GENERALITAT VALENCIANA
Conselleria d'Educació, Investigació, Cultura i Esport

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

1º DAM

Sistemas Informáticos

UD9. Arquitecturas y Protocolos de Red

Juan Bautista Valero Carrasco
jb.valerocarrasco@edu.gva.es

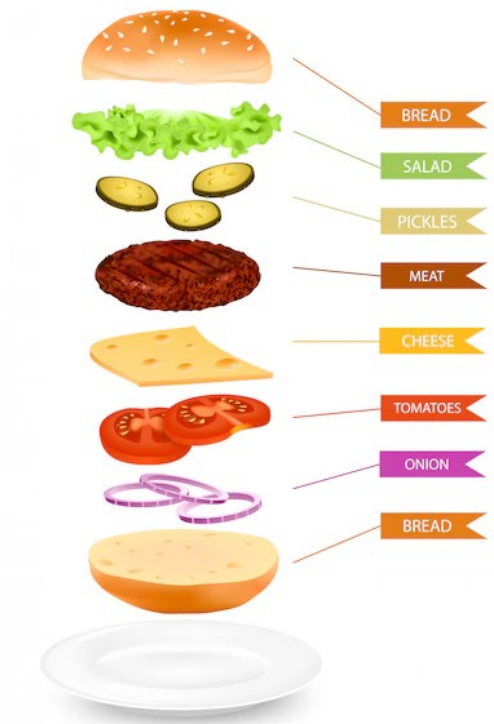
IES Marcos Zaragoza

CONTENIDOS

1. Arquitecturas de redes
   - 1.1 Modelo OSI
   - 1.2 Modelo TCP/IP
2. Protocolos TCP/IP
   - 2.1 Protocolos capa aplicación
   - 2.2 Protocolo capa transporte
   - 2.3 Protocolo capa internet
3. Protocolo IP
   - 3.1 Direccionamiento IP
   - 3.2 Segmentación de redes
   - 3.3 IPv6
3. Protocolo TCP/UDP
   - 3.1 Puertos
   - 3.2 TCP
   - 3.3 UDP

1. ARQUITECTURAS DE REDES

- La **arquitectura de una red** es un concepto amplio que incluye cuestiones relacionadas con el **hardware** y con el **software** de una red.
- Uno de los problemas más importantes a la hora de **diseñar una red** corresponde a la **complejidad** que acarrea considerar la red como un todo. Así, se consideró oportuno **organizar las redes** como una serie de **capas**, donde **cada capa** se ocuparía de alguna **función específica**. De esta forma se reduciría la complejidad del diseño de la red y de las aplicaciones que en ella se utilicen.



1. ARQUITECTURAS DE REDES

- Por tanto podemos **redefinir arquitectura de red:**
  - **Conjunto de capas o niveles**, **junto con los protocolos** definidos en **cada** una de estas **capas**, que hacen posible que un ordenador se comunique con otro ordenador independientemente de la red en la que se encuentre.
- De todo esto podemos concluir que la **arquitectura de red** tendrá que tener en cuenta al menos **tres factores importantes** como son:
  - La forma como se conectan los nodos de una red, que suele conocerse como **topología**, además de las características físicas de estas conexiones.
  - La manera de como compartir información en la red, que en algunos casos obligará a elegir un **método de acceso a la red** y unas reglas para evitar perdida de información.
  - Unas reglas generales que no sólo favorezcan la comunicación, si no que la establezcan, mantengan y permitan la utilización de la información, estás reglas serán los **protocolos de comunicación**.

1. ARQUITECTURAS DE REDES

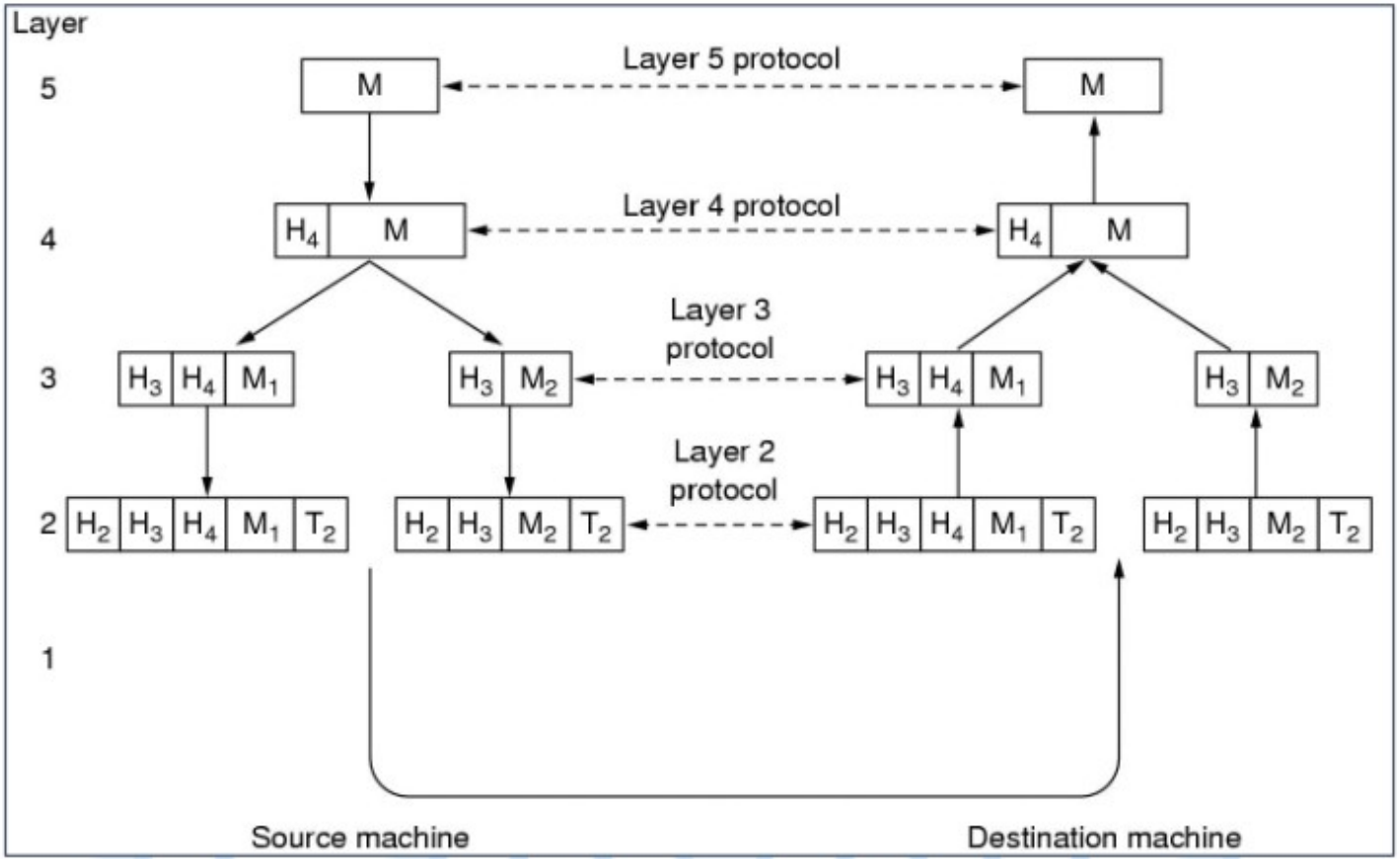
- La **organización** de una **arquitectura en capas** asume ciertas **características**:
  - Las **funciones individuales** a realizar son **más simples**, lo que facilita la implantación de cada nivel (por separado) en la práctica.
  - Para **cada nivel** se **define un servicio**, construido **sobre** los **servicios de niveles inferiores** y que mejora los anteriores gracias a la funcionalidad aportada por el protocolo de ese nivel.
  - Los **protocolos** se establecen entre **niveles gemelos (*peer protocols*),** y cada uno de ellos puede optimizarse
  - Como los **niveles son independientes**, es posible sustituir el protocolo de un determinado nivel por otro, siempre que el servicio ofrecido por dicho nivel no varíe.
- La **capa n** de una máquina lleva a cabo una **conversación** con la **capa n** de otra. Las **reglas y convenciones** que se siguen en esta conversación se conocen colectivamente como **protocolo de la capa n**.

1. ARQUITECTURAS DE REDES

- **Funcionamiento de las arquitecturas** basadas en niveles o **capas**.
  - Las **entidades** que comprenden las capas correspondientes en las diferentes máquinas se denominan **pares**, y son las entidades pares las que **se comunican usando el protocolo** correspondiente.
  - Los **datos no se transfieren directamente** de la **capa n** de una máquina **a la capa n de otra**. Cada capa pasa datos e información de control a la capa inferior, hasta llegar a la capa más baja. Bajo la capa 1 está el medio físico a través del cual ocurre la comunicación real.
  - En la figura siguiente, se muestra en líneas punteadas la comunicación virtual y en líneas continuas la comunicación física.entre

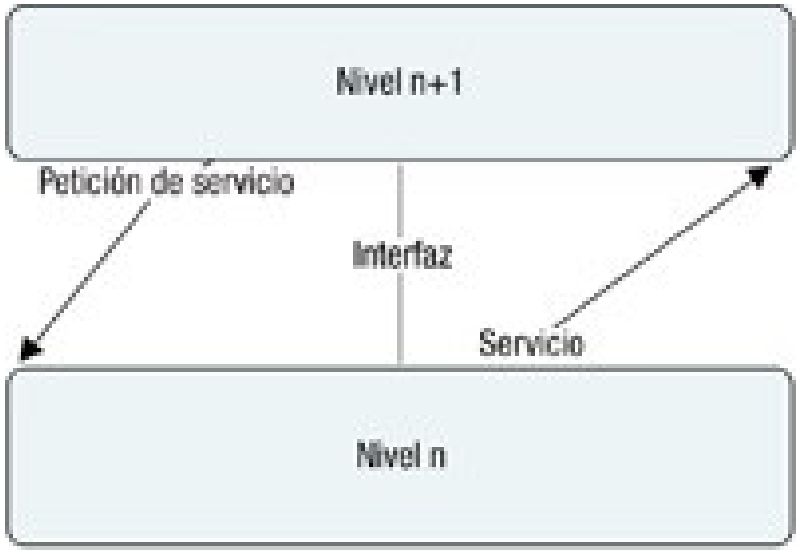
1. ARQUITECTURAS DE REDES

- **Funcionamiento de las arquitecturas** basadas en niveles o **capas**.



