cantidad finita de datos sin validar de la aplicación, es decir esos datos que el transmisor envía al receptor quedan suspendidos en una ventana de transmisión esperando la notificación de arriba.
- Control de Flujo y Congestión:
  - Flujo: La congestión da únicamente en la red en los extremos, el receptor le puede decir al transmisor que deje de transmitir/transmita menos cuando no es capaz de procesar todo lo que le está mandando.
  - Congestión: se da en la red, no distingue si es problema del receptor o la red, presupone que es este último y que son los routers los que se encuentran congestionados incapaces de enviar los segmentos que se le envían descartando datos y no lleguen a la computadora de destino.
- Byte stream:
 

La aplicación genera datos y TCP los toma de la cola de aplicación pero tiene la potestad de mandar la cantidad de bytes que mejor le parezca (a diferencia de lo que vimos en UDP donde la aplicación genera un mensaje en un encapsulado UDP).

## Características

Los datos pasan por distintas capas del modelo de red durante su transmisión por Internet. Cada capa añade su propia cabecera, y este proceso permite que múltiples aplicaciones, dispositivos y protocolos compartan el mismo medio físico (como un cable o una red Wi-Fi). A esto se lo conoce como multiplexación, y se refiere a la capacidad del sistema para manejar múltiples flujos de datos simultáneamente.

Se produce en varias capas del modelo TCP/IP u OSI, y es esencial para que múltiples procesos y protocolos funcionen al mismo tiempo en la red.

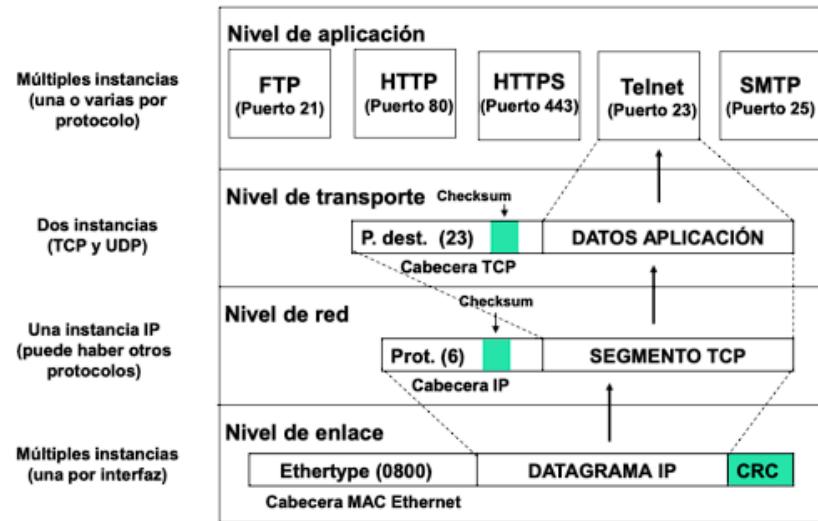


Figura 62: Multiplexación.

## Sesión

Para el caso de UDP y TCP la sesión se identifica de la misma manera.

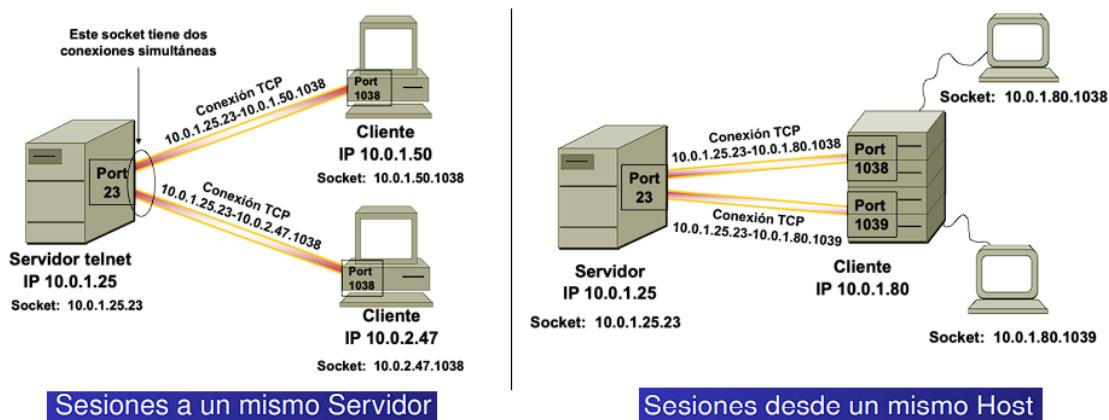


Figura 63: Sesiones a un mismo Servidor / Host.

## Inicio y fin de sesión

Las flechas representadas usualmente en el diagrama de la sesión no son rectas ya que simbolizan líneas de tiempo, e ilustran el *delay* o retardo que ocurre naturalmente en la red durante el establecimiento de la conexión. En esta fase, se activan ciertos valores de control que permiten negociar y coordinar la sesión entre los extremos.

Entre estos valores, se destacan:

- SYN: Bit de sincronismo. Señala la intención de iniciar una comunicación entre los hosts.

- ISN-X (Initial Sequence Number): Número de secuencia inicial enviado por el host  $X$ , que marca el punto de partida de la numeración de los bytes.
- MSS (Maximum Segment Size): Tamaño máximo de segmento aceptado por el host. Este valor es opcional y, si está presente, se indica durante la fase de inicio de la conexión.
- W (Ventana): Representa el tamaño máximo de bytes que el host está dispuesto a recibir sin enviar una confirmación, lo que permite controlar el flujo de datos.
- ACK-X+1: Indicador de acuse de recibo. Especifica el número de secuencia del siguiente byte que el host espera recibir, confirmando así la correcta recepción de los datos anteriores.

Para el cierre cambian los estados de los procesos de los puertos según se hayan transmitido uno u otros segmentos con los flags correspondientes al cierre.

- Se intercambian 3 segmentos
- Se envía el MSS

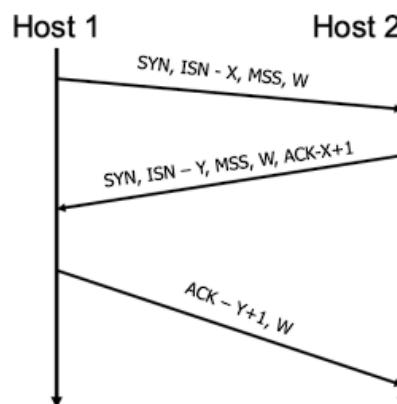


Figura 64: Establecimiento de sesión.

- Se intercambian 4 segmentos(default)

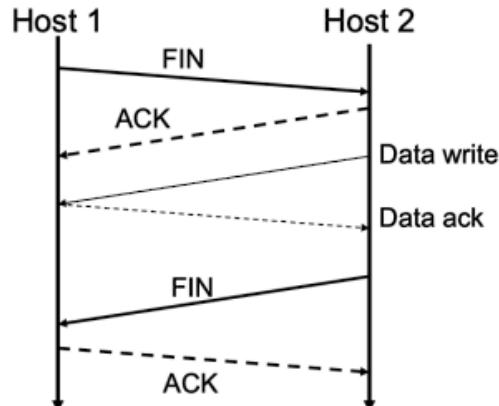


Figura 65: Fin de sesión.

## MSS

El MSS (Maximum Segment Size), si ocurre, solo puede establecerse al inicio de la conexión. Una vez fijado, no puede ser modificado durante el transcurso de la sesión.

- Máxima longitud del segmento.
- Valor por defecto: 536 Bytes.
- Puede ser diferente para cada sentido de la conexión.
- Vinculado al valor de la MTU (Maximum Transmission Unit).

## SYN estados

El establecimiento de la llamada vista como una maquina de estado.

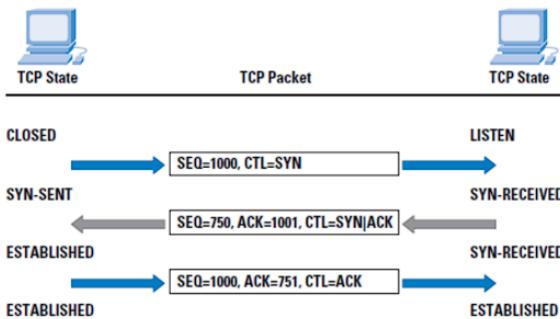


Figura 66: SYN - Estados.

De un estado de close entra a listen, cuando llega una solicitud con un control de encendido en el bit de SYN. Con el segundo segmento hace la aceptación de la llamada y entonces se da como establecida la sesión del lado inicial. Este ultimo a su vez hace el acuse de recibo ACK en el tercer envío y entonces la estación de la derecha da como establecida la comunicación.

## Netstat

Es un comando que se utiliza para monitorear el estado de la red y los protocolos. Permite ver información sobre las conexiones activas, los puertos abiertos, y los sockets definidos en un momento determinado. Es útil para diagnosticar problemas de red y verificar qué servicios están en uso.

---

U:\>netstat -a -n -p TCP			
Active Connections			
Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:12345	0.0.0.0:0	LISTENING
TCP	10.10.2.38:139	0.0.0.0:0	LISTENING
TCP	10.10.2.38:1132	0.0.0.0:0	LISTENING
TCP	10.10.2.38:1132	10.10.6.170:139	ESTABLISHED
TCP	10.10.2.38:1200	10.10.6.99:1027	ESTABLISHED
TCP	10.10.2.38:1203	10.10.6.105:1306	ESTABLISHED
TCP	10.10.2.38:1207	0.0.0.0:0	LISTENING
TCP	10.10.2.38:1207	10.10.3.228:139	ESTABLISHED

Figura 67: Netstat.

### Campos del encabezado TCP

- **Sequence Number:** identifica el número de secuencia del primer byte de los datos contenidos en el segmento.
- **Acknowledgement Number:** indica el número del próximo byte que el host espera recibir. Todos los bytes anteriores (hasta ese número menos uno) se asumen correctamente recibidos.
- **Header Length:** especifica la longitud del encabezado TCP, medida en palabras de 32 bits.
- **Flags (bits de control):**
  - URG: indica que existe información urgente. Solicita que se entregue a la aplicación el contenido del búfer hasta el valor indicado por el puntero urgente, sin esperar a completar el resto.
  - ACK: está activo en casi todos los segmentos, salvo en el primer segmento del establecimiento de conexión.
  - PSH: obliga a que los datos sean entregados de inmediato a la aplicación, sin esperar almacenamiento intermedio.
  - RST: provoca un cierre no natural de la sesión.
  - SYN: indica el inicio de la sesión (se utiliza en el establecimiento de la conexión).
  - FIN: marca el final de la sesión (señala que no se enviarán más datos).
- **Options:** campo opcional que permite incluir parámetros adicionales, como por ejemplo el valor de MSS (Maximum Segment Size).

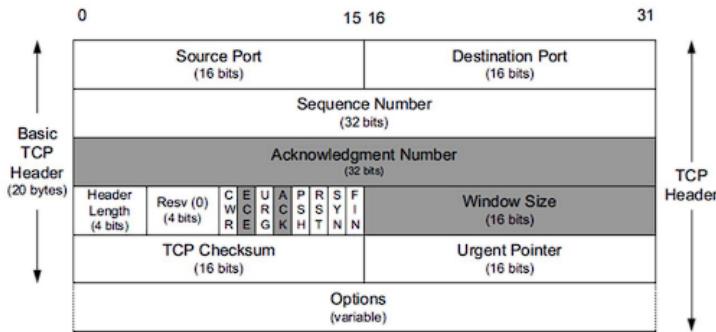
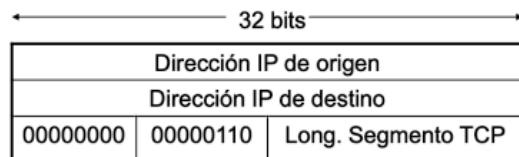


Figura 68: Segmento.



- Se construye localmente solo para la generación/cálculo del Checksum
- No se envía
- Verifica la integridad del segmento
- El valor  $110_2 = 6_{10}$  Indica protocolo TCP

Figura 69: Pseudocabecera.



- El Upper-Layer packet Length es la longitud de la cabecera y payload del protocolo de nivel superior.
- Toma el valor de la longitud del paquete en los protocolos que así lo indiquen(UDP)
- Los que no (TCP) resulta de la diferencia de la longitud del payload del header de IPv6 menos la longitud de los headers de extensiones

Figura 70: Pseudocabecera IPv6.

*Nota:* La pseudocabecera IPv6 tiene la particularidad que el encabezado tiene

longitud variable.

## Número de Secuencia

- Número de 32 bits
- Valor inicial fijado en el establecimiento de la conexión (ISN)
- Cada segmento tiene asociado el número de secuencia
- Se incrementa según los bytes transmitidos
- Indica su ubicación en el stream de bytes

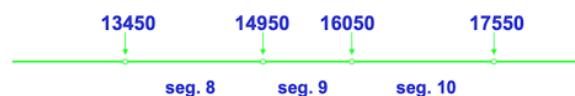


Figura 71: Número de Secuencia.

*Nota:* la linea verde es el buffer de lo que se quiere transmitir. El número de secuencia de cada segmento es el primer byte.

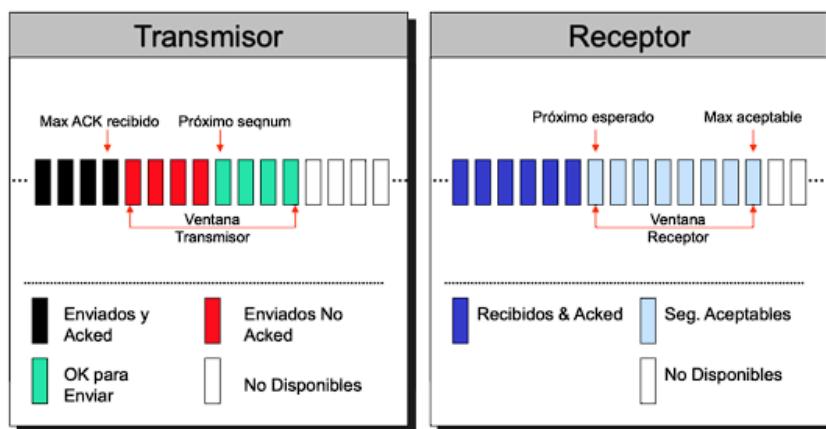


Figura 72: Ventana deslizante TCP.

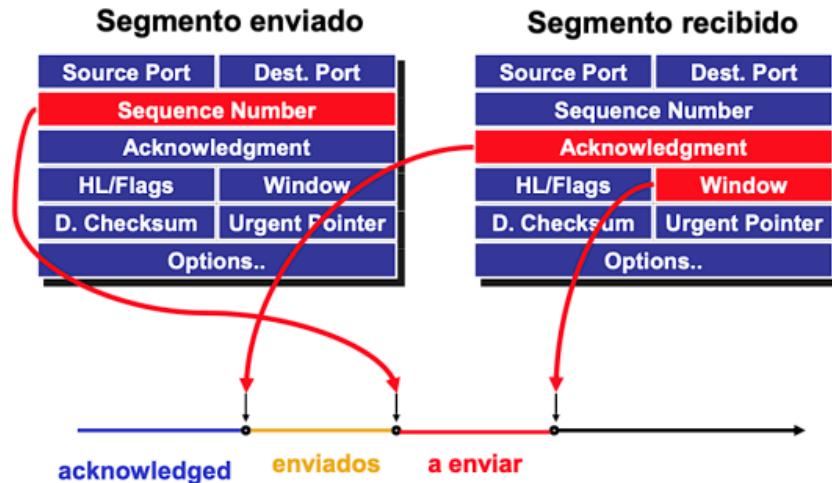


Figura 73: Ejemplo de ventana TCP.

### RTT y Timeout

a espera se calcula de forma dinámica porque los extremos no siempre presentan el mismo *delay*. Por eso, TCP ajusta constantemente sus parámetros, como el tiempo de espera antes de retransmitir, en función del comportamiento real de la red.

- RTT (Round Trip Time) { Tiempo de ida y vuelta:  
Es el tiempo que tarda un paquete en ir desde el emisor hasta el receptor y volver con una respuesta (por ejemplo, un ACK). Se mide en milisegundos.  
¿Para qué sirve? - Estimar la latencia de la red.  
No es exclusivo de TCP.
- RTO (Retransmission Timeout) { Tiempo de espera para retransmitir:  
Es el tiempo que TCP espera por un ACK antes de volver a enviar el segmento.  
¿Para qué sirve? - Saber cuándo retransmitir un segmento.  
Es exclusivo de TCP.

- **Problema:**

- El RTT varía sustancialmente
- Demasiado largo --- > Subutilización
- Demasiado corto --- > Retransmisiones inútiles

- **Solución:**

- Timeout dinámico, estimando el RTT

Figura 74: Timeout y RTT.

- ✓ Teorema de Chebyshev's
- ✓  $\text{Max RTT} = \text{Avg RTT} + k \times \text{Desv}$
- ✓  $\text{Avg rtt} = (1 - y) \times \text{Avg RTT} + y \times \text{Muestra RTT}; y = 0,1$
- ✓ Probabilidad de error  $\leq 1/(k^2)$
- ✓ Válida para toda distribución de muestras

$$\text{Timeout} = \text{Average RTT} + 4 * \text{Desv}$$

Se suele fijar en múltiplos de 200, 500, 1000 mseg

Figura 75: Estimación del RTT.

### 3.3. TCP Performance

#### Control de flujo y control de congestión

El Control de flujo se da por un ajuste de la velocidad de transmisión por una falta de capacidad en el receptor para recibir la cantidad de segmentos que le están llegando. No hay problemas en la red, los switches funcionan bien, despachando bien los segmentos pero en destino no hay suficiente espacio en buffer, no hay capacidad de procesamiento suficiente para la cantidad de segmentos que están llegando.

El control de flujo, es una acción de control que posee TCP (no todos lo tienen) y que se hace mediante el achicamiento de la ventana (W en la cabecera).

La congestión ocurre en la red (no en los equipos extremos como en control de flujo). TCP se ve obligado a tener este control porque IP no lo hace y no tiene como actuar en el router que tiene problemas, hace el control bajando la intensidad de tráfico.

**Controlar:** permanecer a la izquierda del “cliff”.

**Evitar:** permanecer a la izquierda del “knee”.

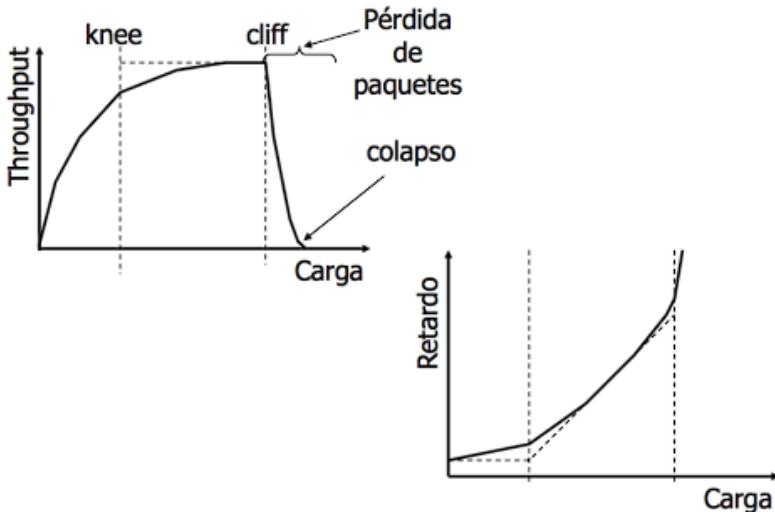


Figura 76: Throughput vs Congestión.

### Modelos de Control de congestión

- **Modelo end-to-end:**
  - Los extremos son la fuente de la demanda.
  - Los extremos deben estimar los tiempos y grado de congestión y reducir la demanda.
  - Los nodos intermedios deben monitorear el estado de la red.
- **Modelo basado en la red:**
  - Los extremos no son confiables.
  - El nodo de la red tiene control sobre el tráfico.
  - Acciones más rápidas.