1. ARQUITECTURAS DE REDES

- **Funcionamiento de las arquitecturas** basadas en niveles o **capas**.
  - **Entre cada par de capas adyacentes** hay una **interfaz**. La interfaz define qué **operaciones y servicios** primitivos **ofrece la capa inferior** a la superior.



1. ARQUITECTURAS DE REDES

- **Funcionamiento de las arquitecturas** basadas en niveles o **capas**.
  - El intercambio de información entre dos capas consiste en que **cada capa** en el **sistema fuente** le **agrega información de control a los** datos, y **cada capa en el sistema de destino analiza y quita la información de control** de los datos.
    - Si una computadora (A) desea enviar datos a otra (B), en primer término los datos deben empaquetarse a través de un proceso denominado **encapsulamiento**, es decir, a medida que los datos se desplazan a través de las capas, reciben **encabezados, información final y otros tipos de información**.
    - De está manera la información llegará a su destino y cada nivel sólo se ocupará de los datos y la información de control que necesite, según el protocolo utilizado, sin preocuparse de lo que hagan o necesiten los otros niveles.

1. ARQUITECTURAS DE REDES

1.1 Modelo OSI

- Las **primeras redes** de ordenadores utilizaban **protocolos específicos de cada fabricante** que las hacían **incompatibles entre ellas**.

ISO International Organization for Standardization

- Como esta situación suponía un problema tanto para los usuarios como para los propios fabricantes, en 1977, la **Organización Internacional de Normalización** (ISO, *International Standard Office*) comenzó el desarrollo de un **estándar abierto que facilitara la interoperabilidad** entre las **redes** de diferentes fabricantes.

OSI Open Systems Interconnection

1. ARQUITECTURAS DE REDES

1.1 Modelo OSI

- Para **su desarrollo se basó en varios modelos existentes**, como **DECnet** de DEC (*Digital Equipment Corporation*), **SNA** (*Systems Network Architecture)* de IBM, o **TCP/IP** desarrollado por DARPA (*Defense Advanced Research Projects Agency*) para la red ARPANET.
- El estándar se publicó en **1984** y fue bautizado como **modelo OSI (*Open Systems Interconnection*)**. Aún así, a partir de 1985, el **modelo TCP/IP** comenzó a **ganar protagonismo**, siendo el que se impuso finalmente.
- En realidad, el **modelo OSI no se ha implementado** en ningún sistema, pero es fundamental entenderlo porque se utiliza a en el **ámbito académico y de investigación** como referencia para compararlo con otros modelos.

1. ARQUITECTURAS DE REDES
1.1 Modelo OSI

- **Estructura del modelo OSI**
  - Así pues, el **modelo de referencia OSI se divide en siete capas**:

1. ARQUITECTURAS DE REDES
1.1 Modelo OSI

Estructura del modelo OSI

- **Aplicación**: Ofrece la interfaz entre el software de aplicación y la red, implementando los servicios que serán invocados por éste.
  - Aquí se definen los protocolos que usan los programas para intercambiar datos, como FTP, HTTP, POP, SMTP, Telnet, etc.
- **Presentación**: Se encarga de ofrecer los datos a la capa de Aplicación. Por lo tanto, se encarga de traducir y codificar los datos obtenidos de la aplicación a un formato genérico antes de transmitirlo por la red. A la inversa, los datos que recibe son convertidos de un formato genérico al formato que espera la capa de Aplicación.
  - De esta capa depende, por ejemplo, que puedan comunicarse sistemas que utilizan diferentes juegos de caracteres (como EBCDIC o ASCII). También se encarga de comprimir/descomprimir y de cifrar/descifrar la comunicación, cuando es necesario

1. ARQUITECTURAS DE REDES
1.1 Modelo OSI

Estructura del modelo OSI

- **Sesión**: En un equipo puede haber varias aplicaciones transmitiendo datos a través de la red, incluso varias instancias de la misma aplicación. Para cada una de esas instancias, se creará una sesión diferente, y será la capa de Sesión quien se encargue de hacerle llegar los datos que le correspondan.
  - Será en esta capa donde se creen, administren y destruyan las sesiones. Además, controla y coordina la comunicación entre sistemas
- **Transporte**: Se centra en la transferencia de los datos de extremo a extremo de la red, encargándose de establecer una conexión lógica entre los dos equipos implicados e independizándolos de la red física que se esté usando.
  - Proporciona la entrega de datos (que puede ser fiable, o no) y ofrece control de flujo y recuperación de errores.

1. ARQUITECTURAS DE REDES
1.1 Modelo OSI

Estructura del modelo OSI

- **Red**: Aquí se identifica el origen y el destino mediante direcciones lógicas únicas en la red. Si ambos participantes en la comunicación se encuentran en redes distintas, éstas deben encontrarse unidas *routers*, que elegirán la mejor ruta posible para entreg (determinación de ruta).

  Por lo tanto, los *routers* utilizan la capa de red para realizar su trabajo (direccionamiento lógico, determinación de la ruta y reenvío). También lo hacen algunos *firewalls* para descargar paquetes en función de su origen.

  - Un dato importante es que los routers sólo se encargan de hacer llegar los paquetes a la red de destino. Una vez allí, el receptor concreto se identificará en la capa de enlace de datos.

1. ARQUITECTURAS DE REDES
1.1 Modelo OSI

Estructura del modelo OSI

- **Enlace de datos**: Todos los ordenadores de una red disponen de una dirección física y otra lógica. La dirección física sólo es significativa dentro de la red local. Es el nivel de enlace de datos el que centra su atención en la transferencia de datos dentro de la red local usando direcciones físicas.

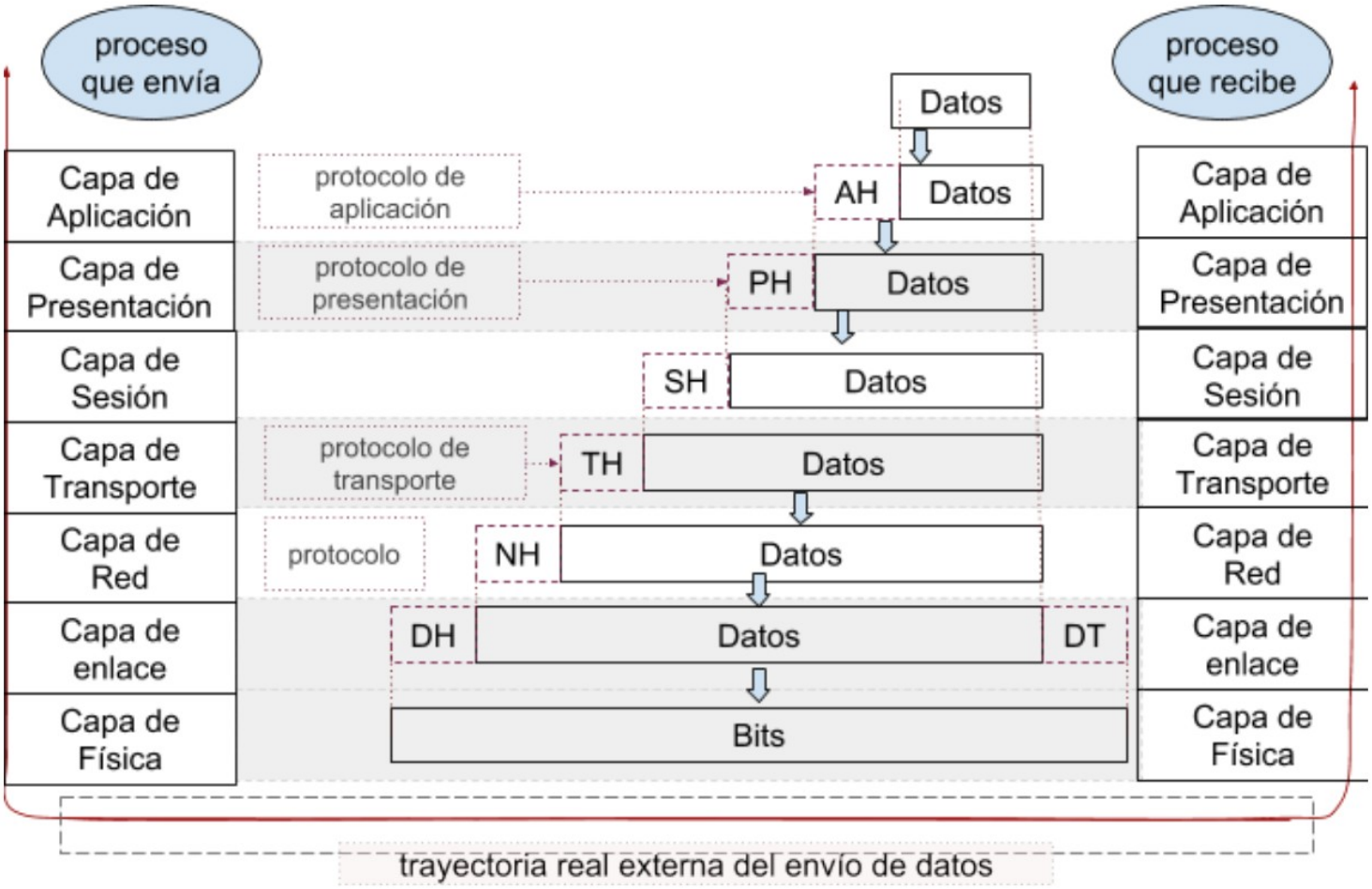
  Cuando en un medio pueden coincidir varios equipos tratando de enviar paquetes al mismo tiempo, para evitar colisiones, se debe asegurar que sólo uno de ellos lo utiliza cada vez. En esta capa se establecen los protocolos usados para el envío y recepción de datos a través de los medios. Estos deberán determinar si el medio está disponible y detectarán las colisiones y los posibles errores en los datos recibidos.

- **Física**: En esta capa es donde se gestionan los dispositivos físicos encargados de la comunicación. En ella se envían o reciben los bits que forman el mensaje sin interpretar su significado y será la encargada de iniciar, mantener y finalizar la comunicación entre sistemas (por ejemplo, entre un equipo y un switch).

  De esta capa dependen las transformaciones que se realizan en los bits, que dentro del ordenador están representados por diferentes niveles de electricidad, para adaptarlos al medio de transmisión utilizado (óptico, infrarrojo, láser, radio, microondas, etc.).

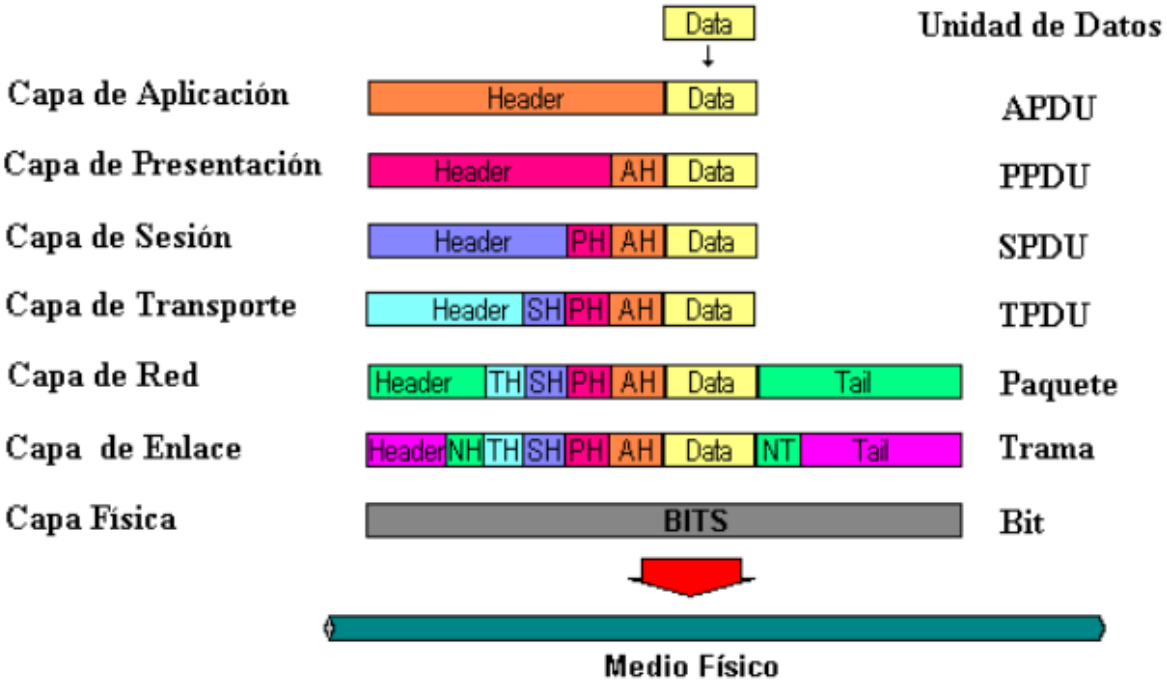
1. ARQUITECTURAS DE REDES
1.1 Modelo OSI

- **Funcionamiento del modelo OSI**



1. ARQUITECTURAS DE REDES
1.1 Modelo OSI

- **Funcionamiento del modelo OSI**
  - Los **datos** reciben una serie de **nombres y formatos específicos** en función de la **capa** en la que se encuentren, debido a la adhesión de una serie de encabezados e información final. Los formatos de información son:



APDU
(capa 7) unidad de datos en la capa de aplicación.

PPDU
(capa 6) unidad de datos en la capa de presentación.

SPDU
(capa 5) unidad de datos en la capa de sesión.

TPDU
(capa 4) unidad de datos en el Segmento de transporte o Datagrama.

Paquete
(capa 3) unidad de datos en el capa de red.

Trama
(capa 2) unidad de datos en la capa de enlace de datos.

Bit
(capa 1) unidad de datos en la capa física.

1. ARQUITECTURAS DE REDES

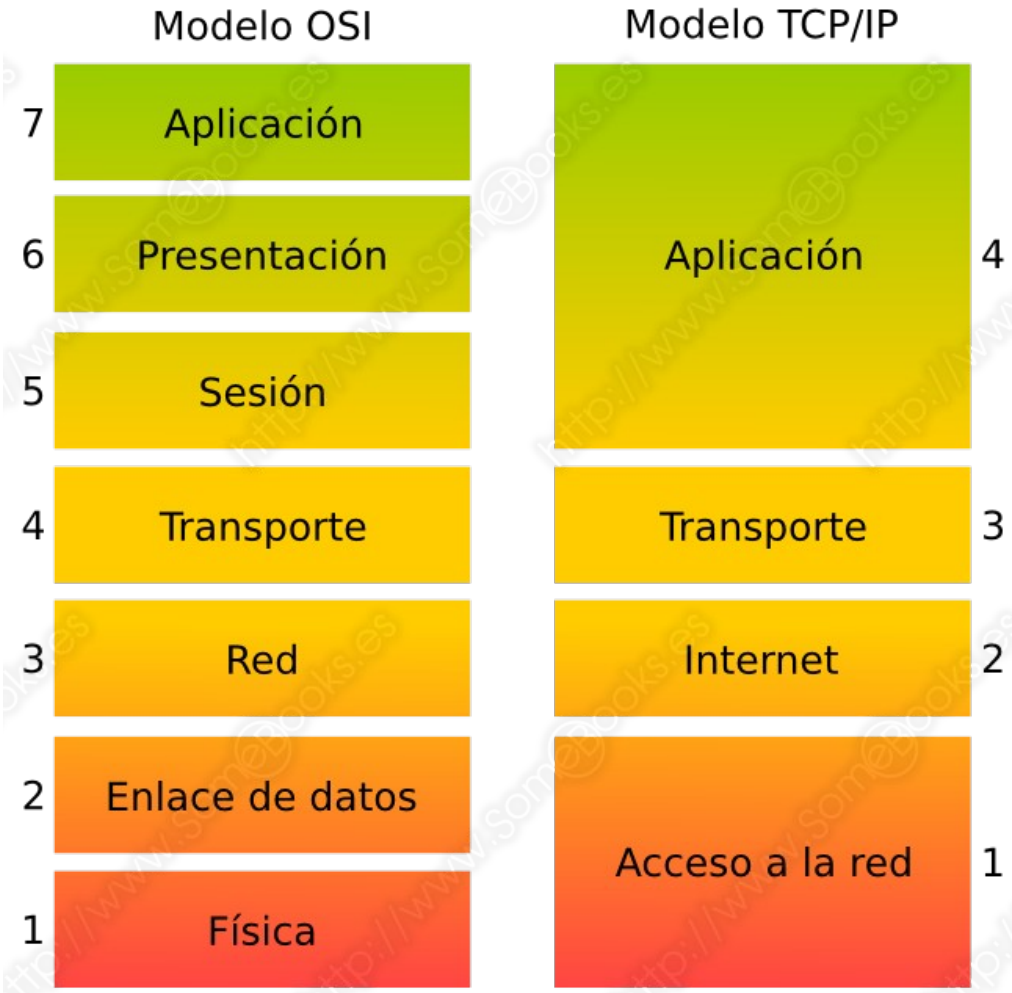
1.1 Modelo TCP/IP

- El **modelo TCP/IP** define una **arquitectura de red** formada por **4 capas**.

	Modelo TCP/IP	Protocolos	Direccionamiento
4	Aplicación	Telnet HTTP FTP SMTP TFTP DNS DHCP ...	Direccionamiento lógico (URL)
3	Transporte	TCP UDP	Puerto
2	Internet	IP, ARP, ICMP IGMP, RARP	Dirección IP
1	Acceso a la red	Ethernet PPP Frame Relay ...	Dirección Física

1. ARQUITECTURAS DE REDES
1.1 Modelo TCP/IP

- **Correspondencia entre el modelo OSI y el modelo TCP/IP**
  - A pesar de que el modelo **OSI** dispone de **siete capas** diferentes y el modelo **TCP/IP** sólo tiene **cuatro**, podemos establecer un paralelismo entre ambos modelos teniendo en cuenta su funcionalidad.
  - Esta correspondencia puede verse en la siguiente imagen:



1. ARQUITECTURAS DE REDES
1.1 Modelo OSI

- **Aplicación**: Implementa los diferentes protocolos de servicio incluidos en las capas de Aplicación (7), Presentación (6) y Sesión (5) incorporados en el modelo OSI. Así, esta capa incluye la comunicación con el software de aplicación, la traducción y la codificación de los datos, y el diálogo entre sistemas.
  - Algunos de los protocolos de aplicación que se implementan en esta capa son: Telnet, HTTP, FTP, SMTP, TFTP, DNS y DHCP, entre otros.
- **Transporte**: Como en el modelo OSI, esta capa administra la conexión lógica entre los equipos que se comunican y el envío de datos entre ellos.
  - En esta capa disponemos de dos protocolos distintos: TCP y UDP.

1. ARQUITECTURAS DE REDES
1.1 Modelo OSI

- **Internet**: Como en la capa de Red del modelo OSI, la capa de Internet identifica el origen y el destino de la transmisión y, cuando se encuentran en redes diferentes, los routers que los unen eligen la mejor ruta posible para entregar cada paquete en su destino. Por lo tanto, los routers utilizan esta capa para realizar su trabajo: direccionamiento lógico, determinación de la ruta y reenvío.
  - En esta capa encontramos cinco protocolos, IP, ARP, ICMP, IGMP y RARP, de los que hablaremos más abajo.

1. ARQUITECTURAS DE REDES
1.1 Modelo OSI

- **Acceso a la red**: Esta capa se corresponde con las capas Enlace de datos y Física del modelo OSI. Contiene las especificaciones relativas a la transferencia de los paquetes recibidos de la capa Internet a través de la red física. El destinatario puede ser otro equipo de la misma red o un router que se encargue de enviar los paquetes a una red distinta. Por lo tanto, a diferencia de la capa Internet, que tiene alcance sobre toda la red, el de la capa de Acceso a la red sólo llega hasta el primer router.
  - En esta capa, se resuelven aspectos como:
    - Enrutamiento de datos a través de la conexión
    - Sincronización de la transmisión
    - Formato de los datos
    - Conversión de señales (analógica/digital)
    - Detección de errores
    - ..
  - Como se puede deducir, en esta capa encontramos un gran número de protocolos, como los protocolos Ethernet, cuando nos encontramos en una red local, o los protocolos PPP (Point-to-Point Protocol) y Frame Relay, cuando nos encontramos en una red WAN.