### Modelo end-to-end

- Se basa en cuatro algoritmos:
  - slow start
  - congestion avoidance
  - fast retransmit
  - fast recovery
- Asocia la pérdida de segmento con congestión.
- Utiliza tres variables:
  - cwnd: ventana de congestión.
  - rsv\_win: ventana del receptor, publicada en el segmento.

- **ssthresh**: valor del umbral de **slow start**. Determina si se utiliza **slow start** o **congestion avoidance**.
- Para el envío:
  - **ventana = min(rcv\_win, cwnd)**

*¿Como detecta TCP que hay congestión en la red?*

A pesar que la congestión se da en la red TCP tiene que actuar y lo hace cuando se vence la temporalización de algún paquete, detección de que no llega la notificación entonces retransmite 'RTO' Retransmission Time (tiempo de retransmisión).

*¿Que es a Ventana en TCP?*

Es la cantidad de bytes que el emisor ha enviado y que todavía están pendientes de notificación, es decir, no han sido confirmados por el receptor. Representa todo lo que está circulando en la red sin que se sepa aún si llegó correctamente o no. Este concepto es clave tanto para el control de flujo como para el control de congestión en TCP.

El tamaño efectivo de la ventana W se calcula como el mínimo entre la ventana de recepción del receptor (informada en el campo *Window* del encabezado TCP) y la ventana de congestión del emisor (la cual depende del estado actual de la red y no es visible para el receptor).

Asocia perdida (pero no de un solo segmento porque si fuera así con un Retransmite Out, RTO, y se soluciona) de segmento con congestión.

### Slow-Start

Cuando una conexión TCP comienza, no se sabe cuánta capacidad tiene la red, así que se empieza despacio.

El algoritmo de Slow start se inicializa con la variable llamada ventana de congestión, **cwnd**. para limitar cuántos bytes se pueden enviar. El valor inicial suele ser igual al tamaño de un segmento, osea 1 MSS (Maximum Segment Size) que es anunciado por el receptor en el SYN.

Si el MSS no fue anunciado, se utiliza un valor por defecto, **536 bytes**.

Entonces, por cada RTT (tiempo ida y vuelta), la cantidad de datos que se pueden enviar se duplica, osea cada vez que se recibe un ACK, cwnd aumenta 1 MSS. Conforme van llegando los acuse de recibo, ACK, la cwnd aumenta 1 MSS, lo que provoca que dicha variable crezca exponencialmente en bytes, hasta que:  $cwnd \geq ssthreshold$  (umbral).

Ventaja: busca maximizar el rendimiento, desventaja es la detección tardía porque lo detecta cuando se perdió el dato.

## Congestion Avoidance

Se inicia este mecanismo una vez que el cwnd supera el umbral. La idea es no saturar la red y mantener un flujo constante de datos.

Como ahora ya se tiene una idea de cuánto tráfico soporta la red, se vuelve más cauteloso. Por lo que en vez de duplicarse la cwnd, aumenta 1 MSS por cada validación RTT hasta que se produzca una perdida (RTO) y en ese momento se actualiza el umbral a la mitad de lo que estaba en transito.

- ✓ Se utiliza al comienzo de la sesión o después de la recuperación de una pérdida por retransmisión
  - ✓ Inicializa el sistema y descubre la congestión rápidamente.
  - ✓ Incrementa cwnd hasta la congestión o hasta  $cwnd \geq ssthreshold$
  - ✓ Estima el óptimo cwnd
  - ✓ Detecta congestión por pérdida de segmentos
  - ✓ Desventajas
    - Detección tardía
    - Enlaces de alta velocidad --> ventanas mayores --> mayor pérdida
- ✓ Se inicia con  $cwnd = IW$ (Initial Window)
  - If  $SMSS > 2190$  bytes:  $IW = 2 * SMSS$  bytes
  - If  $(SMSS > 1095$  bytes) & ( $SMSS \leq 2190$  bytes):  $IW = 3 * SMSS$  bytes
  - If  $SMSS \leq 1095$  bytes:  $IW = 4 * SMSS$  bytes

Interesados en IW --> RFC 3390

  - ✓ Despues de cada segmento validado  $cwnd += SMSS$ (MSS del emisor)
  - Se recomienda:  
 $cwnd += \min(N, SMSS)$  ( $N$ =bytes restantes de confirmar)
  - ✓ TCP detiene el crecimiento de cwnd cuando  $cwnd \geq ssthresh$
  - ✓ Pese al incremento unitario resulta exponencial

Figura 77: Fase inicial del slow start.

Figura 78: Crecimiento exponencial de la ventana de congestión.

$cwnd \geq ssthreshold$

- ✓  $cwnd += 1$  MSS por cada RTT  
Se recomienda  $cwnd += \min(N, SMSS)$
- ✓ Continúa hasta la congestión
- ✓ Alternativa:  
 $cwnd += SMSS * SMSS / cwnd$
- ✓ Pérdida por RTO pero no retransmitió:  
 $ssthresh = \max(FlightSize/2, 2 * SMSS)$   
FlightSize son los bytes pendientes en la red.
- ✓ Si retransmitió  $ssthresh = cte$
- ✓ Luego Slow Start

Figura 79: Mecanismo Congestion Avoidance.

## Fast Retransmit - Fast Recovery

Cuando se pierde un paquete, TCP tiene dos formas de detectar la pérdida:

### ■ Esperar el RTO (Retransmission Timeout):

Se espera a que venza el temporizador RTO, que indica cuánto esperar sin recibir un ACK. Este valor suele ser aproximadamente el doble del RTT. Es un mecanismo confiable pero lento, y degrada el rendimiento de la conexión.

- Usar Fast Retransmit y Fast Recovery (más rápido):

Si se reciben **3 ACKs duplicados**, se asume que un segmento se perdió. Esto activa el mecanismo de **Fast Retransmit**, que retransmite inmediatamente el segmento perdido sin esperar el RTO.

Luego se entra en la fase de **Fast Recovery**:

- Se reduce **ssthresh** a la mitad del valor actual de **cwnd**.
- También se reduce **cwnd**, pero no al mínimo.
- Se continúa en el modo **Congestion Avoidance**, sin volver a Slow Start desde cero.

*Nota:* Se denomina TCP Reno: es una implementación de control de congestiones.

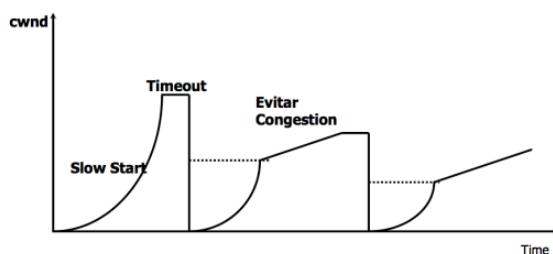


Figura 80: Congestion Avoidance con RTO.

### TCP - Cubic

Surge como mejora de performance en redes de alto BDP (Bandwidth-Delay-Product).

Evolución de BIC (Binary Increase Congestion Control) (INFOCOM 2004).

cwnd con función cubica en lugar de lineal.

Es lo que viene por Default en Linux.

### Opciones TCP

NOP (No Operation): hace de relleno, cuando hace falta alinear a 32bits el encabezado de TCP, ya que en este hay un campo que identifica la longitud del encabezado.

EOL (End Of List): es relleno. Puede haber varias opciones implementadas en el encabezado y la ultima si no quedara alineada se agrega un EOL.

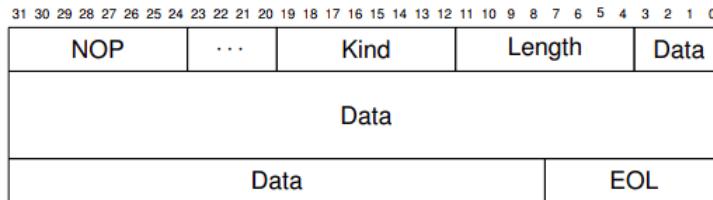


Figura 81: Opciones TCP.

Kind	Length	Name	Reference	Description and Purpose
0	1	EOL	[RFC0793]	End of Option List
1	1	NOP	[RFC0793]	No Operation (used for padding)
2	4	MSS	[RFC0793]	Maximum Segment Size
3	3	WSOPT	[RFC1323]	Window Scaling Factor (left-shift amount on window)
4	2	SACK-Permitted	[RFC2018]	Sender supports SACK options
5	Var.	SACK	[RFC2018]	SACK block (out-of-order data received)
8	10	TSOPT	[RFC1323]	Timestamps option
28	4	UTO	[RFC5482]	User Timeout (abort after idle time)
29	Var.	TCP-AO	[RFC5925]	Authentication option (using various algorithms)
253	Var.	Experimental	[RFC4727]	Reserved for experimental use
254	Var.	Experimental	[RFC4727]	Reserved for experimental use

.-Extraido de TCP/IP Illustrated 2nd. edition

.-RFC 1323 actualizada por RFC 7323

.-RFC 0793 actualizada por RFC9293-STD 7 -TCP.Agosto 2023

Figura 82: Tabla de los valores de las opciones.

### TCP Selective Acknowledge

Si se quiere usar esta opción hay definirla al inicio de la sesión, es decir no se puede establecer a la mitad.

Produce la retransmisión selectiva.

Comprende dos opciones de TCP:

- “Sack-Permitted Option”.
- “Sack Option”.

Los NOP se agregan para alinear la opción a palabras de 32 bits (4 bytes).

Left-edge y Right-edge indican el comienzo y fin del bloque de datos correcto que existe en el buffer de recepción de quien envía este segmento dado que las opciones tienen una longitud máxima de 40 bytes esto hace que no se puedan indicar más de 4 bloques.

Al recibirse se produce la retransmisión de lo necesario.

 Enviada con el SYN  Habilita el uso de SACK
<pre>+-----+-----+   Kind=4   Length=2   +-----+-----+</pre>

Figura 83: SACK Permitted Option.

#### Enviada por el receptor

<pre>+-----+-----+       NOP          NOP      Kind=5   Length   +-----+-----+   Left Edge of 1st Block   +-----+-----+   Right Edge of 1st Block   +-----+-----+   . . .                               Left Edge of nth Block   +-----+-----+   Right Edge of nth Block   +-----+-----+</pre>
--

Figura 84: SACK Option - Estructura.

## Timers

Existen timers que se agregan al manejo de la sesión:

### 1. TCP Persist Timer:

Problema: deadlocks. Solución: el transmisor cuando le llega una ventana en cero activa el Persist Timer, pregunta a los 5 seg si se puede transmitir hasta que lo habilitan.

### 2. Keep Alive Timer (opcional):

Problema: detectar si la conexión sigue viva cuando no se están enviando datos. Solucion: Si el receptor esta activo, el emisor resetea el timer por otras 2 hs. Si el receptor esta caido se genera un timeout a los 75 segundos y el emisor repite la consulta, hasta 10 veces antes de cerrar la sesión.

### 3. Wrap around - PAWS:

Problema: Wraparound: ocurre cuando el campo de número de secuencia de 32 bits se desborda. Riesgo: confusión entre segmentos nuevos y antiguos/retransmitidos o reencarnación (segmentos de sesiones cerradas anteriores).

Solución: PAWS + Timestamps: mecanismo que permite al receptor descartar paquetes con timestamps más antiguos.

## TCP Persist Timer

Se usa cuando el receptor anuncia ventana cero. Entonces TCP envía sondeos (probes) cada tanto para evitar quedar bloqueado indefinidamente (ocurre cuando cada proceso espera que el otro libere un recurso, pero ninguno lo hace, se produce un bloqueo mutuo).

Para evitar el deadlock, el transmisor cuando le llega una ventana en cero (osea un control de flujo) activa el Persist Timer, entonces pregunta a los 5 seg si se puede transmitir, y asi todas las veces necesarias hasta que lo habilite a transmitir.

## Keep Alive Timer (opcional)

Lo determina la aplicación, y lo decide en función de como tiene que funcionar, cuando es muy importante el flujo de datos que intercambia al inicio de sesión, ya que sería más costoso que se caiga la sesión y volver a establecerla, y cuando la aplicación requiere sostener sesiones que tienen largos períodos de tiempo sin intercambio de datos.

Permite detectar si la conexión sigue viva cuando no se están enviando datos:

- Si el receptor está **activo**, responderá al segmento de prueba y el emisor reseteará el timer por otras 2 hs. O que haya un **Fin-reboot**, o sea que el emisor recibe una respuesta y dentro de ella venga el bit de reset seteado y entonces se cierra la sesión.
- Si el receptor está **caído** o **inalcanzable para el emisor**, se genera un timeout a los 75 segundos y el emisor repite la consulta, hasta 10 veces en ese intervalo antes de cerrar la sesión.

Por ejemplo: TELNET, uno abre una sesión para conectarse a un equipo en la red porque quiere gestionarlo/ver qué pasa y acto seguido desconecta la computadora, con lo cual la sesión Telnet quedó colgada, el socket abierto de un lado y del otro ya no hay nadie. Entonces hay que eliminar ese recurso, por lo que mantener el Keep Alive Timer y enviar un pequeño segmento/ACK/un interrogatorio al otro extremo y esperar la respuesta.

## Wrap around - PAWS

TCP utiliza un campo de número de secuencia de 32 bits, lo que significa que puede representar  $2^{32} = 4.294.967.296$  posibles valores. A velocidades altas (como en redes gigabit), es posible que este rango se recorra completamente durante la duración de una conexión. Esto se llama wraparound, ya que los números "vuelven a empezar desde cero".

Problema: Si un número de secuencia se repite en el tiempo (es decir, aparece el mismo valor pero para un segmento completamente diferente), el receptor puede confundir un segmento viejo con uno nuevo o con una retransmisión válida, lo que compromete la integridad de la sesión TCP.

PAWS es un mecanismo definido en RFC 1323 que se basa en la opción de Timestamp TCP para evitar esta confusión.

Cada segmento lleva un timestamp (marca de tiempo).

El receptor compara el timestamp del segmento recibido con el último válido que recibió.

---

Si el timestamp es más viejo o igual, se descarta el segmento como duplicado (posiblemente de una sesión anterior o debido a wraparound).

### **Explicit Congestion Notification en TCP/IP**

El control explicito de la congestión, es opcional puede estar o no. Como mencionamos previamente sabemos que si la congestión se da es en la red. Como la red no se ocupa de resolver el problema de la congestión mas que descartando paquetes, entonces se ocupa TCP ya que le interesa que los paquetes no se pierdan y que si se necesita retransmitir, se pueda lograr. Tiempo después se desarrollo la posibilidad de que al menos IP pueda notificar la situación de congestión.

Generalidades:

- Provee un mecanismo para realizar control de congestión segun el modelo centrado en la red.
- Similar a un viejo protocolo Frame Relay.
- Actua en IP y se complementa con acciones en TCP. Dichas acciones son las vistas: retransmitir, reducir la ventana, bajar a Congestion Avoidance, etc.
- En IP procede al marcado de bits.
- En TCP las marcas generan el inicio del control de congestión según el mecanismo que tenga implementado (Reno, Vegas, Cubic o el que corresponda).

### **ECN en IP**

TOS: Type Of Service. Campo de 8 bits cuyos ultimos 2 bits se reservan para el campo de ECN (Explicit Congestion Notification).

Not-ECT: Significa que no trabaja con ECT.

ECT(1): Indica que hay un congestion.

ECT(0): Indica que no hay congestión.

CE: Notificación que existe la congestión.

### **ECN en TCP**

ECN visto desde el lado de TCP se definen dos nuevos flags: ECE y CWR, que son los bits 8 y 9 dentro del encabezado de TCP en el campo de Flags que mencionamos previamente.

¿Cómo funciona?

1. Se codifica EC(0) o ECT(1) en el datagrama.

2. Un router detecta congestión y al recibir un datagrama ECT lo despacha cambiando los bits a 11 (CE). Todo esto ocurre a nivel IP teniendo en cuenta que es producido por el Router.
3. El que recibe el datagrama con CE avisa a TCP que a la hora de enviar un ACK lo hará con el bit de Flag ECE en 1. El chequeo de CE se hace una vez por ciclo de ventana o RTT que es lo mismo.
4. Al llegar a destino (agente que provoca la congestión), este realiza dos acciones:
  - Inicia el mecanismo de control de congestión que tenga definido.
  - El próximo datagrama que envíe lo hará con el Flag de CWR activado, avisándole al otro extremo que inicio el control de congestión.
5. El otro extremo al recibir un segmento con el Flag de CWR en 1 dejará de enviar los ACK con el Flag de ECE activado.

- Se definen dos nuevos flags, ECE y CWR
  - ECE(Explicit Congestion Echo), es enviado por el que recibió el datagrama marcado (CE).
  - CWR (Congestion Window Reduced) es seteado y enviado por el que recibió el ECE, acusando recibo del ECE y avisando que ha iniciado control de congestión reduciendo su ventana de congestión.
  - La estructura de los bytes 13 y 14 del header de TCP queda:
- |                          |   |   |   |   |   |   |   |                               |   |    |    |    |    |    |    |
|--------------------------|---|---|---|---|---|---|---|-------------------------------|---|----|----|----|----|----|----|
| 0                        | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8                             | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|                          |   |   |   |   |   |   |   | C   E   U   A   P   R   S   F |   |    |    |    |    |    |    |
| Header Length   Reserved |   |   |   |   |   |   |   | W   C   R   C   S   S   Y   I |   |    |    |    |    |    |    |
|                          |   |   |   |   |   |   |   | R   E   G   K   H   T   N   N |   |    |    |    |    |    |    |
- Ahora el byte 14 quedó compuesto íntegramente por Flags.

Figura 85: ECN en TCP.

- ✓ RFC 3168
- ✓ Interactúa con TCP
- ✓ Procede al marcado de bits en el datagrama según criterio prefijado.
- ✓ Utiliza dos bits del campo de TOS del datagrama IP
- ✓ TOS:

0	1	2	3	4	5	6	7
DS FIELD, DSCP						ECN FIELD	

● Codificación del campo ECN:

+-----+-----+		
ECN FIELD		
+-----+-----+		
0      0		Not-ECT
0      1		ECT(1)
1      0		ECT(0)
1      1		CE

Not-ECT indica que el datagrama no acepta trabajar con estos bits  
 ECT(1) y/o ECT(0) indica que el datagrama está preparado para aceptar el marcado  
 CE es el caso del datagrama marcado indicando la congestión

Figura 86: ECN en IP.

Figura 87: ECN Implementación.

## Active Queue Management

A continuación se incluyen como material complementario para ilustrar el funcionamiento del mecanismo RED. No se desarrollan en profundidad en este documento, ya que su análisis excede el alcance de esta guía.

- RFC 2309 – Reemplazada por RFC 7567(2015).
- Modelo de control de congestión basado en la red.
- Controlar la longitud de la cola de salida de los datagramas.
- Monitorear el estado de la cola y calcular la longitud promedio.
- Marcar los datagramas según esa longitud promedio.
- Completa el mecanismo ECN.
- Propone diversos algoritmos.
- Ejemplo: RED (Random Early Detection)

## Nuevo Transporte

Se limita al análisis del protocolo QUIC como representante de las nuevas propuestas en el nivel de transporte, sin considerar otros como SCTP o DCCP.

Diferencias con TCP:

- QUIC está completamente cifrado de forma predeterminada, excepto la versión y algunos bits para determinar el tipo de paquete.
- QUIC retransmite datos con nuevos números de paquetes y los vuelve a cifrar. Esto mejora la detección de perdidas y la medición de RTT.
- QUIC utiliza el puerto UDP 443 del servidor en lugar de TCP 443 para HTTP/3.
- QUIC no tiene bloqueo de cabecera entre transmisiones confiables.