2. PROTOCOLOS TCP/IP

	OSI	TCP / IP	Protocolos					
7	Aplicación	Aplicación	Telnet	Counter Strike	Naveg. http	DNS	Emule	Telegram
6	Presentación							
5	Sesión							
4	Transporte	Transporte	TCP			UDP		
3	Red	Interred	**IP**					
2	Enlace de Datos	Host a red	Ethernet y Wifi (802.XX)			bluetooth		GSM
1	Físico		cable	wifi				

2. PROTOCOLOS TCP/IP

2.1 Protocolos capa de aplicación

- **Telnet**: Es un protocolo que permite emular una terminal de texto de un equipo remoto. El equipo accedido (servidor) debe ejecutar un servicio, también llamado *Telnet*, que atiende las solicitudes de los equipos remotos (clientes).

  En el sistema operativo cliente, también necesitaremos un programa que establezca la conexión y nos muestre el contenido de la terminal de texto del servidor.

  El sistema operativo del servidor procesará las órdenes recibidas en la terminal remota como si procedieran de los dispositivos de entrada locales.

- **HTTP (*Hypertext Transfer Protocol*)**: Facilita la transferencia de páginas web, con todo su contenido relacionado, entre un equipo servidor y un equipo cliente.

  El programa que utiliza este protocolo en el lado servidor suele denominarse servidor web o servidor http. De igual modo, el programa que realiza las solicitudes y recibe los contenidos en el lado cliente suele llamarse cliente web o cliente http. Es común que la tarea de cliente web la realice un navegador de internet.

2. PROTOCOLOS TCP/IP

2.1 Protocolos capa de aplicación

- **FTP (*File Transfer Protocol*)**: Es un protocolo para la transferencia de archivos entre dos equipos. De nuevo, necesitamos un software específico que actúe como servidor y como cliente en ambos lados de la conexión.

  Con el programa cliente, el usuario tendrá acceso al árbol de directorios (o a una parte de éste) y descargar o subir archivos en él.

- **SMTP (*Simple Mail Transfer Protocol*)**: Permite enviar correos electrónicos, desde un programa que actúa como cliente *smtp* al servidor *smtp* de la entidad que provee el correo al usuario remitente.

  Dicho servidor enviará el mensaje al servidor *smtp* que provee el correo al usuario destinatario, usando también el protocolo SMTP.

  Finalmente, éste último envía el mensaje al destinatario usando los protocolos **POP3** o **IMAP**.

2. PROTOCOLOS TCP/IP

2.1 Protocolos capa de aplicación

- **DNS (*Domain Name System*)**: Se trata de un protocolo que permite atender solicitudes para un servidor *dns*. Las funciones de este tipo de servidores consisten en traducir un nombre de dominio en su dirección IP correspondiente. Así se evita que el usuario deba aprender las direcciones IP de los servidores que visita.

  De este modo, cada vez que escribimos la dirección de una página web en un navegador, éste actúa como cliente de un servidor *dns* que la traduce en su dirección IP correspondiente. Después, el navegador establecerá una sesión HTTP con el servidor de dicha dirección IP.

- **DHCP (*Dynamic Host Configuration Protocol*)**: Mediante este protocolo, un equipo de la red puede obtener su dirección IP, la dirección de un servidor *dns*, una máscara de subred, etc., de forma automática. El equipo que lo proporciona recibe el nombre de servidor *dhcp*.

2. PROTOCOLOS TCP/IP

2.2 Protocolos capa de transporte

- **TCP (*Transmission Control Protocol*)**: Es un protocolo confiable, orientado a la conexión, que utiliza en método de acuse de recibo independiente de los niveles inferiores.

  De este modo, los routers que intervienen en la comunicación, y que sólo actúan a nivel de red, únicamente ven un datagrama y no tienen responsabilidad sobre su contenido. Será TCP el encargado de verificarlos y ordenarlos.

2. PROTOCOLOS TCP/IP

2.2 Protocolos capa de transporte

- **UDP (*User Datagram Protocol*)**: Se trata de un protocolo muy simple, y no confiable, que no incluye detección de errores. Es decir, no está orientado a la conexión.

La cabecera UDP sólo contiene la longitud del paquete y un chequeo de sumas. Esto lo hace más ligero y rápido, y lo convierte en la solución ideal para el transporte de información pesada, como vídeo o audio, siempre que la red sea fiable.

En estos casos, las comprobaciones de confiabilidad se atiende en la capa de aplicación.

2. PROTOCOLOS TCP/IP
2.2 Protocolos capa de transporte

Concepto de puerto

- Tanto **TCP como UDP** utilizan el **concepto** de número de **puerto** para realizar su trabajo.
- Un equipo que se encuentra unido a una red, puede estar ejecutando a la vez varias aplicaciones que envíen y/o reciban datos a través de esta. La forma de **separar el tráfico de cada aplicación** consiste en que éstas **usen puertos diferentes**. Tanto TCP como UDP añaden un número de puerto en el encabezado del tráfico que generan.
- El **puerto concreto** que se utilice **dependerá del protocolo de aplicación** utilizado en la **capa de aplicación**. De hecho, los más comunes disponen de un puerto predeterminado en el intervalo 1 hasta 1024. De este modo, los programas que usen dichos protocolos escucharán sus puertos correspondientes TCP o UDP. Analizando la cabecera, sabrán a quién hacer llegar los paquetes que reciban.

2. PROTOCOLOS TCP/IP
2.2 Protocolos capa de transporte

- **Concepto de puerto**
  - En la siguiente tabla disponemos de una lista con los números de puerto asignados a los protocolos de aplicación más usados:

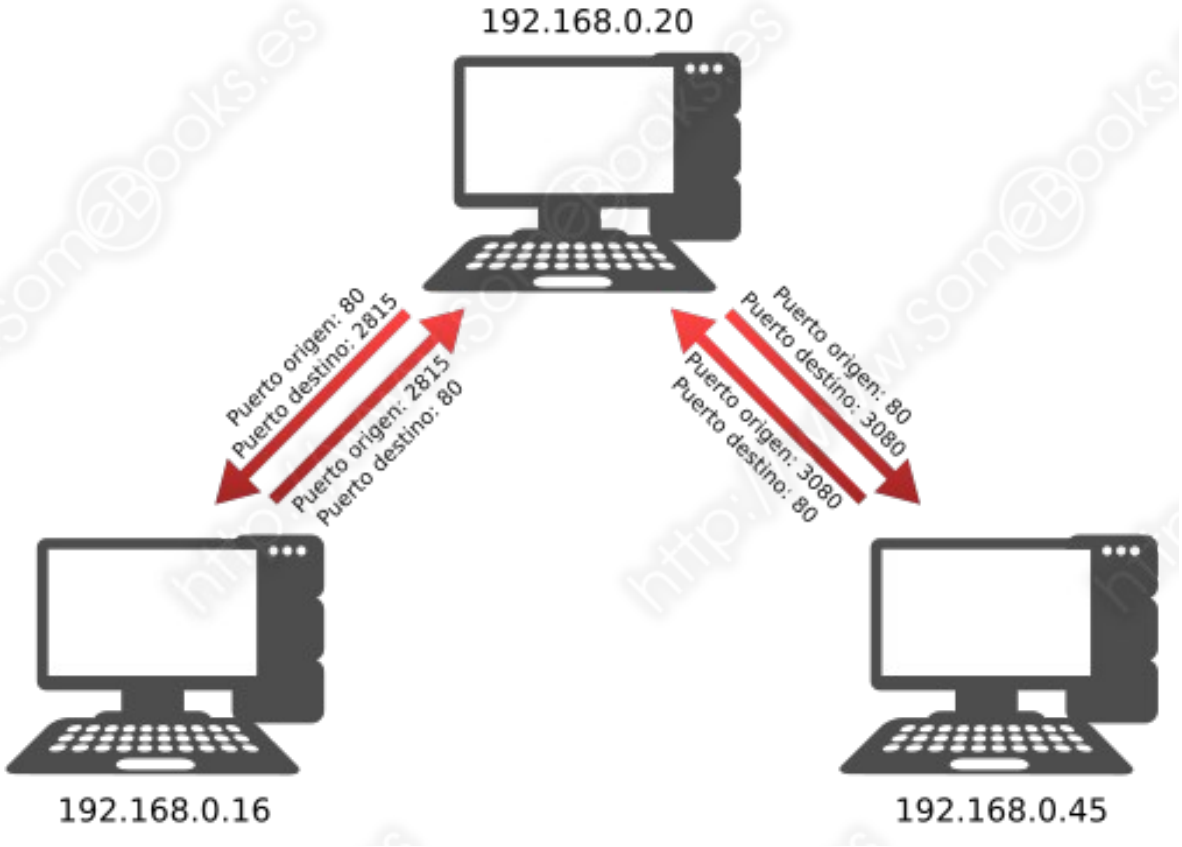
Protocolo de aplicación	Protocolo de transporte	Número de puerto
Telnet	TCP	23
HTTP	TCP	80
HTTPS	TCP	443
FTP (control)	TCP	21
FTP (datos)	TCP	20
SMTP	TCP	25
TFTP	UDP	69
DNS	TCP, UDP	53
DHCP	UDP	67 (server) 68 (cliente)
SSH	TCP	22

2. PROTOCOLOS TCP/IP
2.2 Protocolos capa de transporte

- **Concepto de puerto**
  - En el equipo que inicia la comunicación también se asigna un número de puerto aleatorio a cada sesión (siempre mayor que 1024) para recibir el tráfico de retorno.
  - El conjunto formado por la **dirección IP** del equipo, el **protocolo** utilizado (**TCP o UDP**) y el **número de puerto** recibe el nombre de **socket**.
  - El concepto de **socket** permite que una aplicación pueda comunicarse con diferentes equipos de la red de forma simultánea. El motivo es que el socket usado con cada uno será diferente.

2. PROTOCOLOS TCP/IP
2.2 Protocolos capa de transporte

- **Concepto de puerto**



2. PROTOCOLOS TCP/IP
2.3 Protocolos capa de internet

- En esta capa encontramos los siguientes cinco protocolos:
  - **IP (*Internet Protocol*)**: Se encarga del transporte de datagramas, pero no garantiza su entrega.
  - **ARP (*Address Resolution Protocol*)**: Lo utilizan los enrutadores (routers) para obtener información sobre las diferentes redes que se encuentran a su alcance.
  - **ICMP** (*Internet Control Message Protocol*): Facilita la administración de los mensajes de error.
  - **IGMP (*Internet Group Management Protocol*)**: Permite que diferentes equipos formen parte de un grupo de multidifusión. Los routers analizan periódicamente si se mantiene la pertenencia.
  - **RARP (*Reverse Address Resolution Protocol*):** Facilita la resolución de la dirección IP de un dispositivo a partir de su dirección hardware (por ejemplo, su dirección Ethernet).

3. PROTOCOLO IP

- **Funcionalidades** de **nivel 3** o **nivel de red OSI**
  - **1. Encapsulación**: crear una unidad de datos de nivel 3 que recoja los datos de niveles superiores.
    - Los datos de niveles superiores (TCP o UDP) son encapsulados en los denominados **datagramas IP**.
  - **2. Direccionamiento a nivel de red**: determinar de forma unívoca en cada host en las distintas redes.
    - **Direcciones IP**
  - **3. Enrutamiento**: encontrar un camino desde el host origen hasta el host destino en base a dirección de destino IP.
    - Enrutamiento Los algoritmos de enrutamiento pueden ser adaptativos o no adaptativos.
  - **4. Control de la congestión**: evitar que se saturen los nodos (routers) y líneas intermedias a través de ciertos algoritmos de elección óptima de ruta u otros mecanismos.
    - Se pueden utilizar algoritmos adaptativos en los routers para control de la congestión

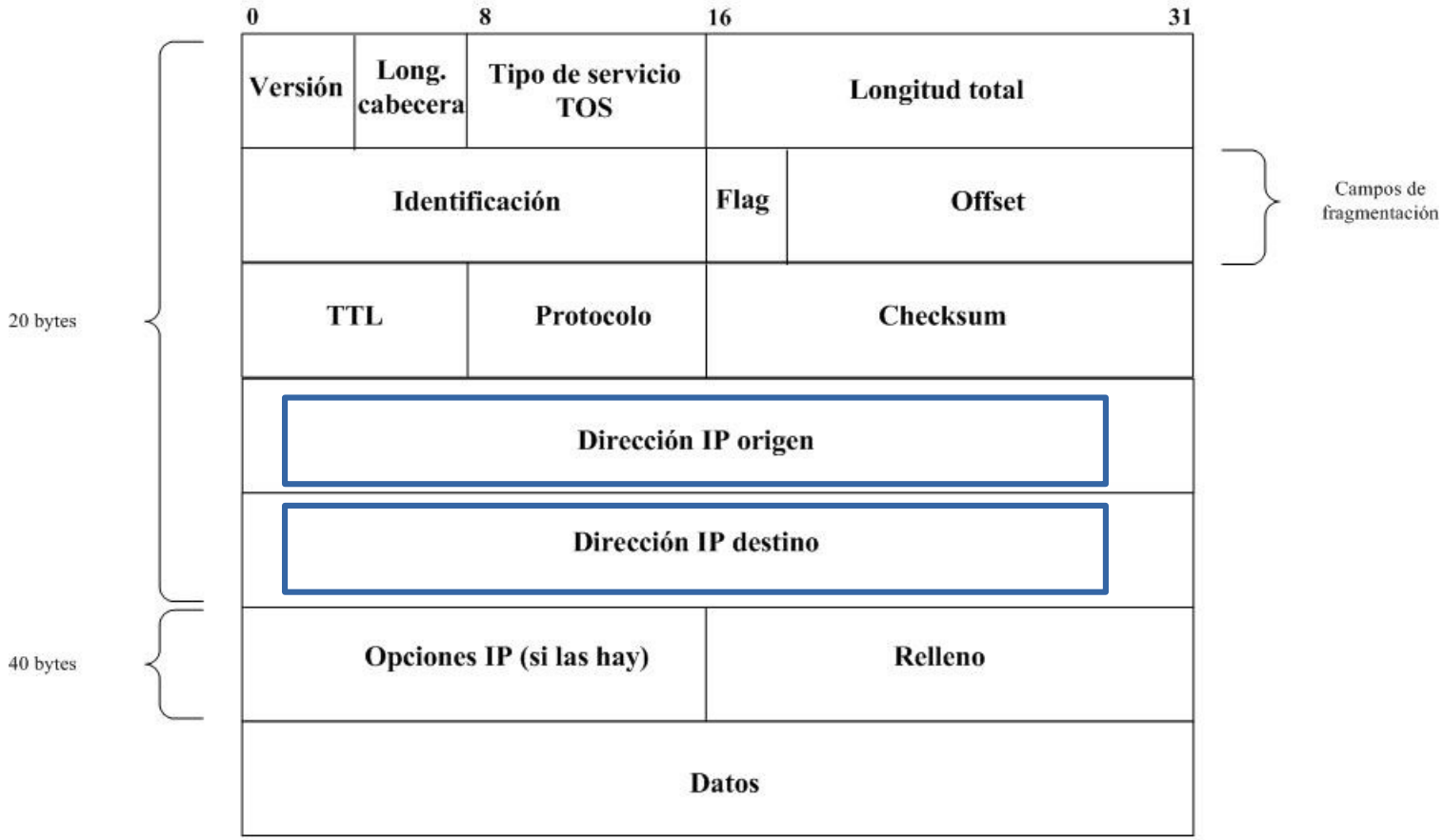
3. PROTOCOLO IP

Estructura Datagrama IPv4

- **Datagrama IP:** unidad de datos correspondiente (PDU) al protocolo IP en el nivel 3 del modelo OSI
- **Composición datagrama IP**:
  - Cabecera IP: que almacena información de control del protocolo IP
  - **Campo de datos**: donde se almacena los datos de nivel superior (TCP o UDP)
- **NOTA**: no se realiza control de errores de datos a nivel IP.

3. PROTOCOLO IP

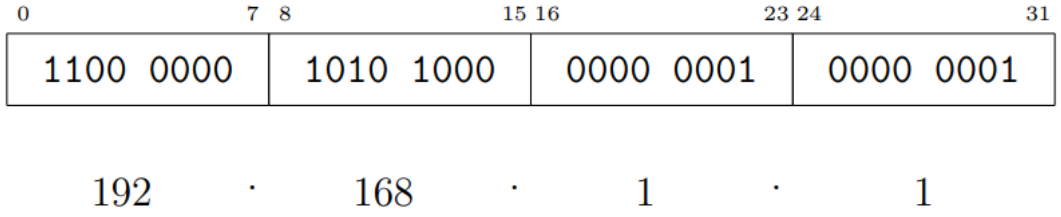
Estructura Datagrama IPv4



3. PROTOCOLO IP
3.1 Direccionamiento IPv4

Direcciones IP

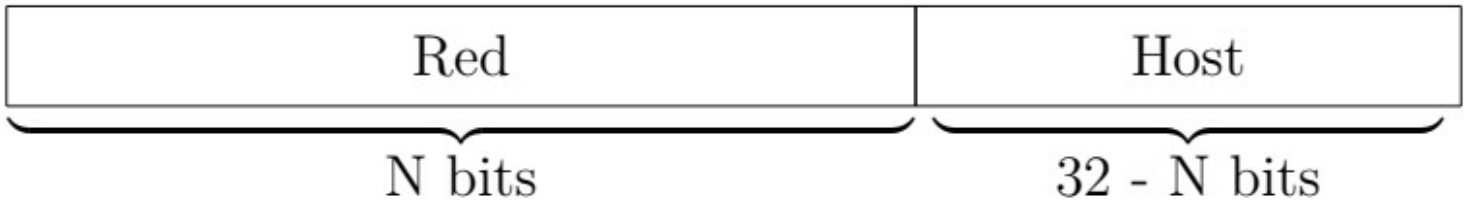
- Formado por **4 Bytes**: **32 bits,** estructurados en **4 octetos de 8 bits**
- Notación punto **decimal** en **4 cifras [0-255]**: aaaa.bbbb.cccc.####
  - Ej: 155.54.15.4, 192.168.3.5
- **Rango total**: 0.0.0.0 => 255.255.255.255
  - 2^32 posibles direcciones (teóricas)
  - 4.294.967.296 posibles direcciones (teóricas)
- Cada **interfaz de red (NIC)** deberá tener **asignada** una **dirección IP** para poder operar correctamente y la misma dirección no podrá asignarse a más de un host.



3. PROTOCOLO IP
3.1 Direccionamiento IPv4

Dirección IP

- Dentro de la **dirección IP** se distinguen **dos partes**:
  - La **parte de red** red identifica a la red, y corresponde a los N primeros bits
  - La **parte de host** identifica a un host concreto dentro de una red, mientras que la parte host corresponde a los 32−N bits restantes



- Las **direcciones de los hosts** pertenecientes la **misma red** de **área local** deben **compartir** del mismo prefijo o **parte de red**).