- Desarrollado por Google
- RFC 9000 - QUIC: A UDP-Based Multiplexed and Secure Transport - Mayo 2021
- QUIC Version 2 RFC 9369 . Mayo 2023
- Encapsula a HTTP/3 . RFC 9114 HTTP/3 Junio 2022

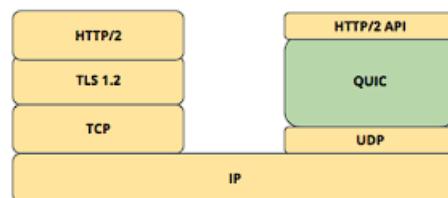


Figura 88: Protocolo Quic.

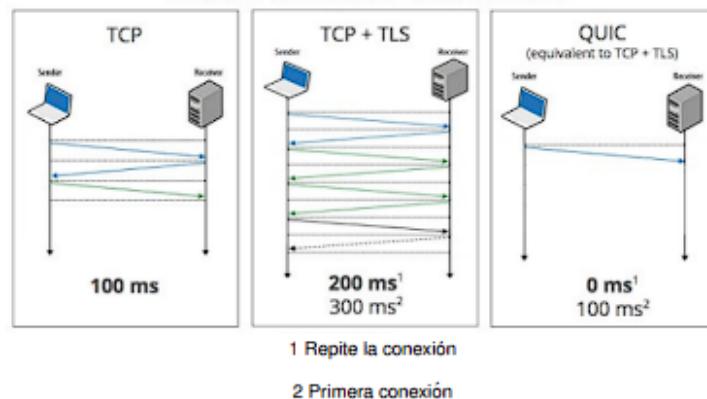


Figura 89: Quic-Sesión.

### 3.4. Aplicaciones y Ejemplos

#### 1. ¿Cómo se puede distinguir a qué aplicación debe entregar UDP el datagrama que acaba de llegar?

La distinción se da por el número de puerto. Para cada aplicación se establece un número entero (de 16 bits) que se conoce como puerto.

Están los well known ( $\leq 1023$ ) y los efímeros ( $> 1023$ ).

- Los puertos bien conocidos son fijos para determinadas aplicaciones, como el 80 para http, 53 para dns, 25 para smtp, 443 para https etc.
- Los efímeros, en cambio, se utilizan para otras aplicaciones o para conectarse a las mismas app pero desde el lado del cliente. (Un cliente que quiera usar http puede usar su puerto 2345 para conectarse al 80 de un servidor http).

A la tarea de repartir cada paquete para su puerto correspondiente le llaman demultiplexar.

#### 2. ¿Cómo se calcula el checksum de UDP?

Para calcular el checksum, se añade una pseudo-cabecera al principio del datagrama UDP (no se transmite, pero sí se usa en el cálculo).

En el emisor (quien envía el datagrama):

1. Se agrupan en palabras de 16 bits:  
 -La pseudo-cabecera.  
 -El encabezado UDP.  
 -Los datos de la capa de aplicación.
2. Se suman todas esas palabras en binario con acarreo (si hay desborde, el bit que sobra se vuelve a sumar al final).
3. Al resultado se le hace el complemento a uno (invertir los bits).
4. Ese resultado se coloca en el campo checksum del encabezado UDP.

En el receptor (quien recibe el datagrama):

1. Se vuelve a agrupar todo igual (su propia pseudo-cabecera reconstruida, encabezado, datos y el checksum recibido), también en palabras de 16 bits.
2. Se realiza nuevamente la suma con acarreo.
3. Si el resultado es todos 1s, el datagrama se considera correcto. Sino hay un error y el datagrama puede descartarse.

**3. Suponga que la ventana de congestión de TCP está en 18 Kbytes. La ventana publicada por el otro extremo de la sesión es de 64 Kbytes. ¿A qué valor llegará dicha ventana si los siguientes 5 segmentos transmitidos resultan exitosos y no se recibió aún ningún ACK? Suponga un tamaño máximo de segmento de 2 Kbytes.**

La ventana de congestión se mantiene el tamaño original de la CWND porque no se recibió ningún ACK. Es decir, la Cwnd queda en  $18 \text{ kB} = 9 \text{ MSS}$ .

Si se refiriera a la ventana del buffer sí cambia, sería  $64 \text{ KB} - (5 \times 2 \text{ KB}) = 54 \text{ KB}$ .

**4. Determine el tamaño óptimo de ventana para una sesión TCP en la que el RTT = 100 mseg, MSS = 600 bytes y velocidad de la interfaz 128 Kbps.**

$$\text{Tamaño de ventana} = \text{Velocidad de interfaz} * \text{RTT}$$

$$T = V * R = 128K \frac{\text{bits}}{\text{segundo}} * \frac{1\text{byte}}{8\text{bits}} * 100\text{ms} = 1600 \text{ bytes}$$

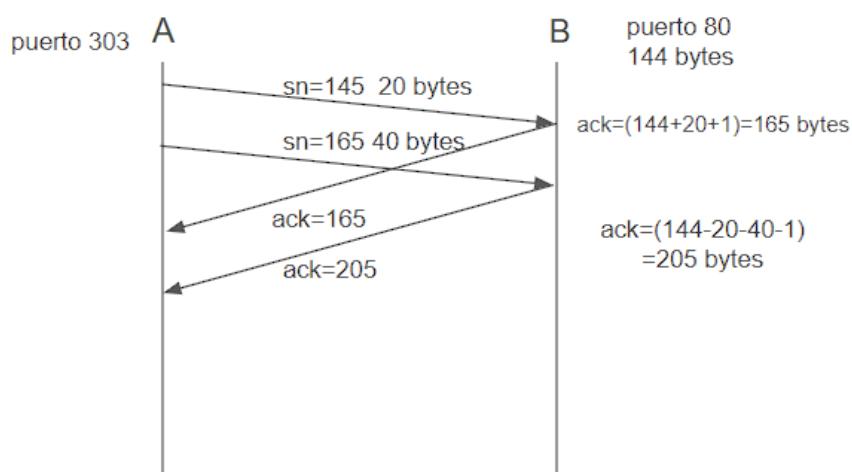
Entonces, el tamaño de ventana sería 1600 bytes. Si lo querés en segmentos sería:  $T[\text{segmentos}] = T[\text{bytes}] / \text{MSS} = 1600 \text{ bytes} / 600 \frac{\text{bytes}}{\text{segmento}} \approx 2,6 \text{ segm.} \rightarrow 3 \text{ segmentos}$

**5. Dos Hosts A y B se comunican a través de una sesión TCP. El host B recibió de A todos los bytes hasta el 144.**

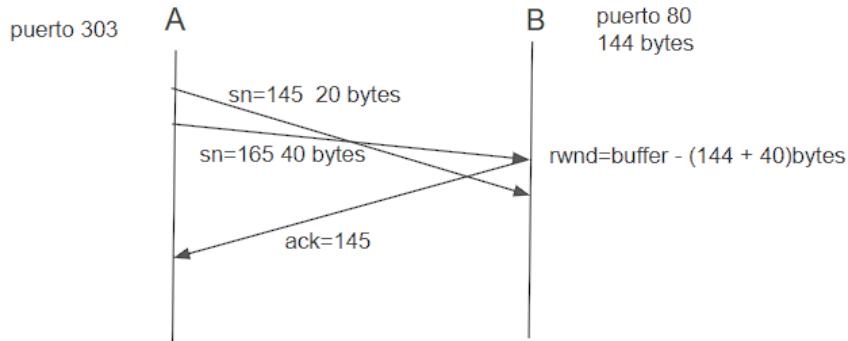
Suponga que el Host A luego envía dos segmentos a B, de 20 y 40 bytes respectivamente. En el primer segmento el número de secuencia es 145, port origen 303 y port destino 80.

El Host B envía un ACK siempre que recibe un segmento de A. Responda para cada situación:

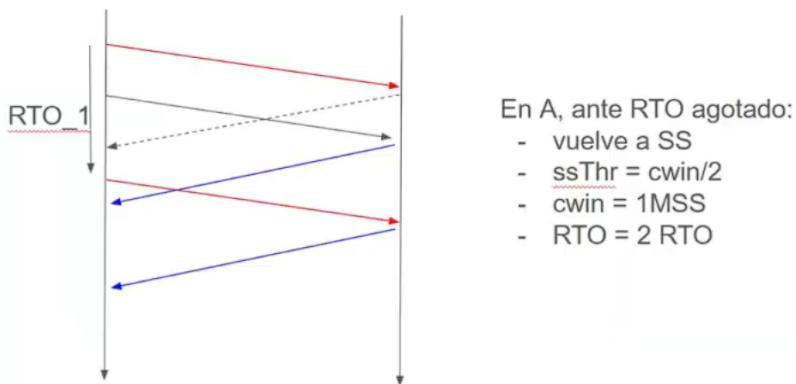
a) ¿Cuál será el número de secuencia, el número de ack, y ports origen y destino en el segmento enviado por B al recibir el segmento de 40 bytes?



- b) Si el segmento de 40 bytes llega antes que el de 20 bytes, indique campos relevantes del segmento que B enviará.



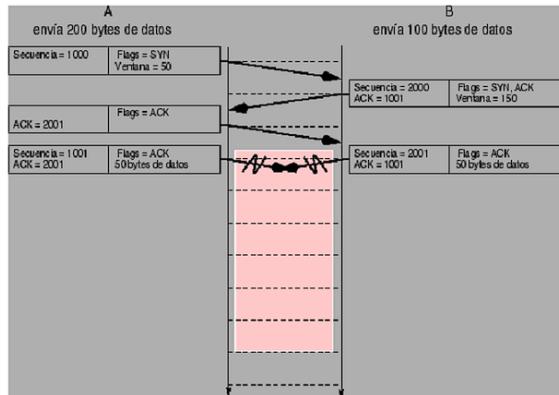
- c) Suponga que los dos segmentos enviados por A llegan a B en orden. El primer ACK se pierde y el segundo segmento llega después que el timeout del primer segmento expire. Indique los segmentos a intercambiar por parte de A y B a continuación.



6. En la secuencia de envío de segmentos TCP reflejada en la figura, en la que las líneas horizontales representan tics de reloj, se sabe que:

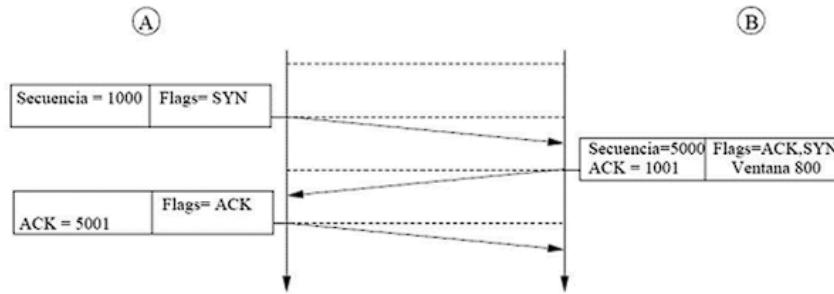
- a. A desea enviar a B 200 bytes de datos.
- b. B desea enviar a A 100 bytes de datos.
- c. A y B usan un tamaño fijo de datos de 50 bytes.
- d. A y B ajustan la ventana acorde con “congestion avoidance”.
- e. Tanto A como B sólo transmiten segmentos coincidiendo con el tic de reloj.
- f. Todos los segmentos tardan en llegar al destino medio tic de reloj, sino se pierden.
- g. A y B tienen un plazo para retransmitir segmentos de 5 tics de reloj.
- h. A y B enviarán segmentos con datos siempre que puedan.
- i. A y B enviarán un asentimiento cada vez que reciban un segmento con datos.
- j. Teniendo en cuenta que la zona sombreada indica un periodo de tiempo durante el cual todos los segmentos transmitidos se perderán y

que fuera de dicho periodo no se perderá ningún segmento, complete la transmisión en la figura (incluyendo el cierre de conexión)



7. Complete la secuencia de envío de segmentos TCP reflejada en la figura, incluyendo el cierre de la conexión, en la que las líneas horizontales representan tics de reloj, sabiendo que:

- No se perderá ningún segmento en la transmisión excepto el cuarto con datos enviados por A.
- Los segmentos no dibujados (excepto el anteriormente citado) tardarán en llegar al destino medio tic de reloj, y no se perderán.
- A está utilizando arranque lento (Slow Start) para prevenir la congestión.
- A tiene que enviar a B 800 bytes de datos, una vez enviados procederá a cerrar la conexión.
- B no desea enviar datos a A.
- B enviará asentimientos a A cuando haya recibido dos segmentos de A desde el último segmento asentido o cuando hayan sucedido 2 tics de reloj desde el último segmento recibido.
- El plazo de retransmisión de segmentos en A (timeout) es de 3 tics de reloj.
- A usa un tamaño fijo de datos de 200 bytes.
- B siempre enviará un valor de 800 en el campo de tamaño de la ventana de recepción.
- Tanto A como B sólo transmiten segmentos coincidiendo con el tic de reloj.
- A enviará segmentos con datos siempre que pueda.



cwin\_A=1MSS  
MSS=200B+header  
algoritmo=SS

A: seq(1001), ack(5001), y 200B de datos  
B: seq(5001), ack(1201), W(800)

En A: cwin\_A=2MSS  
A: seq(1201), ack(5001), y 200B de datos  
A: seq(1401), ack(5001), y 200B de datos  
B: seq(5001), ack(1601), W(800)

cwin\_A=4MSS  
A: seq(1601); ack(5001), y 200B de datos —> se pierde  
se agota RTO\_A luego de 3 tics, actualiza:  
sstrh = cwin / 2 = 2MSS  
cwin = 1MSS  
RTO = 2RTO

A: retransmite: seq(1601); ack(5001), y 200B de datos

B (2 tics después) seq(5001), ack(1801), W=800

A: seq(1801), ack(5001), FIN  
B: seq(5001), ack(1802), W=800, FIN  
A: seq(1802), ack(5002).

8. Se realizó la captura de las siguientes tramas Ethernet: (tenga en cuenta que se extrajeron los bytes de preámbulo). Se pide: Analizar los campos relevantes de la información de nivel de transporte que contienen.

Trama 1:

00 18 f8 4e 70 2f 00 50 2c a4 34 ec 08 00 45 00

00 3e 7f 5e 00 00 80 11 cf aa c0 a8 01 64 c8 2a

61 6f 04 06 00 35 00 2a 2c a8 e4 e8 01 00 00 01

00 00 00 00 00 00 03 77 77 77 08 6d 69 6e 69 6e

6f 76 61 03 6f 72 67 00 00 01 00 01 23 cd ac f2

Encabezado Ethernet (son los primeros 14 bytes):

00 18 f8 4e 70 2f → Dirección destino  
00 50 2c a4 34 ec → Dirección origen  
08 00 → Tipo: IPv4

Encabezado IPv4:

45 00 00 3e → Versión: 4 - Header length: 5 ( $5 \times 4 = 20$  bytes) - Type of Service: 00 - Total Length: 00 3e = 62 bytes  
7f 5e 00 00 → Identificación: 7f 5e - Flags y Fragment offset: 00 00  
80 11 cf aa → TTL: 80 (128) - Protocolo: 11 (UDP) - Checksum IP: cf aa  
c0 a8 01 64 → IP Origen: 192.168.1.100  
c8 2a 61 6f → IP Destino: 200.42.97.111

Encabezado UDP:

04 06 00 35 → Puerto de origen: 1030 - Puerto de destino: 53 (DNS)  
00 2a 2c a8 → Long del datagrama UDP: 42 bytes - Checksum UDP: 2c a8

Payload DNS:

e4 e8 01 00 → ID de trans: e4 e8 - Flags: 01 00 (consulta estándar, recursiva)  
00 01 00 00 → QDCOUNT: 1 - ANCOUNT: 0  
00 00 00 00 → NSCOUNT: 0 - ARCOUNT: 0  
03 77 77 77 → "www"  
08 6d 69 6e → "micr"  
69 6e 6f 76 → "inov"  
61 03 6f 72 → "a.or"  
67 00 → "g."  
00 01 00 01 → Tipo A - Clase IN  
23 cd ac f2 → Checksum o parte de la siguiente trama

Payload = Total Length IP - IP header length - UDP header length

## **9. ¿Qué son Slow Start y Congestion Avoidance? ¿Cómo intervienen sobre el tráfico?**

Slow Start y Congestion Avoidance son algoritmos utilizados por el protocolo TCP para controlar la congestión en la red durante el envío de datos. Ambos mecanismos ajustan dinámicamente la cantidad de datos que un emisor puede enviar antes de recibir un acuse de recibo (ACK), evitando así la sobrecarga de la red.

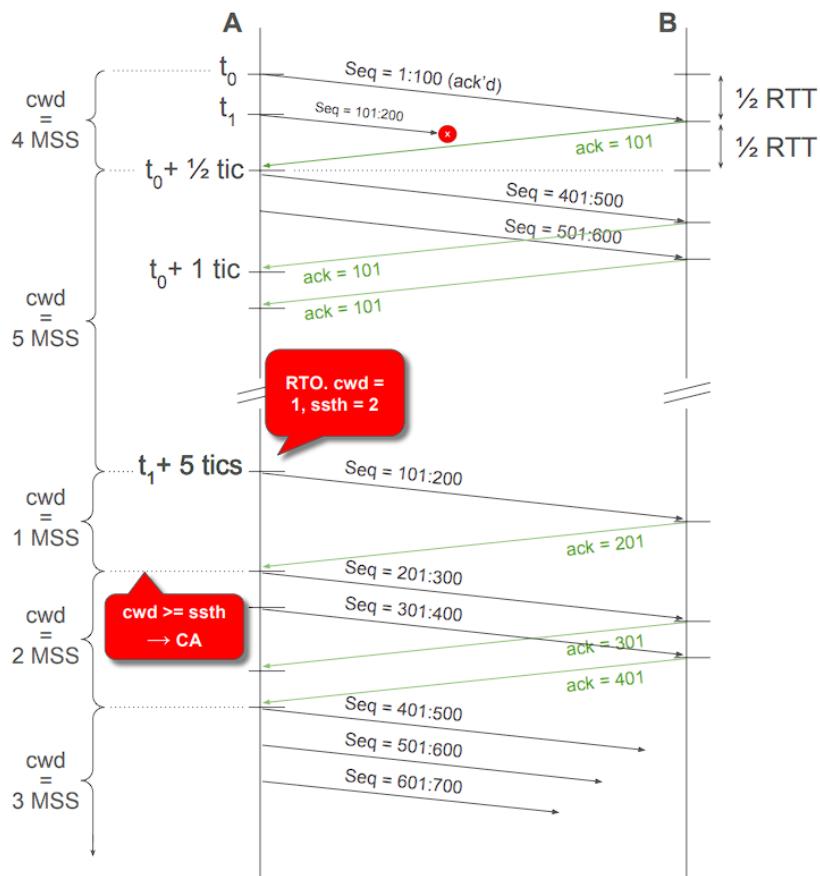
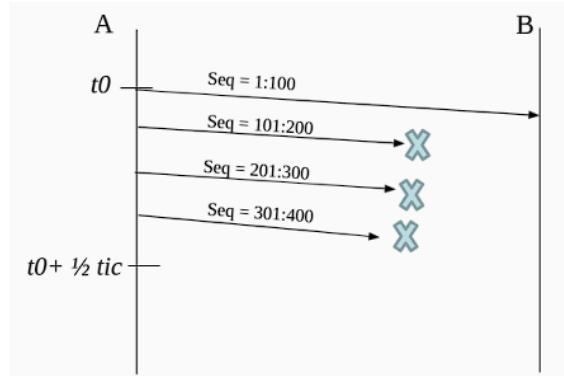
Slow Start ayuda a que el tráfico inicial no sature la red. Mientras que Congestion Avoidance regula el flujo de datos para mantener un equilibrio entre uso eficiente y prevención de congestión.