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

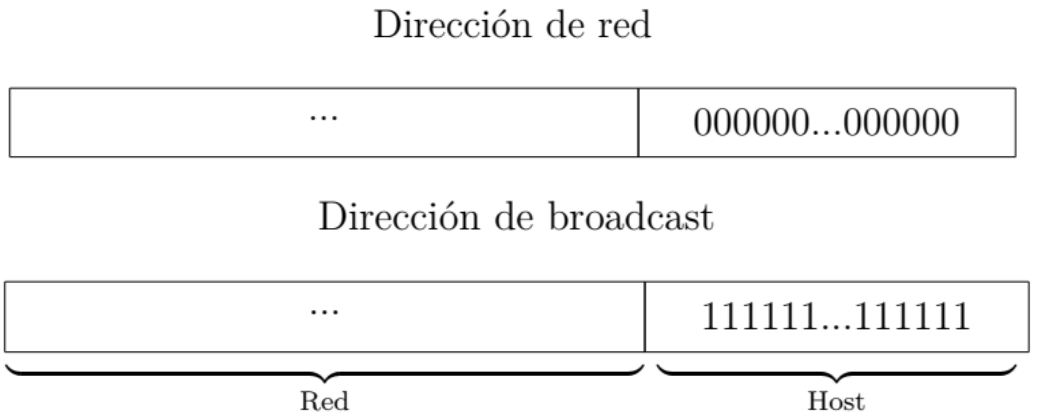
Máscara de red

- Para **distinguir** la **parte de red** de la **parte de host** se necesita lo que se conoce como **máscara de red**:
  - Su **formato** es **idéntico** a las **direcciones IP**, **32 bits en 4 octetos**, en el que se fijan **a 1 los bits de la parte de red** y **a 0 los bits de la parte de host**.
  - Al igual que las direcciones IP las máscaras de red se pueden expresar en notación decimal. Por ejemplo:

```
255.255.255.0     (11111111.11111111.11111111.00000000)
255.0.0.0         (11111111.00000000.00000000.00000000)
255.255.192.0     (11111111.11111111.11000000.00000000)
255.224.0.0       (11111111.11100000.00000000.00000000)
```

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

- Dentro de **cada red** se contemplan **2 direcciones especiales**:
  - **Dirección de red**: La dirección de red consiste en el **prefijo de red** fijando **a '0' todos los bits de la parte de host**.
    - Esta dirección no se puede asignar a ningún host, pues su misión es **identificar a la red**. No se pueden enviar paquetes a esta dirección.
  - **Dirección de broadcast**: La dirección de broadcast consiste en **el prefijo de red fijando '1' todos los bits de la parte de host**.
    - Esta dirección **no es asignable a ningún host**. Cuando se envía un paquete a esta dirección, éste está **destinado a todos los hosts de la red**.



3. PROTOCOLO IP
3.1 Direccionamiento IPv4

- Las **máscaras de red** permiten **separar**, de manera sencilla, la parte de una dirección IP que **identifica la red** a la que pertenece. Para lograrlo, basta con aplicar una **operación lógica AND** entre la **dirección IP** y su **máscara** correspondiente, ambas en formato binario. El resultado será la **dirección de red**.
  - En la siguiente imagen tienes un ejemplo en el que se aplica una máscara de red para una dirección IP:

	Decimal	Binario	
Dirección IP	120.140.3.48	01111000.10001100.00000011.00110000	AND
Máscara de red	255.0.0.0	11111111.00000000.00000000.00000000	
Dirección de red	120.0.0.0	01111000.00000000.00000000.00000000	

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

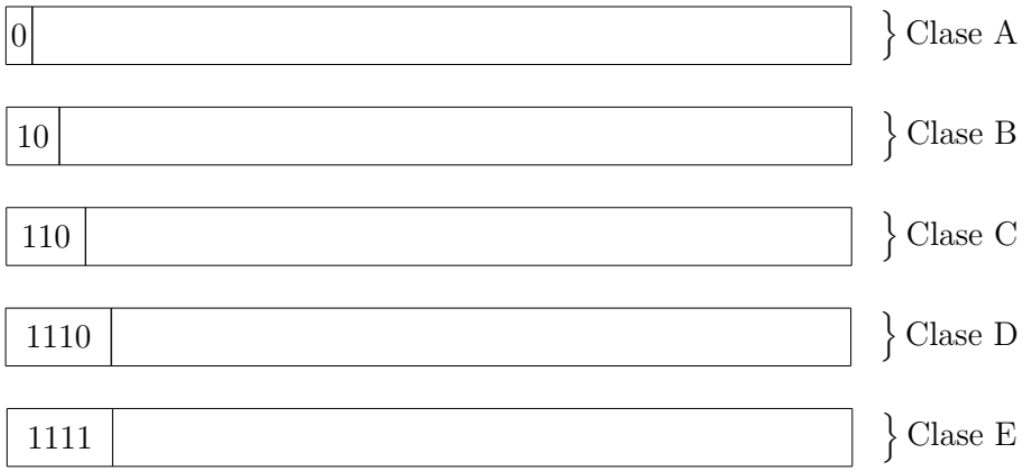
- **PROBLEMA:**
  - Lógicamente, cuanto **mayor** sea el **número de bits** reservados a la **identificación de la red**, **menor** será el **número de equipos** que pueden **formarla**.
  - Análogamente, cuanto **mayor** sea el **número de bits** reservados a la **identificación de hosts en la red la red**, **menor** será el **número de redes distintas** que podemos definir.
  - ¿Cómo decidir cuantos bits se asignan al ID de host y al ID de red dentro de la dirección IP? → **Configuración de la máscara de red** a emplear

- **SOLUCIONES:**
  - **Direccionamiento por clases**
  - **CIDR (Classless Inter-Domain Routing)**

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

Direccionamiento con clases

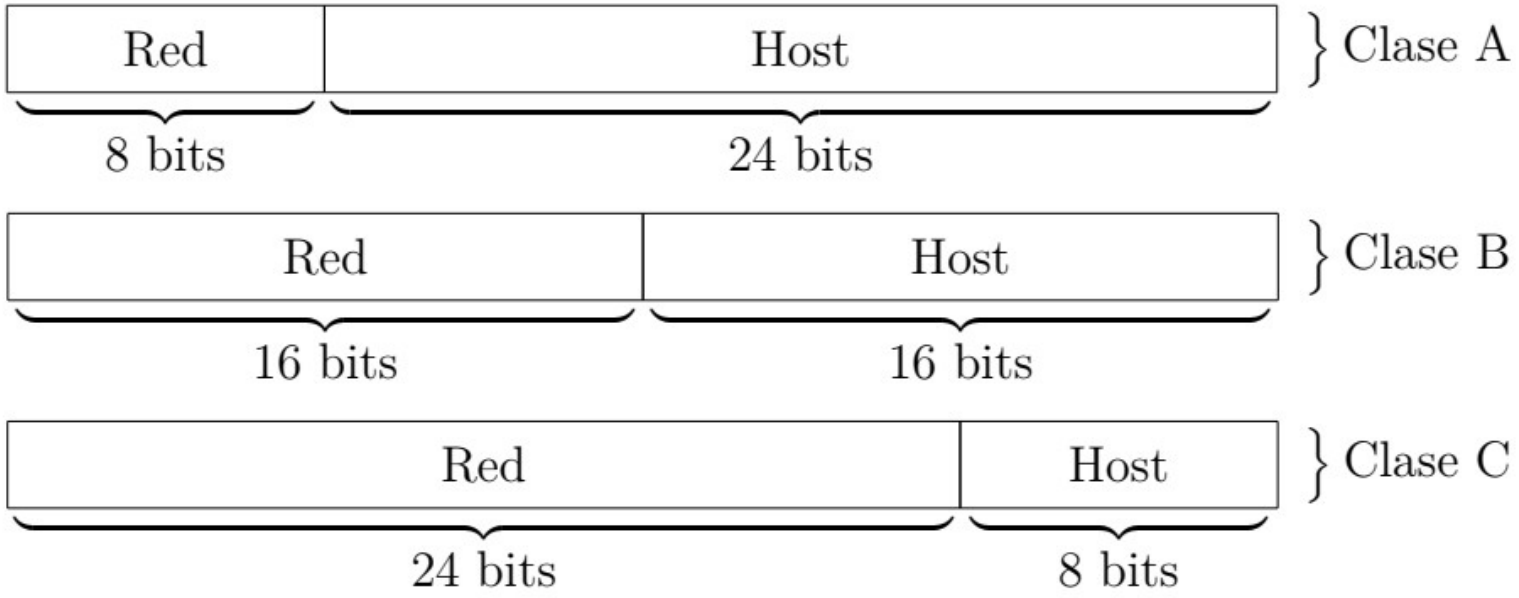
- En las **primeras redes IP** se utilizaba lo que se denomina **direccionamiento con clases**, que se introdujo en 1981 como parte de la definición del protocolo IP, distinguiéndose distintas **clases de direcciones.**
- Las **clases** se establecen en base a los **primeros bits de la dirección IP**.



3. PROTOCOLO IP
3.1 Direccionamiento IPv4

Direccionamiento con clases

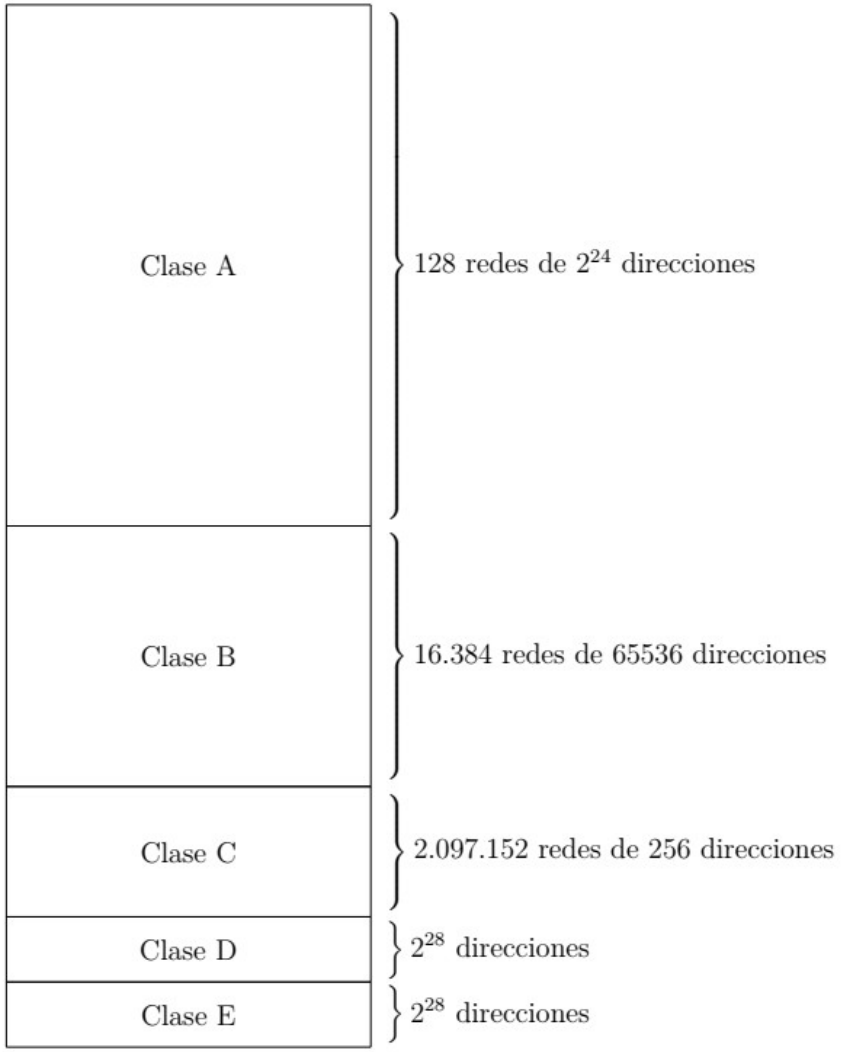
- Las direcciones de distintas clases difieren en los **tamaños de los prefijos de red**, es decir, tienen una **máscara de red asociada a cada clase.**



3. PROTOCOLO IP
3.1 Direccionamiento IPv4

Direccionamiento con clases

- En la figura, se muestra el **reparto de direcciones** entre las distintas **clases** del **rango total de direcciones** (0.0.0.0 a 255.255.255.255).
- Se puede observar que la **mitad de las direcciones IP** estaban destinadas a 128 **redes** de un **tamaño considerable**, **clase A**.
- Una **cuarta parte** va destinada a **redes de clase B**, que seguramente siguen siendo demasiado grandes para la mayoría de las organizaciones.
- Finalmente, una **octava parte** de las direcciones está destinada a **redes de tamaño pequeño**, **clase C**, que probablemente son demasiado pequeñas.



3. PROTOCOLO IP
3.1 Direccionamiento IPv4

Direccionamiento con clases

- Así, las redes se dividen según la siguiente **clasificación de clases**:
  - **Clase A**: Sólo se utiliza el **primer octeto (byte) para identificar la red**. Los 24 restantes se utilizan para identificar equipos en la red. Además, el **primer bit** estará siempre **a 0**, lo que significa que, en realidad, sólo se **utilizan 7 bits para identificar redes**. Es decir, el máximo número teórico de redes en esta clase serían 2^7=128 (de 0 a 127), pero como el valor **127 está reservado** (lo veremos más adelante) y **la red 0 no existe**, sólo nos **quedan 126 posibilidades** (de 1 a 126).
    - Esta clase se utiliza en implementaciones con un **número pequeño de redes** y un **gran número de equipos** en cada red.
  - **Clase B**: Emplea **16 bits** para el **netID** y los otros **16** para los **equipos**. Además, los **primeros dos bits** tienen **siempre el valor 10**, por lo que únicamente nos **quedan 14 bits** para **identificar redes**. Es decir, 2^{14}=**16.384**.
    - Se utiliza para **redes** que se encuentran a **medio camino** entre las de **clase A** y las de **clase C**.
  - **Clase C**: Pensada para **redes pequeñas**, utiliza los primeros **24 bits** para **identificar la red** y sólo **8 para los equipos** que la forman. Además, los **primeros tres bits** tienen **siempre** el **valor 110**, por lo que nos quedan **21 bits para identificar redes**. Esto nos arrojaría, siguiendo los cálculos anteriores, 2^{21}=2.097.152 redes diferentes.

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

Direccionamiento con clases

- **Clasificación de clases**:
  - **Clase D**: Utilizan sólo los **primeros 4 bits** para **identificar la red**, que **siempre** valen **1110**, y se emplean para **multidifusión de contenidos y protocolos de gestión interna de reds**, por lo que no se emplea para direccionar redes/hosts.
  - **Clase E**: Emplea únicamente los **4 primeros bits** para la **identificación de la red**, que siempre **valen 1111**, y está destinada a **uso en investigación**, y no se emplea tampoco para direccionar redes/hosts.

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

- **Direccionamiento con clases**
  - **Clasificación de clases**:

Clases de redes					
Clase	Formato	Intervalo	Redes	Equipos	Aplicación
A	0xxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	1.0.0.0 126.0.0.0	126	16777214	Redes grandes
B	10xxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	128.0.0.0 191.255.0.0	16384	65534	Redes medianas
C	110xxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	192.0.0.0 223.255.255.0	2097152	254	Redes pequeñas
D	1110xxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	224.0.0.0 239.255.255.255	-	-	Multicast
E	1111xxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	240.0.0.0 254.255.255.255	-	-	Investigación

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

Direccionamiento con clases

- Una máscara de red está formada por 32 bits, agrupados de ocho en ocho, con una construcción similar a una dirección IP. Sin embargo, a diferencia de ésta, en una máscara de red, todos los bits que estarían destinados a la identificación de red (**netID**) tendrían el valor 1 y los destinados a la identificación del equipo (**host-ID**) tendrían el valor 0.
- Por lo tanto, **en función de la clase de red**, tendríamos las siguientes **máscaras de red**:

Máscaras de red		
Clase	Binario	Decimal
A	11111111.00000000.00000000.00000000	255.0.0.0
B	11111111.11111111.00000000.00000000	255.255.0.0
C	11111111.11111111.11111111.00000000	255.255.255.0

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

Direcciones especiales

- Cuando hablamos de **direcciones IP**, existen algunos valores que tienen un **significado especial** y **no son asignables a hosts**:
  - **Dirección de red**: Es una dirección IP donde todos los **bits destinados a identificar equipos** aparecen con el **valor 0**. Por ejemplo, la dirección 192.168.1.0. corresponde con la dirección de una red de clase C.
  - **Dirección del equipo**: Al contrario de la anterior,es cuando todos los **bits** que **identifican la red** se ponen **a 0**. Ejemplo de clase C: 0.0.0.12
  - **Dirección de difusión**: Se obtiene poniendo **a 1 todos los bits que identifican al equipo**. Se trata de una dirección IP especial que se utiliza para enviar un **mensaje a todos los equipos** que formen parte **de la red**. Por ejemplo, en la red de clase C 192.168.0.0, la dirección de difusión sería 192.168.0.255.
  - **Dirección de bucle de retorno (*loopback*)**: Las direcciones de la forma **127.x.x.x**, hace referencia al **propio equipo** en el que nos encontramos. Normalmente se emplea el valor 127.0.0.1.

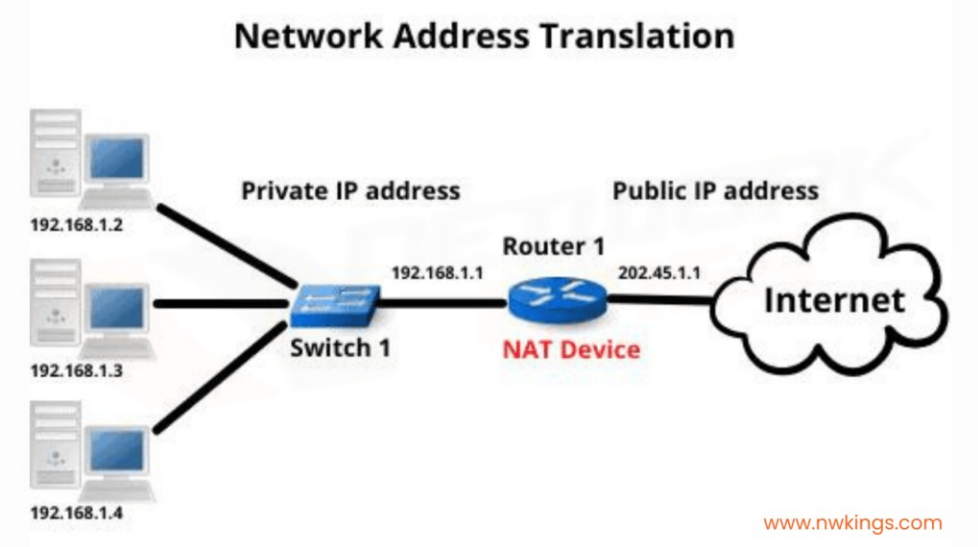
3. PROTOCOLO IP
3.1 Direccionamiento IPv4

- **Direcciones privadas**
  - Por otro lado, la mayor parte de las veces, los equipos de una red no tienen **acceso directo a Internet** y acceden a ésta a través de otro ordenador, que actúa como proxy, o de una pasarela.
  - Para estos casos, el **único dispositivo** que necesita disponer de una **dirección IP** asignada por la ICANN es el que actúa como **pasarela**, se define este tipo de dirección como **IP pública**.
  - Los demás, harán uso de un conjunto de **direcciones de ámbito local**, reservadas por la ICANN para este tipo de situaciones, las direcciones **IP privadas**. De este modo, no se producirán conflictos con las direcciones IP utilizadas en Internet y se liberan más direcciones públicas para asignar dado que las privadas se pueden reutilizar en cada red local. Estas direcciones se encuentran reflejadas en la siguiente tabla:

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

Direcciones privadas

- Estas **direcciones** utilizables **solo en redes privadas**: no son visibles desde Internet.
- **No se enrutan** en Internet
- Si se quiere conectar a Internet se necesita **mecanismos de translación de direcciones** público a privada y viceversa:
  - **NAT**
  - PAT/NAPT
  - Proxy's internos



Direcciones IP reservadas		
Clase	Valor Inicial	Valor fimal
A	10.0.0.1	10.255.255.254
B	172.16.0.1	172.31.255.254
C	192.168.0.1	192.168.0.254

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

- **CIDR (*Classless Inter-Domain Routing)***
- **PROBLEMÁTICA** direccionamiento por clases.
  - El esquema de distribución y clasificación en redes por clases (classful) **"desperdiciaba" gran nº de IP's** en asignación de **redes grandes.**
  - **No era un esquema ágil** para la **asignación** precisa del **nº de IP's** que necesita realmente la organización.
    - Sólo se tienen tres tipos de tamaño de redes A,B,C
    - No es flexible a la hora de asignar un nº específico de IP's
  - Con la expansión de Internet este sistema "**consumía" muchas IP's** sin necesidad

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

- **CIDR (*Classless Inter-Domain Routing*)**
  - En 1993, la IETF introdujo un modo de notación simplificada para las máscaras de red, a la que llamó **CIDR** (***Classless Inter-Domain Routing***).
  - Esta notación propone escribir la dirección IP seguida de una barra inclinada **(*slash*) y un valor** que representa el **número de unos consecutivos** que tiene la **máscara de red por la izquierda**.
    - Así, por ejemplo, si la configuración de un equipo dispone de una dirección IP **192.168.0.32** y una máscara de red **255.255.255.0**, también podríamos representarla, de forma resumida, como **192.168.0.32/24**.