Ambas adaptan dinámicamente el envío según las condiciones de la red, evitando pérdidas excesivas y manteniendo la eficiencia.

## **10. Suponer que la siguiente sesión TCP está en curso. Algunos de sus parámetros actualmente son, para el lado A:**

RTO=5 tics; (RTT= $\frac{1}{2}$  tic)  
 cwd=4 MSSbytes; fase=SS  
 ssth=48MSSbytes.

Indicar cómo continúa la sesión considerando que ya no ocurren más errores o pérdidas de tráfico, pero A tiene todavía 12 MSS bytes para enviar a B.



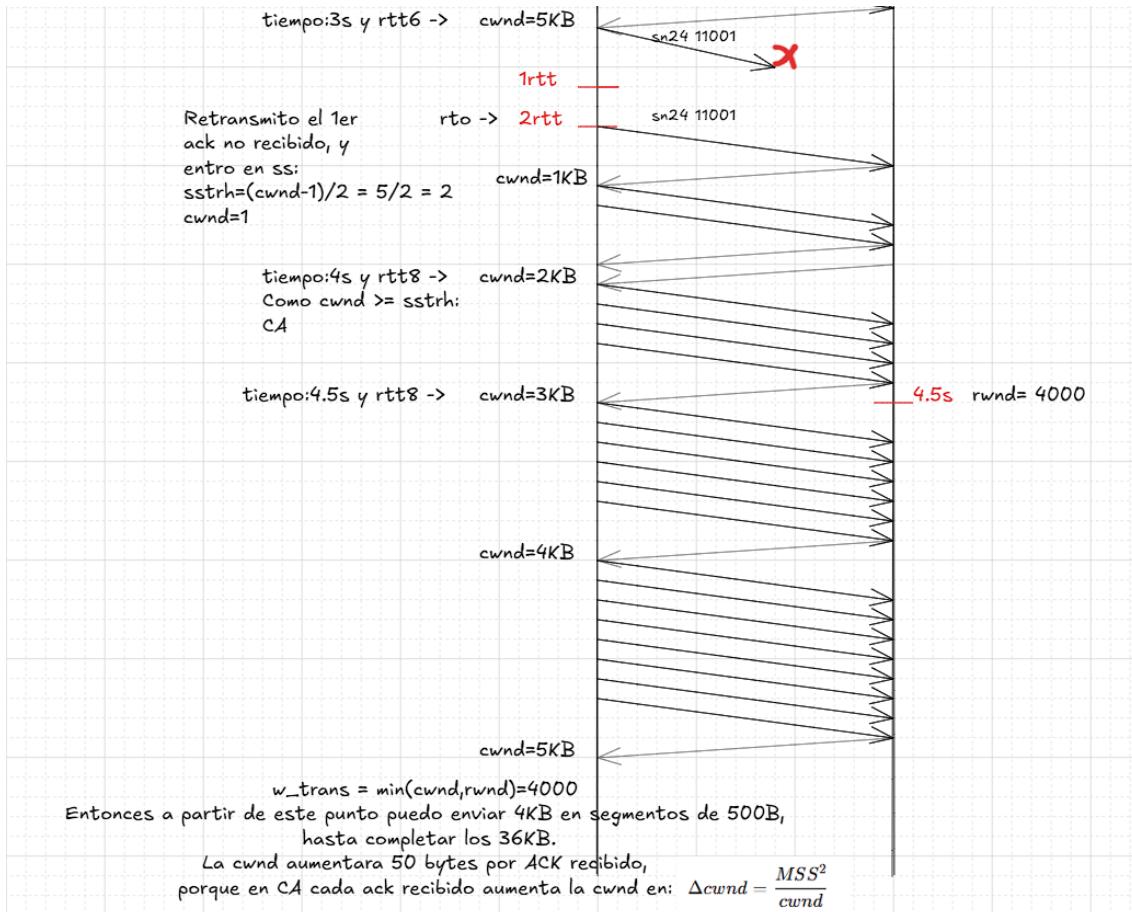
11. Sea un archivo de 36.000 bytes que debe ser enviado por TCP  
 MSS = 500 bytes  
 RTT = 500 ms  
 RTO = 2\*RTT

Rwin = 10000 bytes

SSthr = 8000 bytes

cwd = 500 bytes

¿Cuánto tiempo ocupa la transferencia si todos los segmentos transmitidos al tiempo 3s se pierden y al tiempo 4,5s el receptor actualiza Rwin=4000?



## 4. Arquitectura IP

### 4.1. Protocolo IP (Internet Protocol)

#### Características

El Protocolo de Internet es el componente central del conjunto de protocolos de Internet (TCP/IP), y tiene como objetivo principal encaminar paquetes de datos desde un origen hasta un destino a través de redes interconectadas.

**Implementado en los nodos:** El IP está presente en todos los nodos intermedios (como routers), que se encargan de encaminar los paquetes hacia su destino correcto.

**Y también en los extremos:** El protocolo también está en los extremos de la comunicación (por ejemplo, computadoras, servidores), donde se generan y reciben los datos.

**Extender los límites de las aplicaciones:** IP permite que las aplicaciones se comuniquen más allá de una red local, funcionando sobre múltiples redes (internetworking), lo que permite la escalabilidad global.

**Clasificar servicios:** A través del campo "Tipo de Servicio" (ToS) en IPv4 o "Traffic Class" en IPv6, se puede diferenciar tráfico, priorizando ciertos paquetes (por ejemplo, para voz o video en tiempo real).

**Resolver el encaminamiento:** IP se encarga de definir la ruta más adecuada que debe seguir cada paquete desde el origen hasta el destino, incluso si pasan por múltiples redes intermedias.

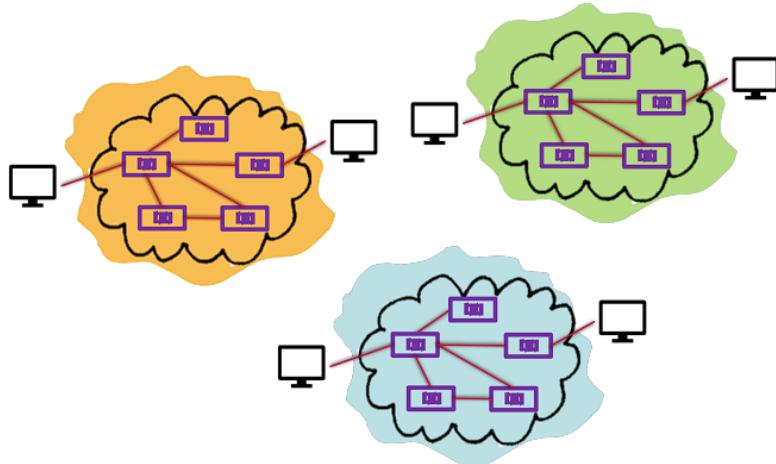


Figura 90: Redes particulares.

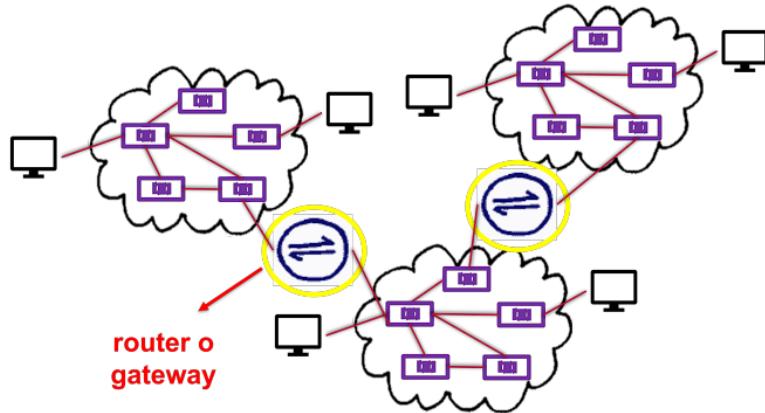
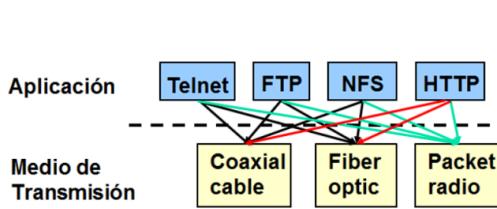
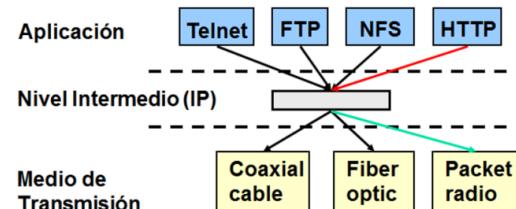


Figura 91: Interconexión mediante IP.



Las aplicaciones debían ser adaptadas a la tecnología existente.



Las aplicaciones sólo necesitan adaptarse a un contexto común.

Figura 92: Antes de IP.

Figura 93: Despues de IP.

*Nota:* Ver <https://www.rfc-editor.org/info/std5>.

## Hoja de Datos IP

### 1. Modo Best-Effort (Mejor Esfuerzo):

IP opera bajo un modelo de "mejor esfuerzo", lo que significa que intenta entregar los paquetes de datos sin garantizar su llegada exitosa, ordenada o sin duplicados. La red no proporciona confirmaciones ni retransmisiones automáticas en caso de pérdida de paquetes.

### 2. Sin Conexión (Connectionless):

IP es un protocolo sin conexión, lo que implica que no establece una conexión previa entre el emisor y el receptor antes de enviar los datos. Cada paquete se trata de forma independiente, sin necesidad de una sesión establecida.

### 3. Con Detección de Errores en la Cabecera.

IP incluye un mecanismo de detección de errores mediante una suma de comprobación (checksum) en la cabecera del paquete. Este mecanismo permite verificar la integridad de la cabecera, pero no de los datos contenidos en el paquete.

**4. Sin Garantía de Entrega:**

El protocolo IP no garantiza que los paquetes lleguen a su destino. Los paquetes pueden perderse, duplicarse o llegar dañados sin que el protocolo tome medidas para corregir estos problemas.

**5. Sin Orden de Entrega:**

Los paquetes IP pueden llegar al destino en un orden diferente al que fueron enviados. IP no proporciona mecanismos para reordenar los paquetes; esta responsabilidad recae en protocolos de capas superiores, como TCP.

**6. Sin Control de Flujo.**

IP no implementa control de flujo, es decir, no regula la cantidad de datos enviados para evitar sobrecargar al receptor. Esta función es gestionada por protocolos de nivel superior cuando es necesario.

**7. Sin Control de Congestión:**

El protocolo IP no posee mecanismos para detectar o mitigar la congestión en la red. No ajusta la tasa de envío de paquetes en función de la carga de la red, lo que puede llevar a pérdidas de paquetes en situaciones de congestión.

Estas características hacen que IP sea un protocolo simple y eficiente para el enrutamiento de paquetes, pero también implican que la fiabilidad y el control de la transmisión deben ser gestionados por protocolos adicionales en capas superiores, como TCP, cuando se requiere una comunicación más robusta.

### Datagrama IP (versión 4)



Figura 94: Datagrama IP versión 4.

**Primera palabra:**

Version(4 bits).

H Len (4 bits) : longitud del encabezamiento en palabras de 32 bits.

TOS (8 bits) : confiabilidad, prioridad, retardo y throughput.

Longitud total (16 bits) : Tamaño del datagrama en bytes.

**Segunda palabra:**

Identificación (16 bits): único para todos los fragmentos del mismo datagrama.

Flags (3 bits): X (Sin asignar), D (Don't fragment), M (More fragments).  
 Fragment offset(13 bits): ubicación de los datos respecto del del datagrama original. En unidades de 8 bytes.

#### Tercera palabra:

Time To Live(8 bits): se decrementa por cada salto.

Protocolo(8 bits): indica el protocolo que contiene el campo de datos.

Header Checksum(16 bits): control de errores del encabezado.

#### Cuarta y quinta palabras:

Dirección IP origen(32 bits): dirección del host origen.

Dirección IP destino(32 bits): dirección del host destino.

#### Opciones y datos:

Opciones: longitud variable. Seguridad, source route, echo, etc.

Padding o Relleno: longitud variable. Completa el encabezado en palabras de 32 bits.

Datos o Payload: de longitud variable en bytes. Encabezados + Datos 65.536 bytes.

### Espacio de Direcciones

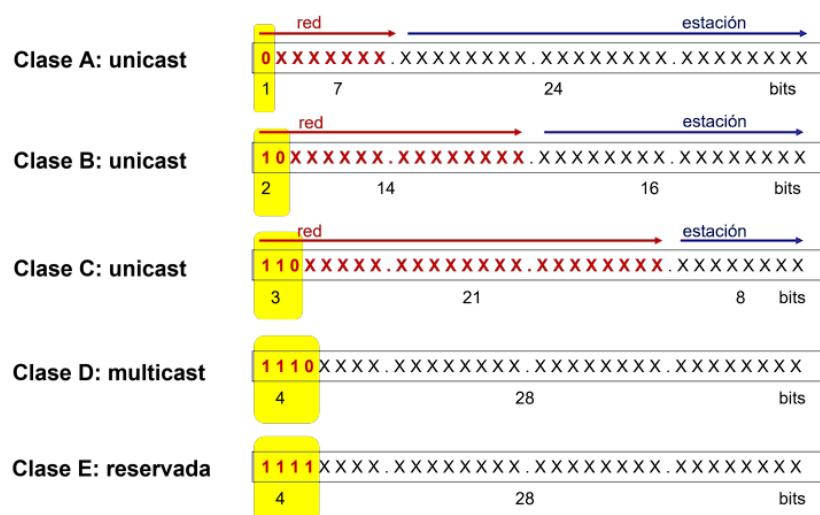


Figura 95: Clases.

#### Tipos de Direcciones (versión 4):

- **Unicast:** Identifican a un único host/interfaz. Ejemplo: 123.13.2.45.
- **Broadcast:** Identifican a todos los hosts de una red. Los bits del campo de host en uno: Ejemplo: 123.255.255.255.
- **Multicast:** Identifican a un grupo de hosts/interfaces. Rango: 224.0.0.0 a la 239.255.255.255.
- **Anycast:** Más que un tipo de dirección es un direccionamiento. Identifica a un grupo de hosts/interfaces y logra que llegue a uno de ellos según el protocolo de ruteo. La dirección se toma del espacio de las direcciones unicast.

Direcciones especiales:

- Los bits del campo de hosts en 0 identifican la red. Ejemplo: 123.0.0.0.
- Los bits del campo de hosts en 1 es la dirección broadcast de la red correspondiente. Ejemplo: 123.255.255.255 **Loopback**. Red 127.0.0.0. Los datagramas con este destino no se transmiten a la red. Produce un loopback dentro del host que la emite.

Direcciones privadas:

- Comunmente utilizadas en Intranets.
- Redes autónomas sin conexión a Internet.
- Los routers de acceso a Internet las filtran.
- RFC 1918.
- Clase A: 10.0.0.0
- Clase B: 172.16.0.0 – 172.31.0.0
- Clase C: 192.168.0.0 – 192.168.255.0

Máscara de red:

- Identificador de 32 bits.
- Prefijo: identificador alternativo. Ej: /16.
- Aplicable a rangos unicast.
- Permite reconocer la red de una dirección IP.
- Permite operar para resolver el direccionamiento o ruteo de los datagramas.

## **Direccionamiento o ruteo de datagramas IP**

### 1. Un identificador IP para cada interfaz del router.

Cada puerto físico o lógico de un router tiene una IP propia, que se usa para comunicarse con dispositivos en esa subred.

### 2. Las estaciones en la misma red tienen el mismo network ID.

Los dispositivos dentro de la misma red comparten la parte de red de sus direcciones IP. Esto es clave para saber si se puede comunicar directamente o necesita un router.

### 3. Interface: conexión entre la estación, router y enlace físico.

La interfaz es el punto de conexión (cable, WiFi, etc.) entre el dispositivo y el router.

### 4. El router actúa según la dirección IP destino de los datagramas utilizando una tabla.

El router consulta una tabla de enrutamiento para decidir por dónde enviar cada paquete, en base a la dirección IP de destino.

### 5. Modo directo vs. indirecto.

Directo: el host está en la misma red → se envía directamente.

Indirecto: el host está en otra red → se envía al router.

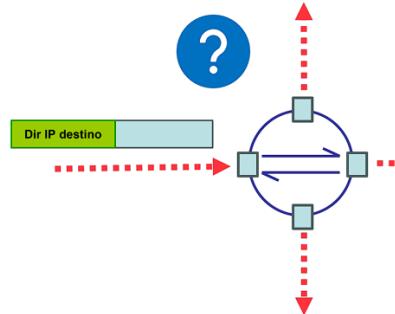


Figura 96: Router.

La figura ilustra cómo el router analiza la IP de destino de un paquete y, usando su tabla de enrutamiento, decide la mejor ruta para enviarlo. Si el destino está en otra red, lo redirige adecuadamente: esto es direccionamiento indirecto.

El rectángulo verde indica que el paquete contiene una Dirección IP de destino.

El router (círculo azul con flechas internas) recibe ese paquete.

Las flechas rojas punteadas indican las posibles rutas hacia el destino.

El símbolo "?" implica que el router necesita decidir cuál camino tomar.

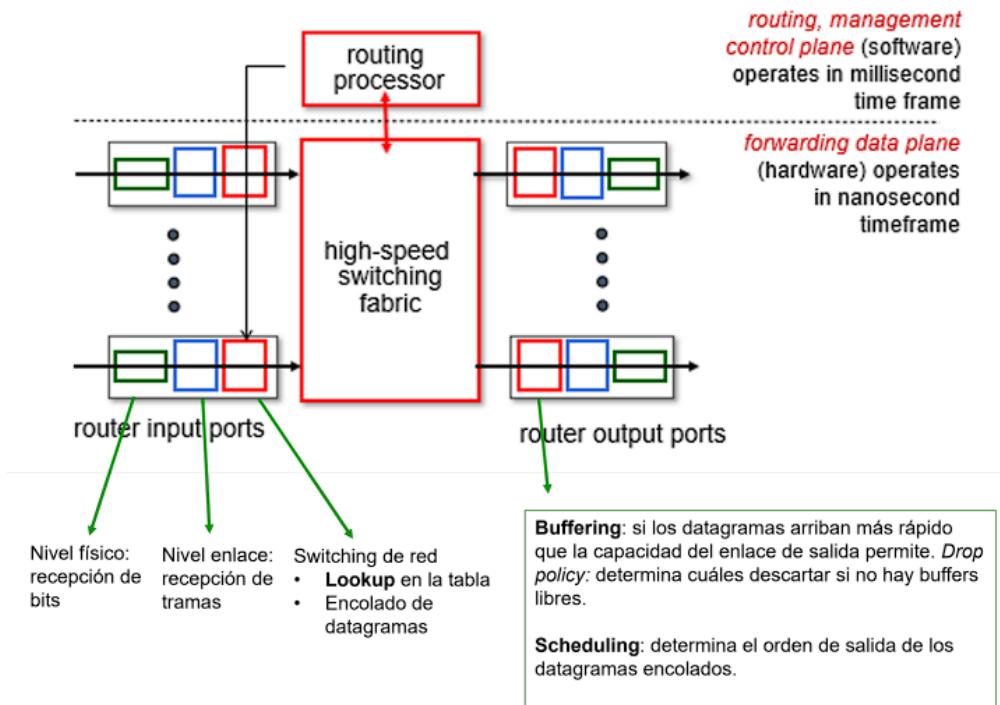


Figura 97: Arquitectura interna del router.

### Despacho o ruteo de datagramas: Tabla de ruteo

La tabla de ruteo contiene información organizada sobre redes alcanzables, lo que permite determinar la interfaz de salida y el próximo salto para un datagrama IP, en función de su dirección de destino.

Cuadro: Tabla modelo

Red	Máscara	Próximo salto	Interface
123.0.0.0	255.0.0.0	123.0.0.99	eth0
190.4.28.0	255.255.0.0	190.4.28.1	ser1
199.13.10.0	255.255.255.0	190.4.28.1	ser1

Figura 98: Ejemplo de tabla de ruteo.

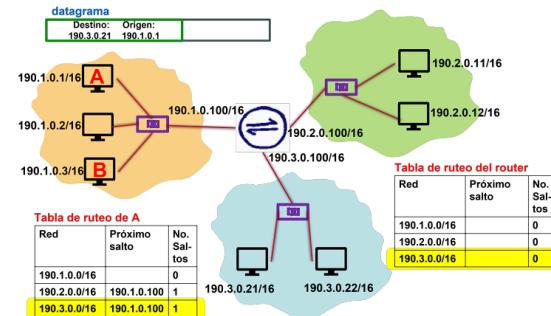
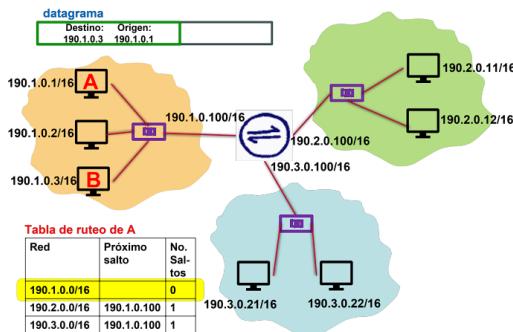


Figura 99: Direccionamiento directo.

Figura 100: Direccionamiento indirecto.

Si el ip origen y el de destino están en la misma red → direccionamiento directo,  
si están en distintas redes → direccionamiento indirecto.

### Consideraciones para el despacho local

- Origen y destino en la misma red de acceso.
- La estación origen requiere la dirección destino o de próximo salto a nivel de enlace.
- Si se usa Ethernet, esta es una MAC address.
- Si no la encuentra acude a ARP (continuará).
- El datagrama IP viaja encapsulado en una trama.
- En todo el proceso las direcciones IP no cambian.

## 4.2. Configuración de direcciones - NAT - Fragmentación - ICMP

### Configuración de direcciones

Asignación de direcciones públicas:

ICANN\* (Internet Corporation for Assigned Names and Numbers). Es el organismo internacional encargado de:

- Asignar direcciones por medio de 5 registros regionales (RRs) (que delegan localmente por países). Luego, cada país puede tener su propio NIC, por ejemplo: Argentina: NIC.ar.
- Administrar la zona raíz del DNS root zone, incluyendo la delegación de los TLD (Top-Level Domains), como .com, .org, .ar.
- Delegar los TLDs (Top-Level Domains), como .com, .org, .ar.
- Tipos de Direcciones IP:  
IPv4: 32 bits → 4.3 mil millones de direcciones (muchas ya asignadas).  
IPv6: 128 bits → cantidad prácticamente ilimitada.
- NAT (Network Address Translation) para la conversión público/privado. Dado que las direcciones IPv4 públicas son limitadas, se utiliza NAT para traducir direcciones privadas (como 192.168.0.1) a una pública, y permitir que varios dispositivos usen una única IP pública al salir a Internet.

\*Nota: <http://www.icann.org/>.

### Asignación del ISP (Internet Service Provider)

El Proveedor de Servicios de Internet puede asignar 1 o un grupo de direcciones IP.

Puede haber una delegación a otros ISPs más pequeños.

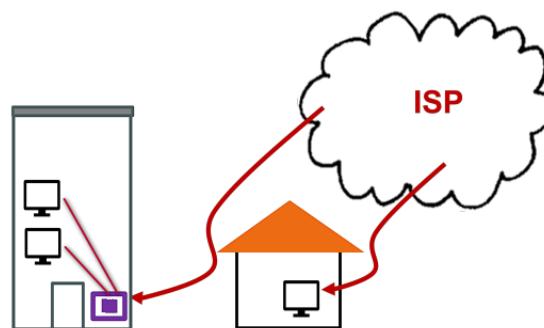


Figura 101: Asignación del ISP.

#### 1. Asignación estática individual:

Un administrador de red o un usuario que conoce un rango de direcciones IP disponibles selecciona y configura las estaciones estéticamente.

#### 2. Asignación dinámica.

DHCP (Dynamic Host Configuration Protocol) es un protocolo que se

utiliza para asignar automáticamente direcciones IP y otros parámetros de red (como puerta de enlace, máscara de subred y servidores DNS) a los dispositivos que se conectan a una red.

Modo cliente-servidor:

Aplicación sobre UDP: port 67 (S) y 68 (C).

Proceso para obtener parámetros de la red dinámicamente: IP, máscara, default gateway, servidor DNS, dominio, etc.

Los parámetros se pueden reutilizar.

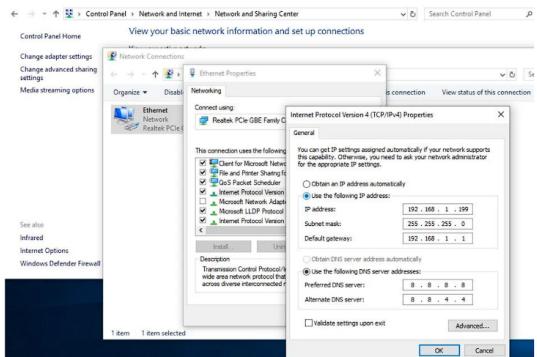


Figura 102: Ejemplo de asignación estática en Windows.

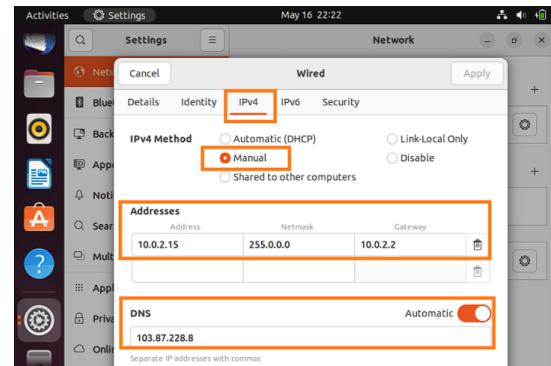


Figura 103: Ejemplo de Asignación estática en Linux.

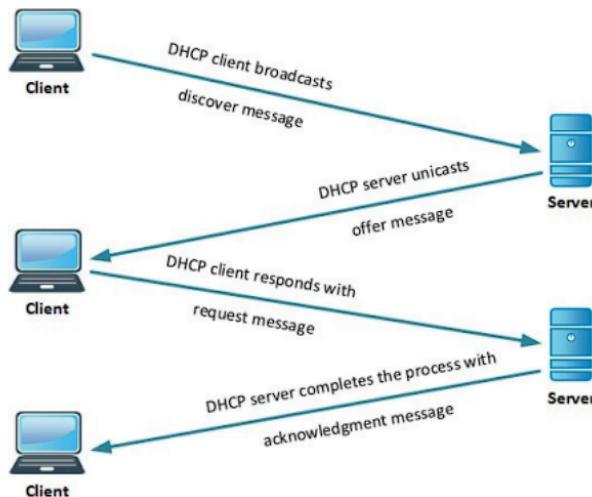


Figura 104: DHCP proceso.

## NAT (Network Address Translation)

Surge de la necesidad de aprovechar más el espacio de direcciones. Se resuelve parcialmente la escasez de direcciones. Se da en los bordes de una red a otra.

No es necesario en IPv6, porque hay muchas direcciones públicas disponibles.

Para que el router lo incorpore tiene que lo incorpore a su tabla de ruteo tiene que haber un envío desde el lado privado, sino no puede entrar nadie del lado público ya que no va a encontrar la asociación.

---

Características Nat:

- **NAT** = Network Address Translation.
- RFC 2663, 3022
- Permitir a extremos de redes privadas acceder a la red pública.
- Se implementa a partir de un *borde* en la red.
- Se conserva un registro estático o dinámico.
- Es transparente a los extremos.
- Puede impactar sobre algunas aplicaciones.
- Variantes: SNAT, DNAT, PAT, etc.

### **SNAT (Source NAT)**

Típicamente, se aplica la técnica a partir del tráfico saliente de la red privada. SNAT con PAT: Adicionalmente, se puede traducir el puerto origen de la sesión.

Ejemplo:

- Lado privado una casa, sin servidores. El mapeo en este caso es DINAMICO, y en sentido SALIENTE.

### **DNAT (Destination NAT)**

Desde la red pública surge la necesidad de acceder a la red privada. Caso típico: acceder a un servidor interno.

Ejemplo:

- Lado privado: una organización que publican un web server que debe estar disponible en la red pública, pero que internamente tiene una dirección distinta dentro de esa red osea un servidor. Dirección privada: 10.0.0.10 pero ademas se hace una reserva de las direcciones, normalmente las organizaciones tienen mas de una dirección publica.

En este caso el mapeo va a dar en el sentido entrante a la red. El primer datagrama osea el que da origen al uso de la asociación va a venir desde el lado publico, por lo tanto el mapeo es ESTÁTICO (osea hay que configurar el router y dejarlo que perdure así para cualquier hora y dia el que quiera consultar con el servidor encuentre la asociación y pueda ingresar al servicio), en el router tengo que configurar que cada vez que un datagrama llegue a mi red con dirección destino 209.1.33 con el puerto 25, lo mapeo con la ip 10.0.0.10 puerto 25. osea le doy acceso solo a esa ip y puerto.

*NOTA:* En este caso como estamos exponiendo un puerto, un servicio. A nivel IP no se puede hacer mucho contra esto, osea para inspeccionar la huella de vulnerabilidad hay que inspeccionar el contenido del segmento y eso no lo hace un router. Por lo tanto si quisiera agregar algún mecanismo de protección, se podría hacer con otra función propias de firewall.

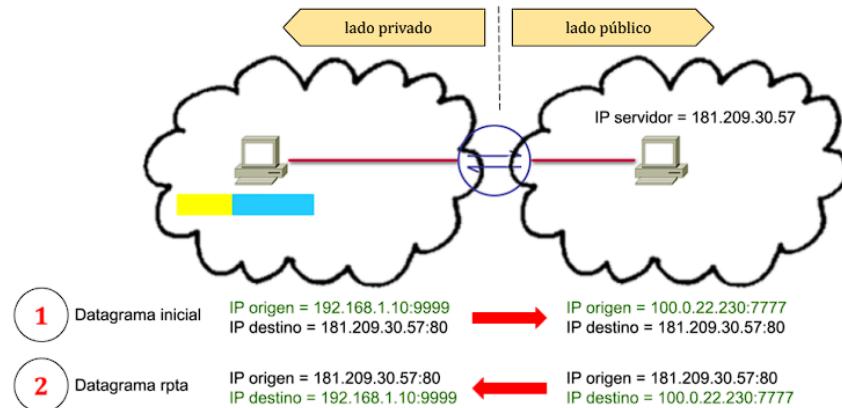


Figura 105: IP - Source NAT.

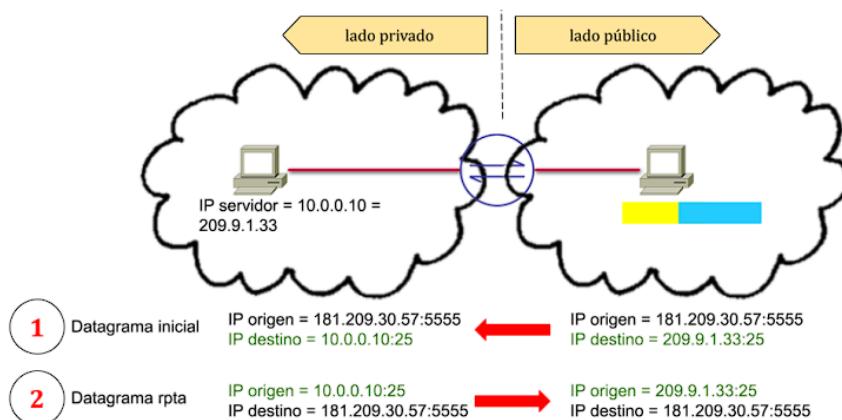


Figura 106: IP - Destination NAT.

## Fragmentación IP

- **Motivación:**

La fragmentación IP permite ajustar el tamaño de un datagrama al MTU\* del enlace que debe atravesar.

Existen mecanismos para indicar si se permite o no la fragmentación: el flag DF (Don't Fragment) en el encabezado IP señala si el datagrama puede ser fragmentado.

- **Implementación:**

Cuando se fragmenta un datagrama, el encabezado IP original se copia en cada fragmento, con las modificaciones necesarias (como por ejemplo: el campo de desplazamiento del fragmento y los flags).

Algunas opciones del encabezado se copian en todos los fragmentos.

Cada fragmento se trata como un datagrama IP independiente: puede tomar caminos distintos en la red y tener su propio enrutamiento.

Se reensambla en destino.

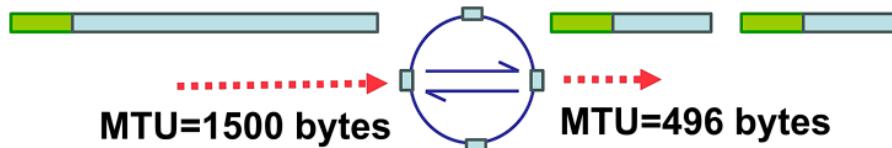
Hay un tiempo máximo de espera para los fragmentos.

Path MTU: MTU mínimo de todo el camino, para evitar la fragmentación.

El mecanismo de descubrimiento del Path MTU (PMTUD) consiste en: el host envía datagramas IP con el flag DF = 1. Si algún router intermedio no puede reenviar, lo descarta y envía un ICMP “Fragmentation Needed” al origen, indicando cuál es el MTU máximo que sí puede pasar.

\*MTU significa Maximum Transmission Unit y es el tamaño máximo de datos (en bytes) que se pueden enviar en una sola unidad a nivel de capa de enlace.

Ejemplo:



Datagrama original					
HL	t.len	ID	DF	More	Offset
5	908	1342	0	0	0

Fragmentos					
	5	492	1342	0	1
	5	436	1342	0	0

Figura 107: Ejemplo de fragmentación.

Un datagrama IP original de 908 bytes de longitud total (t.len = 908) debe atravesar un enlace con un MTU de 496 bytes.

Como el datagrama no entra completo, lo fragmenta:

HL = Header Length = 5 →  $5 \times 4 = 20$  bytes (típico).

- Datos útiles =  $908 - 20 = 888$  bytes.
- Máximo de datos por fragmento =  $496 - 20 = 476$  bytes.

El campo de offset de fragmento se mide en bloques de 8 bytes, entonces los fragmentos deben tener tamaño múltiplo de 8. Múltiplo de 8 más cercano a 476 → 472 bytes.

Finalmente: Fragmento 1 t.len = 492 (472 datos + 20 de header).  
Fragmento 2 t.len = 436 ( $888 - 472 = 416$  bytes + 20 de header).

### ICMP: Internet Control Message Protocol

Requerido, es obligatorio implementarlo. Sin embargo se puede configurar si uno lo quiere ejecutar o no, es decir cuales de los tipos de mensajes icmp se quieren usar y cuales no. Por ejemplo para no sobrecargar la operación de un router, esconder información, etc. depende de los administradores.

- No hace más confiable a IP, tampoco ordena ni controla.
- Sigue y envía información.

- Reporta errores.
- RFC 792, parte del STF 5.
- Actualizada con RFCs 950, 4884, 6633, 6918.
- Viaja encapsulado sobre IP, protocol 1 (significa que lo que va adentro del campo de datos de ip es un mensaje del tipo icmp.).



Type 0 – Echo reply  
 Type 3 – Destination unreachable (network, host, port)  
 Type 5 – Redirect Message  
 Type 8 – Echo Request  
 Type 11 – Time Exceeded  
   Code 0 - Time-to-live exceeded in transit  
   Code 1 - Fragment reassembly time exceeded.

Figura 108: Formato y tipos más habituales del mensaje icmp.

Funciona un mensaje ICMP de error:

Viaja un datagrama de la dirección de ip del usuario A hacia el B pero en el medio el ruter encuentra que hay un problema, un **Type 11 – Time Exceeded** por ejemplo.

Entonces cuando son mensajes de tipo ERROR, el router que esta adentro de la red fabrica un datagrama nuevo, escribe un mensaje ICMP indicando el tipo de error que produce el porque se tiene que descartar ese datagrama y se pone como destino la dirección origen del datagrama que esta por descartar e incluye en la parte de datos los primeros 64 Bytes del datagrama que esta por descartar (incluyen los 20 obligatorios de TCP y si fuera UDP es mas chico pero siguen siendo 64 Bytes en total) y lo descarta.

El router tiene muchas direcciones IP, normalmente va a utilizar en el datagrama que genere como IP de origen aquella que corresponde al puerto que el utilice para enviar el datagrama.

NOTA: Dependiendo del tipo y código, puede tener encabezado extendido y datos o no.

## Herramientas

Es una aplicación que funciona directamente encapsulada en IP (no usa nivel de transporte ni aplicación). Atras del header IP viene el header ICMP.

### 1. Ping:

Funciones principales: verificar si un host está disponible, detectar pérdida de paquetes y comprobar la ruta física o lógica básica hacia un destino.

- El Host origen envía mensajes ECHO REQUEST.
- Al recibir los correspondientes ECHO REPLY del Host destino, el mide reporta el tiempo de ida y vuelta.
- Luego de una serie de repeticiones, informa por pantalla el mínimo, medio y máximo del RTT.

¿Cómo funciona? → Utiliza mensaje de solicitud y envío de información que son los Tipos 0 y 8. Entonces la aplicación fabrica mensajes, datagramas, cuyo contenido es el protocolo ICMP con el tipo de mensaje ECHO REQUEST y alguna dirección destino. Y en el campo de datos un texto, por ej. En el campo de código va a poner nros incrementales (puede empezar con 1 o ser aleatorio, por default hay implementaciones que generan 5 mensajes osea 5 códigos a menos que uno le pida lo contrario). Por cada datagrama con el mensaje de echo request genera un timer, espera dentro de la vida de ese timer recibir ECHO REPLY con el mismo código de modo de reportar por pantalla el tiempo de ida y vuelta que le lleva a la aplicación conseguir la respuesta.

Finalmente luego de esa serie de repeticiones, en la pantalla aparece un reporte de tiempo mínimo, medio, máximo dependiendo como la implementación la implemente.

### 2. Traceroute:

Funciones principales: identificar la ruta que siguen los paquetes, detectar en qué punto de la red hay problemas o retrasos, Ayudar a diagnosticar fallos de conexión y mapear.

- Similar, pero en cada mensaje, incrementa el campo TTL progresivamente.
- Reportar las IPs de los routers intermedios que descartan los mensajes intermedios.
- El procedimiento termina cuando se recibe el ECHO REPLY.

¿Cómo funciona? → Utiliza mensajes ECHO REQUEST y REPLY con cada mensaje TTL incremental.

Si el TTL agotado, entonces el router le envía a la estación un mensaje diciendo que el mensaje se tuvo que descartar por TTL agotado. Entonces en la pantalla la aplicación de Traceroute me va a mostrar la IP del router que me descarto y el RTT (Round Trip Time (RTT) o tiempo de ida y vuelta) hasta ese router, osea la medición de tiempo, y genera un nuevo datagrama con un mensaje ECHO REQUEST con un TTL incrementado. Finalmente, en pantalla me muestra la lista de direcciones IP, que es la ruta que recorrió. NOTA: se puede forzar rutas con las opciones.

### 3. Path MTU Discovery:

Función principal: determinar el máximo tamaño de paquete (payload) que puede atravesar toda la ruta hacia el destino sin necesidad de fragmentación.

- Utiliza los flags del datagrama.
- Reportar cuando no puede atravesar un enlace, incluyendo el MTU

local.

- Reenvía el datagrama según el MTU recibido.

### 4.3. Subnetting - Máscara Fija - Máscara Variable

- Direccionamiento “classfull” ineficiente.
- Direcciones a punto de agotarse.
- Con subnetting:
  - Se subdivide la red classfull en varias subredes.
  - El límite entre network ID y host es variable.
  - El límite queda fijado por la máscara.
  - La máscara tiene el formato de una dirección IP.
  - Actualmente se la llama prefijo (/xx) donde xx indica la cantidad de bits que determinan la red-subred.
- RFC-917, RFC-940, RFC-950.
- Estático:
  - Todas las subredes provenientes de la misma red utilizan la misma máscara.
  - Se desaprovechan direcciones.
- Longitud variable:
  - Se pueden utilizar diferentes máscaras para cada red.

Por ejemplo:

Supongamos que se trata de una organización:

La clase A (con máscara /8) permite direccionar pocas redes, pero con una gran cantidad de hosts por red.

La clase C (con máscara /32) ofrece la posibilidad de contar con muchas redes, aunque con un número muy limitado de hosts, lo cual puede no ser adecuado para ciertas necesidades.

Por otro lado, utilizar una clase B (máscara /16) implica que las direcciones IP disponibles podrían agotarse con mayor rapidez.

Para resolver esta problemática, se creó el concepto de *subnetting*, que brinda una mayor variedad de opciones de máscaras, permitiendo así ajustar la longitud de la máscara (es decir, la cantidad de bits en 1) de acuerdo con las necesidades específicas de direccionamiento de la organización.

Este *subnetting* puede ser:

- Estático, donde todas las subredes tienen la misma longitud de máscara, o
- Dinámico, en el cual la longitud de máscara puede variar entre las subredes, adaptándose dinámicamente a diferentes requerimientos.

#### Máscara Fija

Red clase A: 12.0.0.0.

Se toman los 7 primeros bits del campo de host para definir las subredes.

Resultan 27 2 subredes.

Los 17 bits restantes definen los hosts dentro de cada subred.

Resultan 2<sup>17</sup> 2 hosts direccionables.

La mascara resultante es: 255.254.0.0.

Una de las subredes posibles: 12.24.0.0/15.

## Supernetting

- En muchos casos se asignan bloques de direcciones IP.
- Una organización que requiera 1500 direcciones IP se le asignarán 8 direcciones clase C contiguas, no una clase B, por ejemplo: 194.32.136 – 194.32.143.
- El rango considerado tiene un prefijo común de 21 bits.
- El ruteo se realiza acorde con el prefijo IP.
- El bloque se identifica como: 194.32.136.0/21.
- Es una clase C pero solo usa 21 bits para indicar máscara.
- ¡Es Classless!

## CIDR:

- Classless Interdomain Routing.
- La combinación de VLSM y Supernetting.
- Se reduce el tamaño de las tablas de ruteo.
- Se reduce el tráfico de intercambio de tablas.
- En el ejemplo anterior el router publicará: 194.32.136.0/21.
- RFC-1338, RFC-1517, RFC-1518, RFC-1519, RFC-4632.

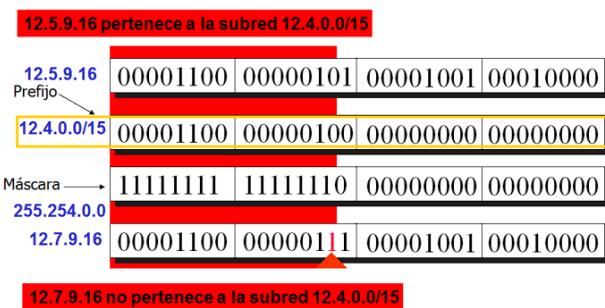


Figura 109: Ejemplo de implementación de mascara fija.

Se dispone de la red de la dirección de red indicada en la figura y con esa dirección se deben definir las tres subredes que surgen de la figura.

La subdivisión se hará no teniendo en cuenta futuro crecimiento en redes o hosts. Es decir simplemente para permitir direccionar lo que está indicado.

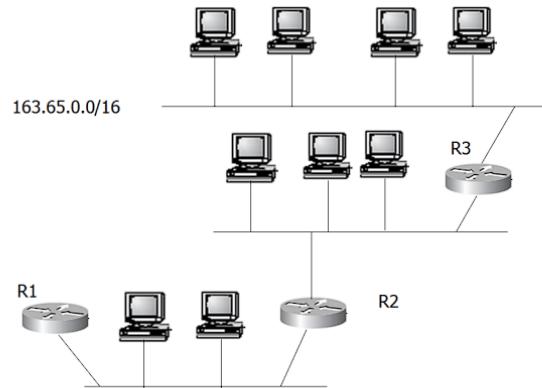


Figura 110: Construcción de máscaras.

- ✓ Necesitamos definir 3 subredes
- ✓ Tomamos 3 bits de subred. Con dos bits solo podemos definir 2 subredes. Con 3 bits 6 subredes
- ✓ No aceptamos la subred 0. (Los bits de subred en cero. Actualmente son aceptables)
- ✓ El prefijo es /19
- ✓ La máscara 255.255.224.0
- ✓ Para asignar las subredes utilizamos:  
001  
010  
011

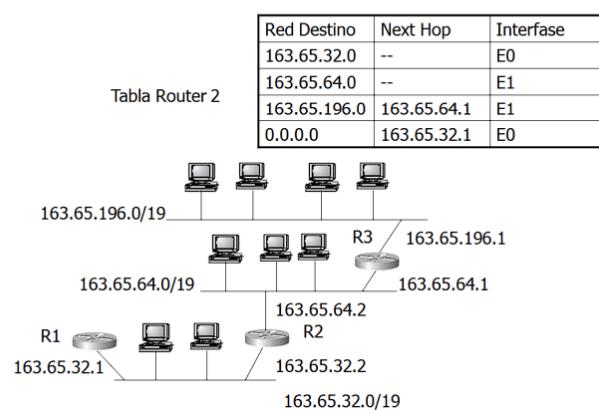
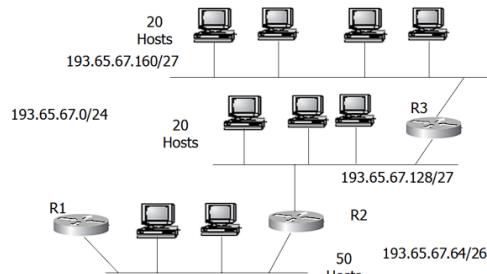


Figura 111: Ejemplo de construcción con máscara fija.



RFC-1009, RFC-1878

Figura 112: Ejemplo de construcción con máscara variable (VLSM).

#### 4.4. IPv6

El Internet actual enfrenta varios problemas debido a la escasez de direcciones IPv4 disponibles. Como solución temporal se implementó **NAT (Network Address Translation)**, que permite compartir una única dirección pública entre varios dispositivos privados. Sin embargo, esto genera:

- Un aumento excesivo en el número de entradas en las tablas de ruteo.
- Cuellos de botella en los routers, ya que deben manejar y procesar muchas rutas.
- Incremento progresivo en el tiempo de búsqueda para resolver nombres DNS y en la latencia general.

#### Características de IPv6

- Diseñado para **evitar los inconvenientes** mencionados con IPv4 y NAT.
- Proporciona un espacio de direcciones mucho más amplio para **conectar todos los dispositivos** de manera única y global.
- Facilita el **desarrollo de Internet e Internet móvil**, soportando mejor la movilidad y el crecimiento exponencial de dispositivos.
- Incluye soporte nativo para **QoS (Calidad de Servicio)**, mejorando el manejo de tráfico según prioridades y tipos de servicio.
- Mayor espacio de direcciones.
- Auto-configuración (stateless).
- Seguridad intrínseca en IP (IPSec integrada).
- QoS y CoS para gestión avanzada del tráfico.
- Soporte para Multicast.
- Soporte para Anycast.
- No hay fragmentación en los routers (fragmentación solo en el host origen).
- Payload mayor a 65.535 bytes (jumbograma).

- Datagramas alineados a 64 bits, acorde con las capacidades de los procesadores actuales.
- Cabecera de longitud fija, que facilita el procesamiento en los routers.
- Movilidad.
- Concepto de nodo.

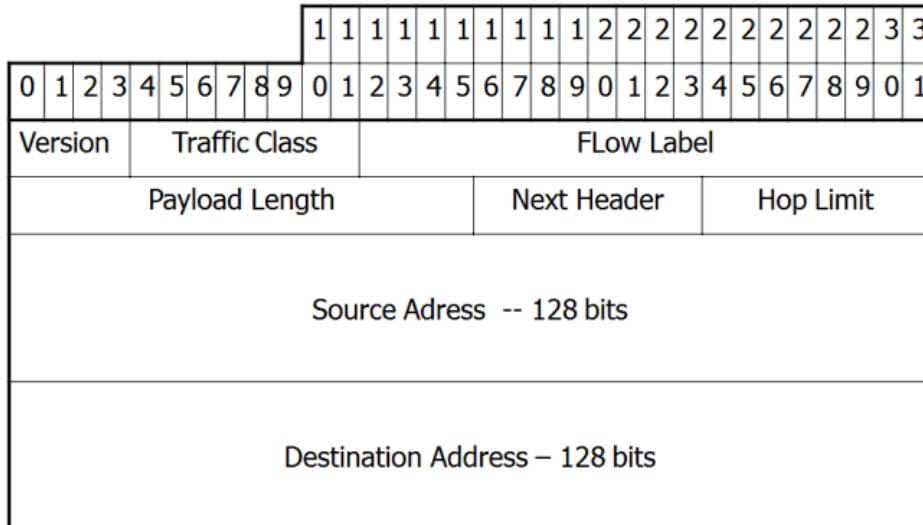


Figura 113: Cabecera IPv6.

Traffic Class (8 bits): Permite al nodo origen y/o routers diferenciar las clases y/o prioridades de los paquetes. Su uso principal está en aplicaciones de *QoS* (Calidad de Servicio). Por ejemplo, el protocolo *RSPV* utiliza el campo Traffic Class para gestionar prioridades de tráfico.

Flow Label (20 bits): Identifica un flujo de tráfico, complementando el uso del campo Traffic Class y facilitando la implementación de QoS, tal como se define en el RFC 1809 “Using the Flow Label Field in IPv6”.

El valor del Flow Label es único cuando se combina con la dirección de origen. Un valor de 0 indica que no se está utilizando.

Todos los datagramas con el mismo Flow Label deben tener la misma dirección de destino, opciones Hop-by-Hop y encabezados de enrutamiento (*routing headers*). Esto permite que un router, al observar el Flow Label, pueda enrutar eficientemente el paquete sin necesidad de examinar toda la cabecera.

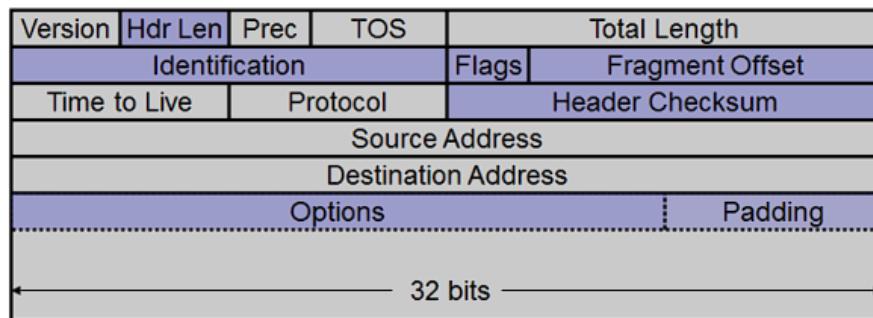
Este mecanismo sigue un modelo de QoS end-to-end, iniciado en el origen y completado en el destino, y está alineado con la idea del circuito virtual. El concepto de flujo de datos también es fundamental en la definición de *DiffServ*.

Es importante destacar que el Flow Label es unidireccional, diferenciándolo de la sesión TCP.

## Payload

Todo aquello que está más allá del encabezado IPv6 forma parte del payload.

En el caso de *jumbogramas* (RFC 2675), el campo Payload Length se pone en 0 y la longitud real del payload se especifica en una opción especial de jumbograma que utiliza un campo de 32 bits, permitiendo transportar payloads mayores a 65.535 bytes.



- ✓ Lo sombreado desapareció
- ✓ Menor tiempo de procesamiento

Figura 114: Cabecera IPv6 frente a IPv4.

## IPv4 vs. IPv6

- El header de IPv6 es el doble de largo que el de IPv4.
- Las direcciones en IPv6 son 4 veces mayores que en IPv4 (128 bits vs 32 bits).
- Desaparecen en IPv6:
  - Longitud del header.
  - Campos de fragmentación.
  - Header checksum.
- Reemplazos en IPv6:
  - Longitud del datagrama por longitud del payload.
  - Protocol type por next header.
  - TTL por límite de hops.
  - Tipo de servicio (precedencia) por clase y flow label.
- Agregados en IPv6:
  - Flow label.
  - Campos de longitud fija.
  - Campo de opciones reemplazado por headers de extensión.
  - Disminuye el tiempo de procesamiento en los routers.

## Headers de Extensión

Los headers extendidos si aparecen deben hacerlo en un determinado orden para disminuir el tiempo de procesamiento en los routers. Esto implica que si el router detecta una extensión que no debe procesar entonces deja de analizar el header por cuanto en caso de existir mas extensiones seguro que tampoco debe procesarlas.

Muchos de ellos se procesan en el destino.

El orden es importante.

Fragmentación en el origen.

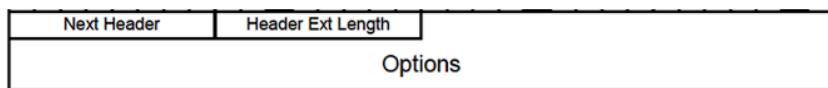


Figura 115: Header de Extensión.

## Fragmentación

Característica	IPv4	IPv6
¿Quién puede fragmentar?	El host origen y los routers intermedios	Solo el host origen
¿Hay Path MTU Discovery?	Sí, pero opcional y propenso a fallas (depende de ICMP)	Sí, obligatorio. Se basa en ICMPv6 "Packet Too Big"
¿Se usa tunneling por MTU?	No es común usar tunneling solo por MTU	Sí, se puede usar tunneling cuando no se puede evitar la fragmentación
¿Cómo se evita la fragmentación?	Puede ignorarse: los routers fragmentan si es necesario	El host origen debe ajustar el tamaño del datagrama con Path MTU Discovery
¿Cómo se hace la fragmentación?	Se hace directamente en el header base mediante los campos: <i>Identification</i> , <i>Flags (DF, MF)</i> , y <i>Fragment Offset</i>	Se agrega un <i>Fragmentation Extension Header</i> con: <i>Identification</i> , <i>Fragment Offset</i> y <i>M flag (More Fragments)</i>
¿Tiene headers de extensión?	No. Solo permite opciones dentro del header base (opcional y costoso de procesar)	Sí. Usa una cadena de headers de extensión como Routing, Hop-by-Hop, Fragment, etc.

Cuadro 2: Comparación de fragmentación, MTU y headers entre IPv4 e IPv6.

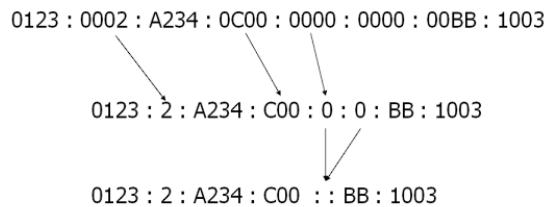
## RFC 4291 – IPv6 Addressing Architecture

Documento que define la arquitectura de direccionamiento de IPv6.

- Longitud de 128 bits

- Total de direcciones posibles:  $2^{128} = 3,4 \times 10^{38}$  (340.282.366.920.938.463.463.374.607.431.768.211.456).
- Aproximadamente  $6,65 \times 10^{23}$  direcciones por m<sup>2</sup> de la superficie terrestre (similar al número de Avogadro).
- Formato hexadecimal, con posibilidad de expresar los últimos 32 bits en decimal.

### Formato de direcciones



- ✓ Prefijo - Máscara  
0123 : 2 : A234 : C00 : 0 : 0 : BB : 1003/48
- ✓ La dirección de red :  
0123 : 2 : A234 : C00 ::/48

Figura 116: Formato de direcciones IPv6.

### Tipos de direcciones IPv6

1. Unicast.
2. Anycast.
3. Multicast (No existe broadcast): prefijo FF00::/8.

### Alcance de direcciones Unicast

1. Locales.
2. De sitio.
3. Globales: prefijo 2000::/3.

### Link Local Address:



✓ **Prefijo:** FE80::/10

✓ **Alcance:** link-local, solo la red directamente conectada

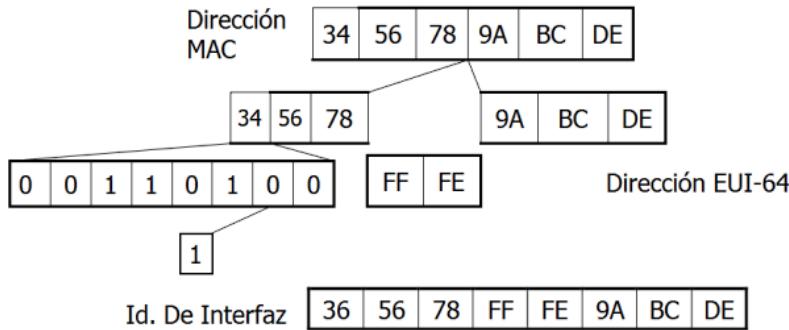
Figura 117: Direcciones Locales.

### Identificador de Interfaz

- Identifican las interfaces en un enlace.
- Deben ser únicos dentro del enlace.
- En general, coinciden con la dirección de la capa de enlace de la interfaz.
- El mismo identificador puede usarse en múltiples interfaces del mismo nodo.
- Corresponden a los 64 bits menos significativos de la dirección IPv6.
- Pueden estar basados en la dirección MAC (formato EUI-64), según RFC 2464 y RFC 6085.
- También pueden ser generados aleatoriamente para mayor privacidad, según RFC 3041 y RFC 4941.

### Direcciones especiales

- **Loopback** (::1): Dirección que representa la propia máquina (localhost). Se usa para pruebas internas de red en el mismo dispositivo.
- **Túneles Dinámicos/Automáticos de IPv6 sobre IPv4** (::<dirección IPv4>): Direcciones que permiten encapsular tráfico IPv6 dentro de paquetes IPv4 para transportar IPv6 sobre infraestructuras IPv4, facilitando la transición entre ambos protocolos.



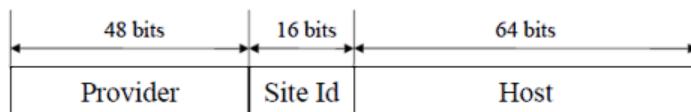
La dirección MAC se compone de:

CC:CC:CC:UU:UU:UU

Donde C identifica a la compañía y U es el identificador único.

- El bit 0 que se cambia por 1 es el Universal/Local, que en general va a estar en 0 indicando que la dirección MAC es Universal.

Figura 118: ID de Interfaz.



- ✓ **Prefijos:** Cedidos por el proveedor (ISP)

- ✓ **Alcance:** Internet

Figura 119: Direcciones IPv6 Globales.

- ✓ Actúa como identificador para un grupo de nodos

8 bits	4 bits	4 bits	112 bits
11111111	000T	ámbito	Identificador de Grupo

- ✓ El bit T indica si la dirección es permanente o temporal

- ✓ Los de ámbito:

0,3-4,6-7,9-D,F: Reservado/Asignado  
 1: Ámbito local de nodo  
 2: Ámbito local de enlace

5: Ámbito local de sitio  
 8: Ámbito local de organización  
 E: Ámbito global

Figura 120: Direcciones Multicast (RFC2375).

## Propiedades ICMPv6

- Combina funciones de **ICMP**, **ARP** e **IGMP**: En IPv6, ICMPv6 integra las funcionalidades de estos protocolos, lo que simplifica la gestión y control de la red.
- Realiza el *Neighbour Discovery* en reemplazo de ARP: En lugar del ARP clásico de IPv4, IPv6 usa Neighbour Discovery para resolver direcciones de enlace local.
- Realiza el *Path MTU Discovery* para evitar la fragmentación en la red: Determina el tamaño máximo de paquete que puede atravesar la ruta sin fragmentarse, mejorando la eficiencia y el rendimiento.
- **Next header = 58**: En los encabezados IPv6, el valor 58 indica que el siguiente protocolo es ICMPv6, responsable del control y gestión de mensajes importantes.

bits	8	16	32
Tipos	Código	Checksum	

✓ Dos tipos:

- Mensajes de error, 0 en el bit más significativo
- Mensajes informativos.

Figura 121: Estructura ICMPv6.

## Neighbour Discovery (ND) - RFC 4861 y 6980

- Equivale a ARP en IPv4.
- Utiliza mensajes ICMPv6.
- Define 5 tipos de mensajes:
  1. *Solicitud de Router (Router Solicitation)*.  
Solicita a los routers que se anuncien inmediatamente.  
Código ICMPv6 = 133.
  2. *Anuncio de Router (Router Advertisement)*.  
Enviado por los routers cada 4 a 1800 segundos o como respuesta a una solicitud, a través de multicast.  
Código ICMPv6 = 134.
  3. *Solicitud de Vecino (Neighbor Solicitation)*.  
Código ICMPv6 = 135.
  4. *Anuncio de Vecino (Neighbor Advertisement)*.  
Código ICMPv6 = 136.
  5. *Redirección (Redirect)*.  
Código ICMPv6 = 137.

- Base para la auto-configuración de IPv6.
- Descubre routers sin necesidad de protocolos de enrutamiento adicionales.
- Los anuncios de routers incluyen direcciones de enlace, evitando mecanismos adicionales.
- Los routers pueden anunciar a los hosts el MTU del enlace.
- Los mensajes de redirección contienen la dirección de la capa de enlace del nuevo salto.

## Mensajes ICMPv6

1. *Destination Unreachable (Destino Inalcanzable)*.

Se envía cuando un paquete no puede ser entregado a su destino final debido a diversas causas como falta de ruta o restricciones de seguridad.

2. *Packet Too Big (Paquete demasiado grande)*.

Indica que un paquete excede el tamaño máximo permitido en el enlace y debe ser fragmentado o reducido.

3. *Time Exceeded (Tiempo excedido)*.

Se genera cuando el campo Hop Limit llega a cero antes de alcanzar el destino, indicando que el paquete fue descartado para evitar bucles.

4. *Echo Request (Solicitud de eco)*.

Mensaje usado para solicitar la respuesta de un nodo, comúnmente utilizado para pruebas de conectividad (ping).

5. *Echo Reply (Respuesta de eco)*.

Respuesta a un Echo Request que confirma que el nodo está activo.

6. *Parameter Problem (Problema con un parámetro)*.

Indica que un paquete IPv6 contiene un campo inválido o erróneo y especifica cuál para que se corrija.

## Transición de IPv4 a IPv6

1. **Dual-stack (Doble Pila)**:

Implementación simultánea de IPv4 e IPv6 en los mismos dispositivos y redes, permite que ambas versiones coexistan en un mismo dispositivo y red.

2. **Túneles Estáticos**:

Configuración manual de túneles para transportar tráfico IPv6 sobre infraestructuras IPv4.

3. **Túneles Automáticos**:

Mecanismos automáticos para crear túneles IPv6 sobre IPv4, incluyen:

- Web tunnel broker, 6to4, ISATAP.
- NAT traversa.
- TSP tunnel broker, Teredo.

#### 4. Traslación:

Permite la comunicación entre IPv4 e IPv6.

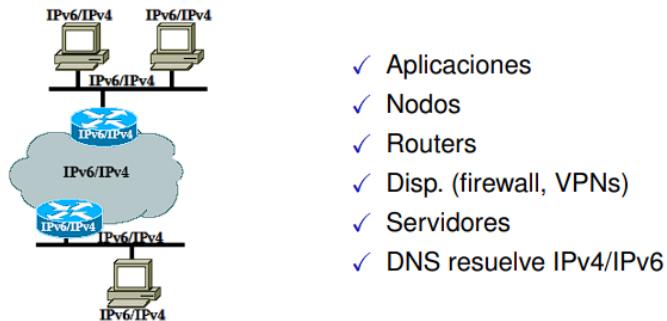


Figura 122: Dual Stack.

#### IPv6 sobre IPv4

- Router a router
- Host a router
- Host a host
- Router a host
- Protocol = 41

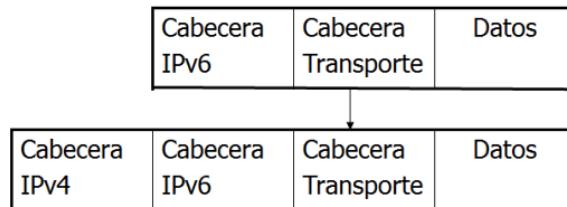


Figura 123: Túneles IPv6 sobre IPv4.

## 4.5. Aplicaciones y Ejemplos

### 1. ¿Cómo se calcula la longitud de los datos en un datagrama IP?

Long. de datos = long. datagrama - encabezado IP

Ejemplo:

Total Length = 1000 bytes

IHL (Internet header lenght) = 5 (long min) → Longitud del encabezado = 5 × 4 = 20 bytes

Longitud de los datos = 1000 - 20 = 980 bytes

### 2. Explique la función del campo de Identificación del datagrama IP.

Identifica todos los fragmentos de un mismo datagrama, el cual fue fragmentado para poder transmitirlo. Todos los fragmentos tienen el mismo número de identificación (16 bits).

### **3. ¿Cuál es la utilidad del campo TTL?**

Determina el número de saltos máximos que puede dar un datagrama, entre routers, antes de ser descartado. Se hace para que no queden datagramas perdidos circulando en la red.

### **4. ¿Es necesario y útil el protocolo ICMP? Justificar.**

ICMP = Internet Control Message Protocol, si bien no evita que se pierdan los datagramas, el ICMP es necesario porque reporta errores y es capaz de enviar y recibir información. También te da info sobre la red, como por ejemplo: las ip por donde pasa y los tiempos.

El IPv4 es opcional (pero recomendado) y en IPv6 es obligatorio.

### **5. ¿Qué tipos de mensajes puede llevar el protocolo ICMP?**

Type 0 – Echo reply

Type 3 – Destination unreachable (network, host, port)

Type 5 – Redirect Message

Type 8 – Echo Request

Type 11 – Time Exceeded

Code 0 - Time-to-live exceeded in transit

Code 1 - Fragment reassembly time exceeded.

### **6. ¿Por qué se utiliza la aplicación ‘traceroute’ para relevar las direcciones IP del camino que sigue un datagrama IP en llegar a destino en lugar de utilizar dicho datagrama con la opción IP ‘Record Route’ para almacenar las direcciones IP del camino utilizado?**

Se prefiere el ‘traceroute’ porque:

- No tiene límite práctico de saltos o IPs almacenadas.
- Es compatible con la mayoría de los routers.
- No depende de campos que suelen estar deshabilitados o filtrados (como opciones IP).
- Es mucho más confiable y ampliamente soportado.
- Mientras que el IP tiene disponibilidad limitada a la hora de almacenar direcciones IP y depende de las opciones IP.

### **7. ¿Qué significa un servicio Best-effort? ¿Se puede implementar QoS utilizando IP?**

#### **8. Indicar cuáles de las siguientes afirmaciones son correctas:**

- a) Si el offset de un datagrama IP es nulo es condición suficiente para concluir que no fue fragmentado.
- b) El host destino puede reensamblar los fragmentos de un datagrama IP.

- c) Un router puede reensamblar los fragmentos de un datagrama IP si el MTU de la red por la que forwardea puede soportar el datagrama reensamblado.
- d) Si el host destino no logra reunir los datagramas IP en un determinado intervalo de tiempo envía un mensaje ICMP de TTL Exceeded.
- e) Si un router no logra reunir los datagramas IP en un determinado intervalo de tiempo envía un mensaje ICMP de Time Exceeded.
- f) Si el TTL se hace nulo en alguno de los fragmentos, el router que procesa dicho fragmento lo descarta y envía un mensaje ICMP Destination Unreachable.

**9. Dado un datagrama de 1000 B que debe atravesar una red de MTU = 300 B, escriba los headers de todos los fragmentos.**

Datagrama 1000B y el header IP ocupa 20B → 980B de Payload (son los datos que se tienen que enviar)

MTU 300B → 280B son de datos y 20B son de header

Entonces:  $280 * 3 = 840$ B (3 fragmentos de tamaño 280B)

Faltan  $980 - 840 = 140$ B (1 fragmento de 240B)

Finalmente,

- El datagrama se fragmenta en 4:
  - 3 datagramas de 300B → 280B son datos + 20B header)
  - 1 datagrama de 160B → 140B de datos + 20B header)
- Header:
  - Nro. de identificación = todos iguales Flag DF (Don't Fragment) = 0 xq se fragmentó
  - Flag MF (More Fragments) = todos en 1 menos el último (el de 160 B)
  - Fragment offset:
    - Datagrama 1 (280 B) = 0
    - Datagrama 2 (280 B) =  $280/8 = 35$
    - Datagrama 3 (280 B) =  $2*280/8 = 70$
    - Datagrama 4 (160 B) =  $3*280/8 = 105$

El campo "Fragment Offset." en el header IP indica la posición del fragmento en múltiplos de 8 bytes (no en bytes absolutos).

**10. Si en el ejercicio anterior el datagrama atravesara otra red de MTU = 200 B, escriba los headers de todos los fragmentos.**

MTU = 200B → payload = 180B.

Se fragmenta el datagrama con payload de 980B en numeros multiplo de 8 y de valor maximo 180 →  $176*5 + 100$

Flags: MF = todos en 1 menos el ultimo.

DF = 0.

Fragment Offset:

Datagrama 1 (176B) = 0

Datagrama 2 (176B) =  $176/8 = 22$

Datagrama 3 (176B) =  $22+22= 44$

Datagrama 4 (176B) =  $22+22+22=66$  Datagrama 5 (176B) = 88 Datagrama 6 (100B) = 110

**11. En el siguiente esquema de red, el host A envía un ping al host B pero el mismo se encuentra apagado. Suponiendo que todos los routers y hosts tienen activados sus protocolos ICMP sin filtrar, indicar si el ping se realiza con éxito o qué sucede en caso contrario.**

El ping no se realiza con éxito porque el host B está apagado por hipótesis. Entonces, según vimos arriba, el router “B” podría enviar un mensaje ICMP tipo 3 (destination unreachable message), código 1 (host unreachable).

## 5.6 ICMP: The Internet Control Message Protocol

The Internet Control Message Protocol (ICMP), specified in [RFC 792], is used by hosts and routers to communicate network-layer information to each other. The most typical use of ICMP is for error reporting. For example, when running an HTTP session, you may have encountered an error message such as “Destination network unreachable.” This message had its origins in ICMP. At some point, an IP router was unable to find a path to the host specified in your HTTP request. That router created and sent an ICMP message to your host indicating the error.

ICMP is often considered part of IP, but architecturally it lies just above IP, as ICMP messages are carried inside IP datagrams. That is, ICMP messages are carried

Figura 124: Texto extraído de Kurose.

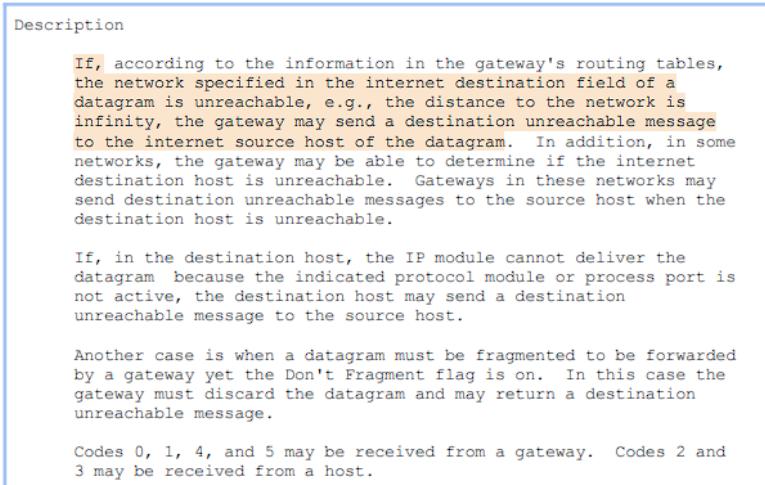


Figura 125: La la RFC que describe el mensaje del protocolo ICMP.

**12. Un host A envía un ping a un host B y configura el TTL del datagrama IP que encapsula el mensaje ICMP Echo Request en un valor igual a 200. Entre el host A y el host B existe un único camino comprendido por 5 saltos, sin embargo, el TTL del datagrama IP que encapsula el mensaje ICMP Echo Reply generado por el host B hacia el host A tiene un valor igual a 15. A qué se debe?**

Esto se debe a que los sistemas operativos pueden usar distintos valores de TTL por defecto al enviar un paquete. El valor de TTL = 15 para el Echo Reply fue asignado por el host B, y no tiene relación directa con el TTL del mensaje original (Echo Request).

**13. Dado el siguiente datagrama IP (Los dígitos son hexadecimales):**

```
45 00 00 2c 06 00 40 00
20 06 28 4f 82 39 14 0a
82 39 14 01 04 02 02 0c
00 00 32 98 00 00 00 00
60 02 20 00 13 03 00 00
02 04 05 b4
```

**a) Indique las direcciones IP origen y destino en su formato correspondiente.**

82 39 14 0a = 130.57.20.10 IP origen

82 39 14 01 = 130.57.20.1 IP destino

**b) ¿Si la red destino tiene definido un MTU = 20 bytes, el datagrama será fragmentado?**

00 2c = 44 total length → 44 - 20 header = 24 datos (payload)

Como MTU = 20, se debe de fragmentar.

**c) ¿Cuántas redes podrá atravesar antes de llegar a destino?**

TTL: 20 (hexa) = 32 (decimal) → Por lo que puede dar 32 saltos antes de llegar al destino o descartarse.

**d) ¿Tiene opciones? Justifique.**

Como header length = 5, se sabe que el header IP tendrá 5 palabras de 4 bytes (osea 4 bytes por linea) = 20 bytes totales. El resto tiene que ser el payload.

Si header length > 5, entonces el encabezado incluye opciones IP. El payload comienza después de ese encabezado extendido.

**e) ¿El campo de datos contiene protocolo auxiliar o de transporte?**

El campo Protocol del datagrama IP es 06 (hexa) → TCP

Protocol	
1	ICMP (Internet Control Message Protocol)
6	TCP (Transmission Control Protocol)
17	UDP (User Datagram Protocol)

**14. Usted obtuvo la dirección IP 205.25.67.0. Determinó que necesita crear 5 subredes. Complete la siguiente tabla teniendo en cuenta que las subredes “todo 0” y “todo 1” no se pueden utilizar:**

Número mínimo de bits para las subredes	[valor]
Máscara	[valor]
Primera dirección de host de la quinta subred	[valor]
Dirección de Broadcast de la tercera subred	[valor]
Número total de direcciones que no se utilizan	[valor]

205 . 25 . 67 . 0  
 5 subredes → 3 bits mínimo →

000	no usar
001	1ra subred
010	2da subred
011	3ra subred
100	4ta subred
101	5ta subred
110	---
111	no usar

máscara : 255.255.255.224/27  
 (1111111. 1111111. 1111111. 11100000)

Host 5ta subred:  
 205 . 25 . 67 . 160 (subred)  
 205 . 25 . 67 . 161 (host)

Broadcast 3ra subred:  
 205 . 25 . 67 . 96 (subred)  
 205 . 25 . 67 . 127 (broadcast {5últimos bits en 1})

Direcciones que no se usan: 106  
 32 de la subred 000  
 32 de la subred 1111  
 32 de la subred 110  
 2 x 5 = 10 (la dir para identificar a la red y broadcast del host)

15. Dada la siguiente tabla de ruteo basada en clases, reescríbala utilizando CIDR intentando minimizar lo más posible el número de entradas:

212.128.175.0	15.0.0.1
212.128.176.0	15.0.0.1
212.128.177.0	15.0.0.1
212.128.178.0	15.0.0.1
212.128.179.0	15.0.0.1
212.128.180.0	15.0.0.1
212.128.181.0	15.0.0.1
212.128.182.0	15.0.0.1
212.128.183.0	15.0.0.1
212.128.184.0	15.0.0.1
212.128.185.0	15.0.0.1

$$\begin{aligned}
 175 &= 128 + 32 + \quad + 8 + 4 + 2 + = 1010 \text{ 1111} \\
 176 &= 128 + 32 + 16 \quad \quad \quad = 1011 \text{ 0000} \\
 \dots \\
 183 &= 128 + 32 + 16 \quad + 4 + 2 + 1 = 1011 \text{ 0111} \\
 184 &= 128 + 32 + 16 + 8 \quad \quad \quad = 1011 \text{ 1000} \\
 185 &= 128 + 32 + 16 + 8 \quad + 1 = 1011 \text{ 1001}
 \end{aligned}$$

Destino	Next-hop o gateway
212.128.175.0/24	15.0.0.1
212.128.176.0/21	15.0.0.1
212.128.184.0/23	15.0.0.1

16. Resolver: En la tabla de ruteo siguiente indique la entrada que tendrán correspondencia con la dirección destino 128.9.200.20. Por qué interfaz será transmitido un datagrama que tenga esa dirección destino. Explicar.

128.0.0.0/8	3
128.9.0.0/16	5
128.9.192.0/20	2
128.9.192.0/22	4
128.9.192.0/24	7
128.9.200.0/24	8
128.9.200.0/28	10
128.9.200.16/28	1
128.9.200.16/30	9
128.9.192.8/30	6

Siguiendo la premisa de CIDR, debemos evaluar las máscaras más largas (selectivas) primero:

interfaz 6 → No puede ser porque desde 192 no llegas a 200 con 2 bits

interfaz 9 →  $0001\ 00/11 = 19$  (último Byte) —→ No llega a 20

interfaz 1 →  $0001/0100 = 20$  —→ Debería de enviar el datagrama a esta interfaz porque es la coincidente con prefijo más grande.

**17. Suponer que la siguiente secuencia de bytes, en hexadecimal, es una cabecera IP. Y responder a las siguientes cuestiones:**

45 00 00 4E C3 2A 00 00 80 11 17 44 82 CE AA 94 82 CE AF FF

a) ¿Cuál es la longitud del datagrama?

Longitud del datagrama = Total Length = 004E = 78bytes.

b) ¿Se trata de un fragmento?

MF\* = 0 y Fragment Offset = 0 → No hay

MF = More Fragment

c) ¿A qué hosts (IP) va dirigido?

IP destino = 82 CE AF FF = 130.206.175.255

d) ¿Qué host lo envió?

IP origen = 82 CE AA 94 = 130. 206.170.148

e) Discutir si están o no en la misma red.

170 = 10101010

175 = 10101111

→ 32-11= 21 si el prefijo es /21 como maximo entonces estan en la misma red

**18. Para las siguientes direcciones de hosts y máscaras de subred encuentre la subred a la que pertenece cada host, la dirección de broadcast de cada subred y el rango de direcciones de hosts para cada subred:**

a) 10.14.87.60/19

Subred	10.14.64.0
Broadcast	10.14.95.255
Rango de Hosts	10.14.64.1 -10.14.95.254

Subred: 87 = 010/10111 → 010/00000 = 64

Broadcast: 010/11111.11111111 = 95.255

Rango de Hosts: 32-19 = 13 →  $2^{13} = 8192-2=8190$  direcciones de hosts

1er host: 010/00000.00000001=64.1 - Ultimo host: 010/11111.11111110=95.254

b) 172.25.0.235/27

Subred	172.25.0.224
Broadcast	172.25.0.255
Rango de Hosts	172.25.0.225 - 172.25.0.254

Subred:  $235 = 111/01011 = 224$

Broadcast:  $111/11111 = 255$

Rango de hosts: 1er host:  $111/00001 = 225$  - ultimo host:  $111/11110 = 254$

**c) 172.25.16.37/25**

Subred	172.25.16.0
Broadcast	172.25.16.127
Rango de Hosts	172.25.16.1 - 172.25.16.126

Subred:  $37 = 0/0100101 \rightarrow 0/0000000=0$

Boradcast:  $0/1111111 = 127$

**19. Se desea configurar una interfaz con la dirección 192.168.13.175 con una máscara de 255.255.255.240, hay algún problema?**

4to byte =  $1111/0000 = 240 \rightarrow /28$

4to byte =  $175 = 1010/1111$

$1010/1111 \rightarrow$  todos los bits de host estan en 1, es la IP de broadcast

subred =  $192.168.13.160/28$

$1010/0000 = 160$

192.168.13.175 es la dirección de broadcast de la subred  $\rightarrow$  No se puede usar dicha dirección.

**20. Dados los siguientes resultados en la ejecución del ping:**

C:\>ping 170.6.2.23

Haciendo ping 170.6.2.23 con 32 bytes de datos:

Respuesta desde 170.6.2.23: bytes=32 tiempo=241.4 ms TTL=249

Respuesta desde 170.6.2.23: bytes=32 tiempo=244.8 ms TTL=249

Tiempo de espera agotado para esta solicitud.

Respuesta desde 170.6.2.23: bytes=32 tiempo=237.1 ms TTL=249

Estadísticas de ping para 170.6.2.23:

Paquetes: enviados = 4, recibidos = 3, perdidos = 1 <25% perdidos>,

Tiempos aproximados de ida y vuelta en milisegundos:

Mínimo = 237.1 ms, Máximo = 244.8 ms, Media = 241.1 ms

**a) A cuántas redes de distancia estima que podría estar el destino?**

Para los routers TTL inicial típico = 255

Saltos =  $TTL_{inicial} - TTL_{recibido}$

Saltos =  $255 - 249 = 6$

**b) A qué se podría deber el paquete perdido?**

Se cortó la luz en un router. El paquete se fue por un camino muy largo y se excedió el TTL. Se excedió la MTU de alguna red y el header del ping estaba

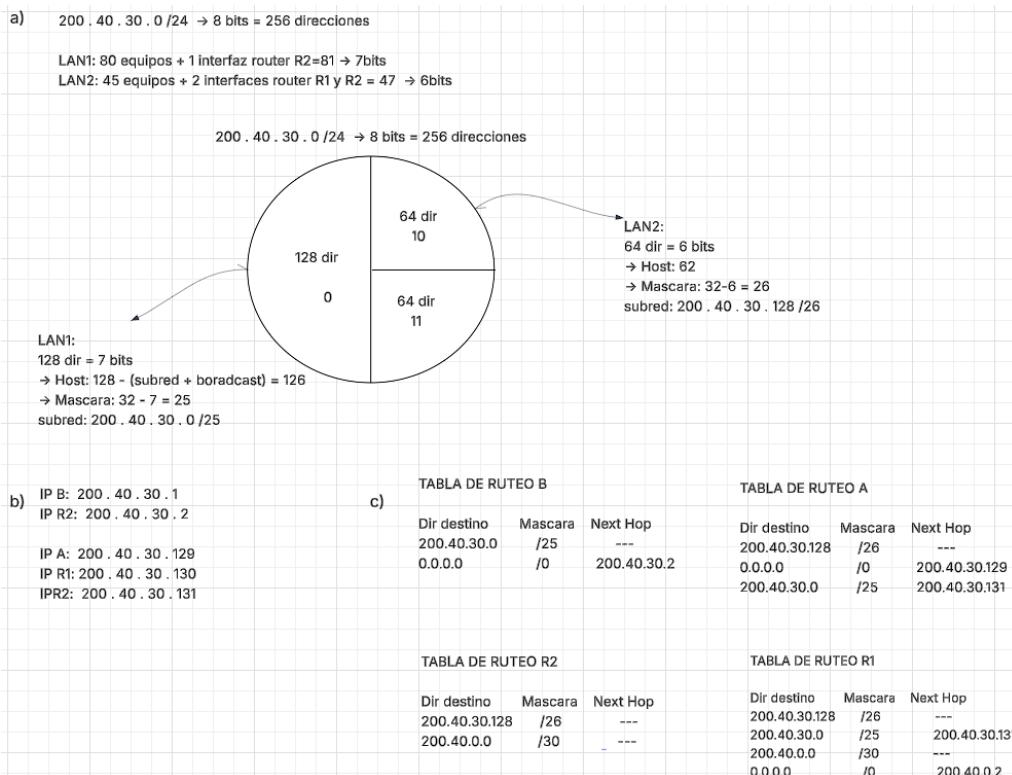
en Don't Fragment.

**21.** Un host A envía a otro host B un datagrama de 8000 bytes (incluida la cabecera IP, que tiene 20 bytes). El datagrama se fragmenta en ruta de forma que B recibe varios datagramas que suman en total 8100 bytes (incluidas las cabeceras). ¿Cuántos fragmentos ha recibido B? Desarrollar la respuesta.

**22.** En el esquema de red de la figura siguiente los equipos A y B son computadoras de usuario mientras que los equipos R1 y R2 son enrutadores. No se utiliza la funcionalidad de NAT (Network Address Translation). En LAN1 se dispone de 80 equipos como B y en LAN2 45 como A.

Se dispone del rango de direcciones 200.40.30.0/24 para asignar a las subredes LAN1 y LAN2. La asignación debe realizarse teniendo en cuenta que no habrá crecimiento de equipos en LAN1 y LAN2.

- Asignar un rango de direcciones a cada LAN.
- Asignar direcciones IP a los equipos A, B, R1 y R2. El enlace de R1 con Internet es un enlace punto a punto configurado con las direcciones indicadas en la figura.
- Especificar las tablas de rutas necesarias en los equipos para que puedan comunicarse entre sí y con cualquier equipo de Internet.



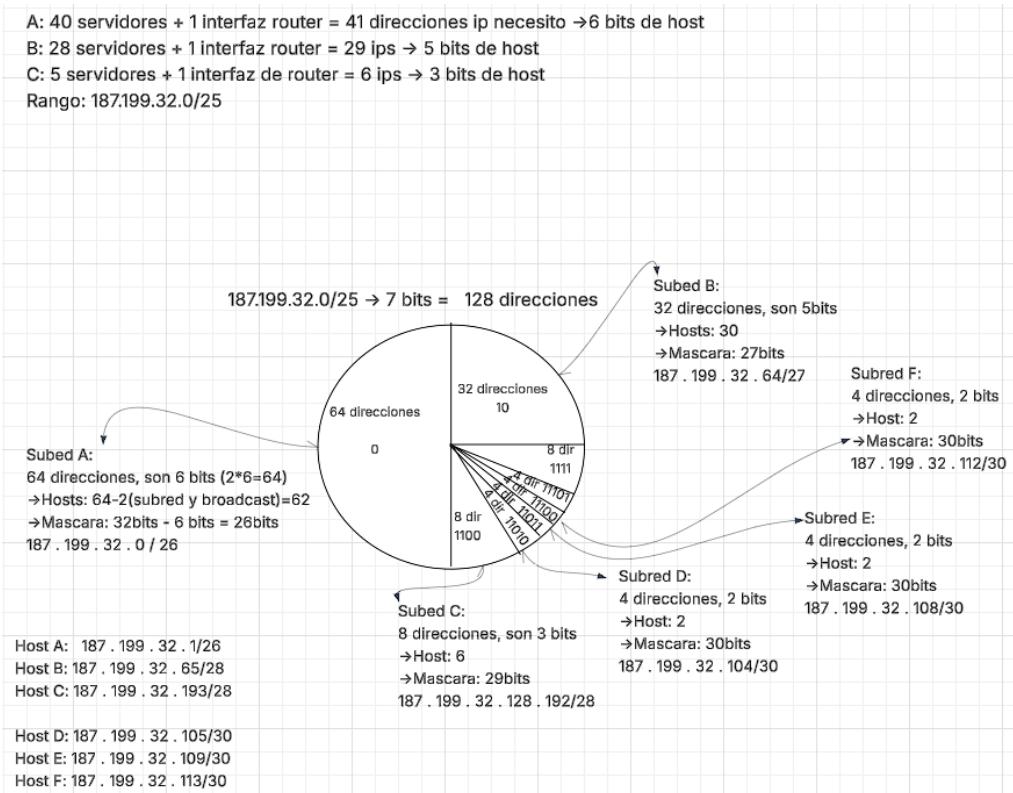
**23.** Una empresa compra el rango de direcciones IP 152.165.10.0/23. Si se quiere dividir la red en dos subredes iguales, ¿cuáles serían (for-

mato a.b.c.d/x) y cuántos equipos podrían direccionarse en cada una de ellas? Indicar también la dirección de broadcast de cada subred.

24. Una gran empresa desea asignar una dirección pública a cada uno de sus servidores y a cada uno de sus routers (R1, R2 y R3). La empresa dispone de 40 servidores en su sede A, 28 servidores en su sede B y 5 servidores en su sede C. Para ello, la empresa adquirió el rango 187.199.32.0/25.

- Realizar la asignación de rangos a las diferentes subredes A, B, C, D, E y F.
- Proponer una IP para cada interfaz de cada router.
- Proponer una IP para un servidor en A, otro en B y otro en C.

A: 40 servidores + 1 interfaz router = 41 direcciones ip necesito → 6 bits de host  
B: 28 servidores + 1 interfaz router = 29 ips → 5 bits de host  
C: 5 servidores + 1 interfaz de router = 6 ips → 3 bits de host  
Rango: 187.199.32.0/25



25. Responda a las siguientes preguntas justificando sus respuestas.

- Dada la dirección IP 192.168.1.1 con máscara de 24 bits en 1 (/24), indicar dirección de red que identifica el bloque, dirección de broadcast y máscara de red en notación decimal separada por puntos.
- Dada la dirección IP 10.1.1.35 y máscara 255.255.255.248, indicar el rango de direcciones IP que pertenecen a su LAN.
- ¿Es posible sumarizar en un solo rango los siguientes bloques de direcciones IP:  
 200.40.0.0/20, 200.40.16.0/21, 200.40.24.0/21, 200.40.32.0/19, 200.40.64.0/18  
 y 200.40.128.0/17, en un solo rango con notación A.B.C.D/M? En

caso afirmativo, hallar A, B, C, D y M.

26. Suponer que usted es el administrador de red de un ISP y se dispone del bloque de direcciones: 128.20.224.0/20. El ISP tiene dos clientes con redes de 1.000 nodos cada una; dos clientes con redes de 500 nodos cada una y tres clientes con redes de 250 nodos cada una. ¿Cuáles serán los bloques de direcciones que usted asignará a los clientes? Suponer que todos los clientes restantes tienen redes de 50 nodos cada una. ¿A cuántos de esos clientes podrá asignarle direcciones?

27. En la figura se muestra una red. Al lado de cada máquina aparece su tabla de ruteo. Al lado de cada interfaz de comunicaciones aparece su dirección IP (salvo en el caso de la máquina B).

a) Asignar una dirección IP a la interfaz de comunicaciones de la máquina B.

b) Responder:

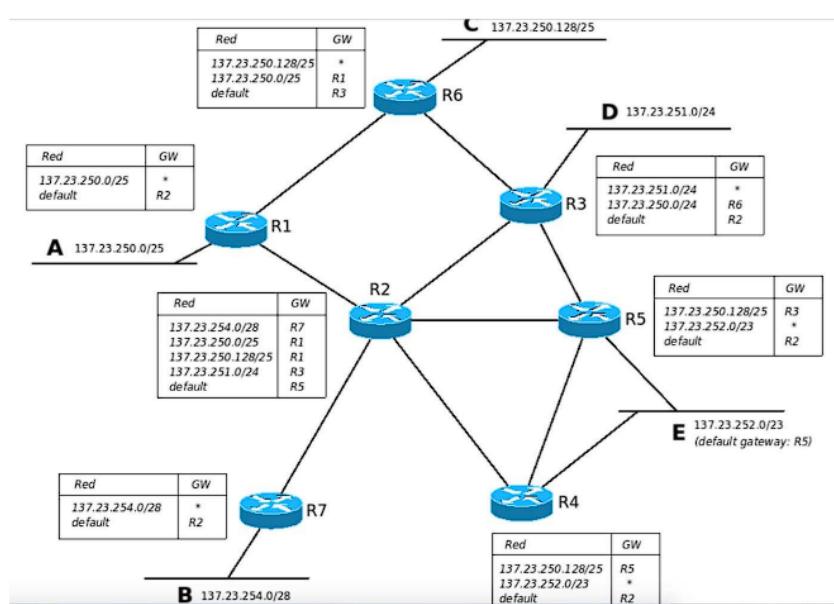
- 1) ¿Puede A enviar datagramas IP a C?
- 2) ¿Puede D enviar datagramas IP a C?

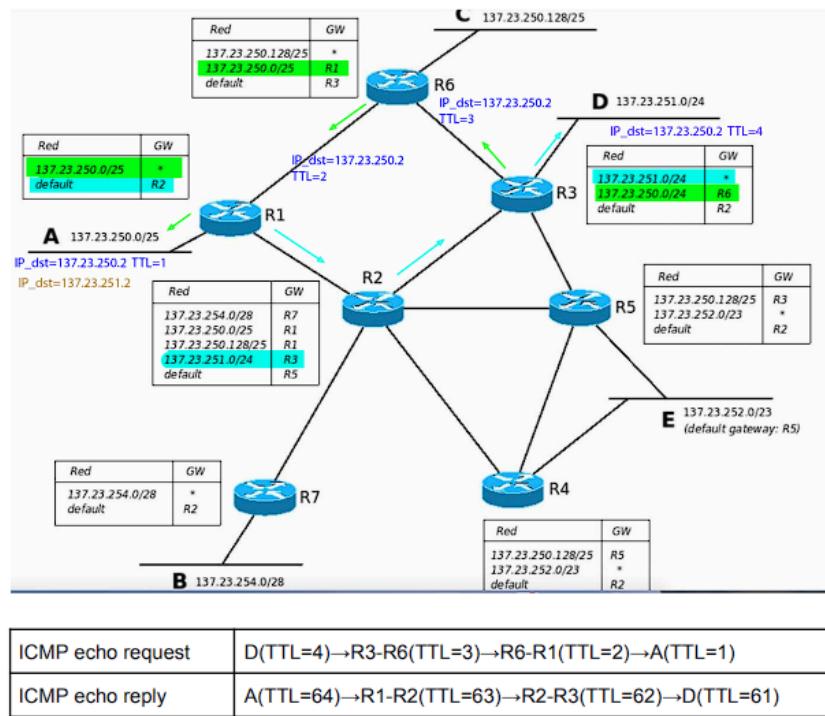
c) Modificar la tabla de ruteo de R2 para que:

D pueda enviar datagramas IP a A

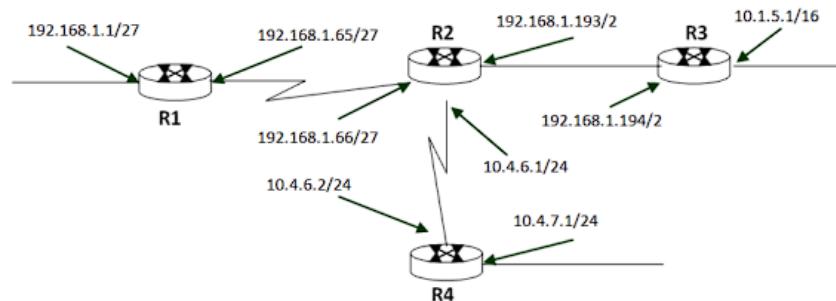
C pueda enviar datagramas IP a A

28. Sea la red de la figura. Si un host en D hace un ping a un host en A configurando el ping con un solo mensaje y TTL = 4 indique todos los datagramas que se generarán como producto de ese ping y sus recorridos. Indicar los campos relevantes de cada uno de ellos.





29. Para el diagrama indicado, configure las rutas en cada uno de los routers de manera de obtener conectividad completa en la red:



30. Dada la siguiente tabla de ruteo de un host, resuelva:

Rutas activas:

Destino de red	Máscara de red	Siguiente Salto	Interfaz	Métrica
0.0.0.0	0.0.0.0	10.6.2.241	10.6.2.52	20
10.6.0.0	255.255.0.0		10.6.2.52	20
10.6.2.52	255.255.255.255		127.0.0.1	20
10.255.255.255	255.255.255.255		10.6.2.52	20
127.0.0.0	255.0.0.0		127.0.0.1	1
224.0.0.0	240.0.0.0		10.6.2.52	20

a) Un datagrama enviado por dicho host cuya dirección destino es 145.57.2.98, por qué interfaz sale y cuál es el siguiente salto para dicho datagrama?

- b) Un datagrama enviado por dicho host cuya dirección destino es 127.0.3.4, por qué interfaz sale y cuál es el siguiente salto para dicho datagrama?
- c) Si se elimina la primer entrada de la tabla de ruteo y se envía un datagrama por dicho host cuya dirección destino es 201.11.3.124, por qué interfaz sale y cuál es el siguiente salto para dicho datagrama?

31. Dadas las siguientes redes:

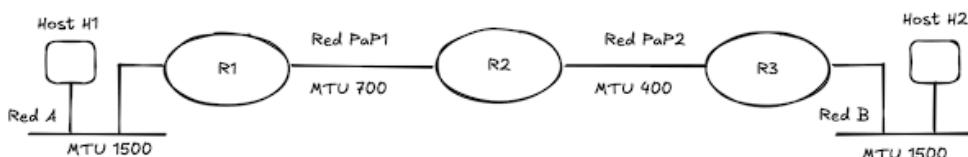
- a) 200.2.4.0/24
- b) 200.5.6.0/24
- c) 200.67.5.0/24
- d) 200.34.5.0/24
- f) 200.96.12.0/24
- g) 200.96.34.0/24
- h) 200.96.200.0/24

Grafe una red tal que permita interconectar a todas ellas. Luego escriba la tabla de ruteo en cada uno de los routers utilizados considerando que se trabaja con ruteo classless.

32. Conteste a las siguientes preguntas relativas a las direcciones IP:

- a) Indique la dirección de red correspondiente a la dirección IP 192.168.100.115 considerando una máscara 255.255.255.240.
- b) Usted está diseñando una red a partir de una dirección clase A. Desea poder asignar 16.000 hosts en cada subred. Determine la máscara a utilizar para satisfacer ese requisito.

33. En el siguiente diagrama de redes, el host H1 envía un ICMP Echo Request al host H2 con una longitud total del datagrama IP de 1500 Bytes.



- a) Indique en cada red si el datagrama original se fragmentó, en caso afirmativo describa en una tabla los campos relevantes al proceso de fragmentación. En el caso de hacer suposiciones explíctelas.
- b) ¿Quién reensambla? ¿Por qué?
- c) ¿Qué pasa con la respuesta al ICMP Echo Request? ¿Cuántos fragmentos llegan a H1?
- d) Suponga que se hubiera perdido un fragmento de ICMP Echo Request, ¿qué hubiera pasado?

## 5. Bibliografía

### Referencias

- [1] Kurose, J. F., & Ross, K. W. (2016). *Computer Networking: A Top-Down Approach* (7th ed.). Pearson.

## 6. Apéndice