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

- **CIDR (*Classless Inter-Domain Routing*)**
  - Sistema de distribución de IP's **no atendiendo a las clases** A, B o C
  - No solo utiliza bloques de redes de máscara 8, 16 o 24 bits sinoque puede utilizar **máscaras de otras longitudes**
  - Va muy asociado a la **segmentación de redes VLSM**
  - Permite una **distribución y uso más eficiente** del espacio de **direcciones IPv4**
  - Permite **sumarización** y **superredes** agilizando así las entradas de las **tablas de enrutamiento**

3. PROTOCOLO IP
3.1 Direccionamiento IPv4

- **CIDR (*Classless Inter-Domain Routing*)**
  - En la siguiente tabla incluimos las **equivalencias** entre los **valores de máscaras de red** representadas con sus **valores decimales** y su equivalencia con **notación CIDR**:

Clase	Decimal	CIDR
Clase A	255.0.0.0	/8
	255.128.0.0	/9
	255.192.0.0	/10
	255.224.0.0	/11
	255.240.0.0	/12
	255.248.0.0	/13
	255.252.0.0	/14
	255.254.0.0	/15
Clase B	255.255.0.0	/16
	255.255.128.0	/17
	255.255.192.0	/18
	255.255.224.0	/19
	255.255.240.0	/20
	255.255.248.0	/21
	255.255.252.0	/22
	255.255.254.0	/23
Clase C	255.255.255.0	/24
	255.255.255.128	/25
	255.255.255.192	/26
	255.255.255.224	/27
	255.255.255.240	/28
	255.255.255.248	/29
	255.255.255.252	/30

3. PROTOCOLO IP
3.2 Segmentación de redes

Subnetting CIDR

- **CIDR** permite tener **máscara** que no sea 8, 16 o 24 bits
- Se utiliza en técnica para **particionar una red en subredes:**
  - Las subredes pueden subdividirse a su vez
  - Las subredes finales resultantes pueden tener distintos tamaños
- Esta técnica **utiliza bits de la parte de HOST** para **subdividir la red** principal
  - Esos bits serán los **bits de subred**
  - **Nº subredes** posibles: 2ˆ(num bits de subred)
  - **Nº de equipos por subred**: (2ˆ(num bits de host))-2

3. PROTOCOLO IP
3.2 Segmentación de redes

Subnetting CIDR

- **Ejemplo: particionado** de la **red 192.168.0.0/24** en **2 subredes**:
  - Utilizamos la **máscara /25** => **1 bit de subred** => **2^1** => **2 subredes**
  - Número de **equipos por subred** => **(2^7)-2** => **126**
    - **1º Subred: 192.168.0.0/25**
      - 1ª IP: 192.168.0.1/25 asignable a host
      - Última IP: 192.168.0.126/25 asignable a host
      - Bbroadcast de la subred: 192.168.0.127/25
    - **2º Subred: 192.168.0.128/25**
      - 1ª IP: 192.168.0.0.129/25
      - Última IP: 192.168.0.254/25
      - Broadcast de la subred: 192.168.0.255/25

3. PROTOCOLO IP
3.2 Segmentación de redes

Subnetting CIDR

- **Ejemplo** de particionado de la red **192.168.0.0/24** en **2 subredes**:

Subred 1	192 . 168 . 0 . 0 /25	&	11000000 . 10101000 . 00000000 . 00000000
	255 . 255 . 255 . 128		11111111 . 11111111 . 11111111 . 10000000
ID de Subred 1	192 . 168 . 0 . 0		11000000 . 10101000 . 00000000 . 00000000
Subred 2	192 . 168 . 0 . 128/25	&	11000000 . 10101000 . 00000000 . 10000000
	255 . 255 . 255 . 128		11111111 . 11111111 . 11111111 . 10000000
ID de Subred2	192 . 168 . 0 . 128		11000000 . 10101000 . 00000000 . 10000000
Dir. IP	192 . 168 . 0 . 126/25	&	11000000 . 10101000 . 00000000 . 01111110
Máscara	255 . 255 . 255 . 128		11111111 . 11111111 . 11111111 . 10000000
ID de Red	192 . 168 . 0 . 0		11000000 . 10101000 . 00000000 . 00000000

bits de subred

bits de host

3. PROTOCOLO IP
3.2 Segmentación de redes

Subnetting CIDR

- También se puede **dividir una red ya segmentada**
- **Ejemplo**: **dividir en 2** la **red 172.16.0.0/19**
  - Utilizamos **/20** => **1 bit más de subred** => **2^1 = 2 subsubredes**
  - **Cada subred** tendrá (2^12)-2 = **4094 equipos por subred**
    - **1º Subred: 172.16.0.0/20**
      - 1ª IP: 172.168.0.1/20
      - Última IP: 172.168.15.254/20
      - Broadcast: 172.168.15.255/20
    - **2º Subred: 172.16.16.0/20**
      - 1ª IP: 172.168.16.1/20
      - Última IP: 172.168.31.254/20
      - Broadcast: 172.168.31.255/20

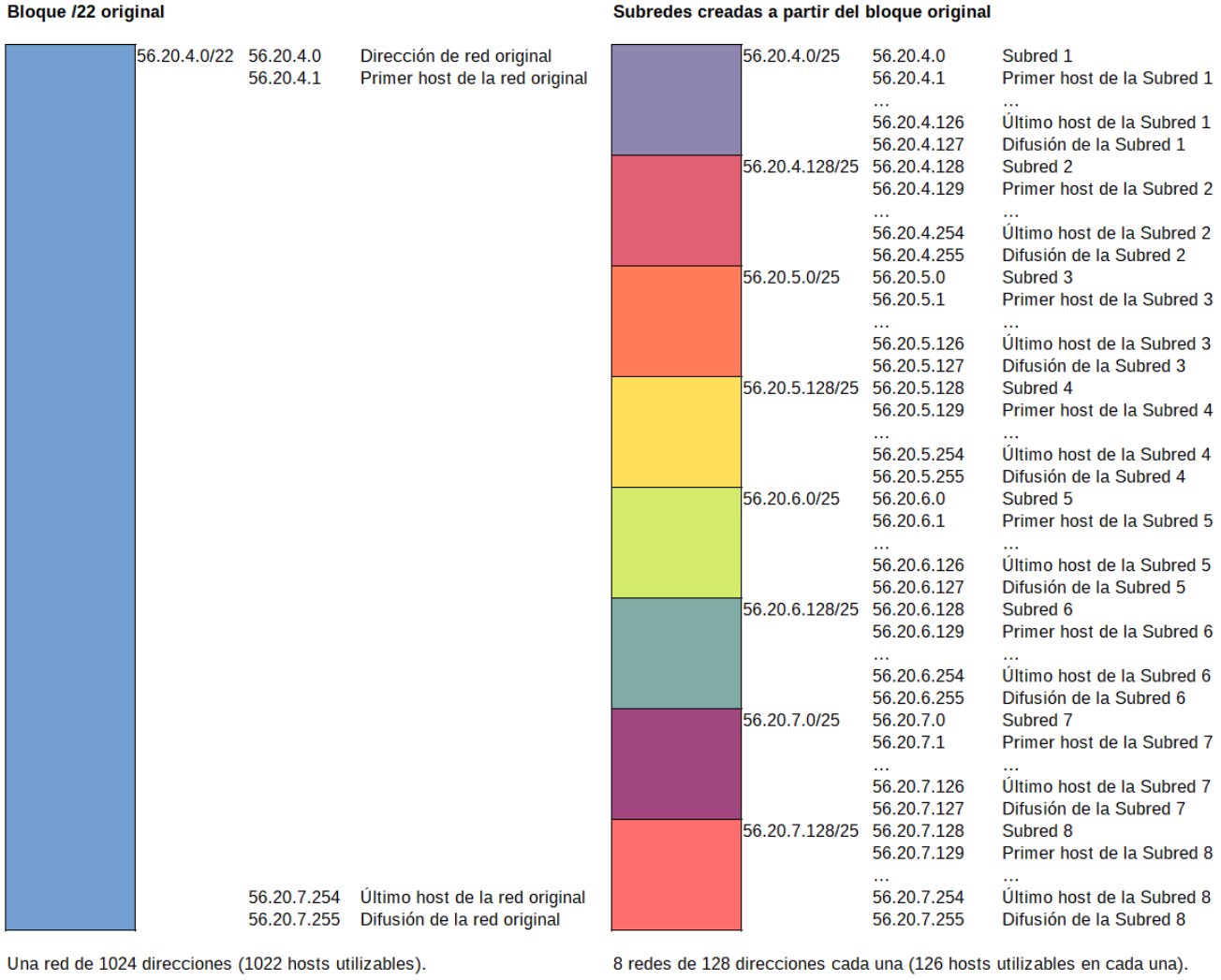
3. PROTOCOLO IP
3.2 Segmentación de redes

- **Subnetting VLSM (*Variable Length Subnetting Mask*)**
  - Se trata de poder **subdividir una red** original en subredes que tengan **distintos tamaños.**
  - Permite **adaptar el tamaño** de las **distintas subredes** creadas a las necesidades específicas de cada caso.
  - Está muy relacionado con **CIDR** ya que al hacer esto los routers deben saber anunciar las redes VLSM con CIDR (sin clases) para poder enrutar bien.

3. PROTOCOLO IP
3.2 Segmentación de redes

Subnetting VLSM

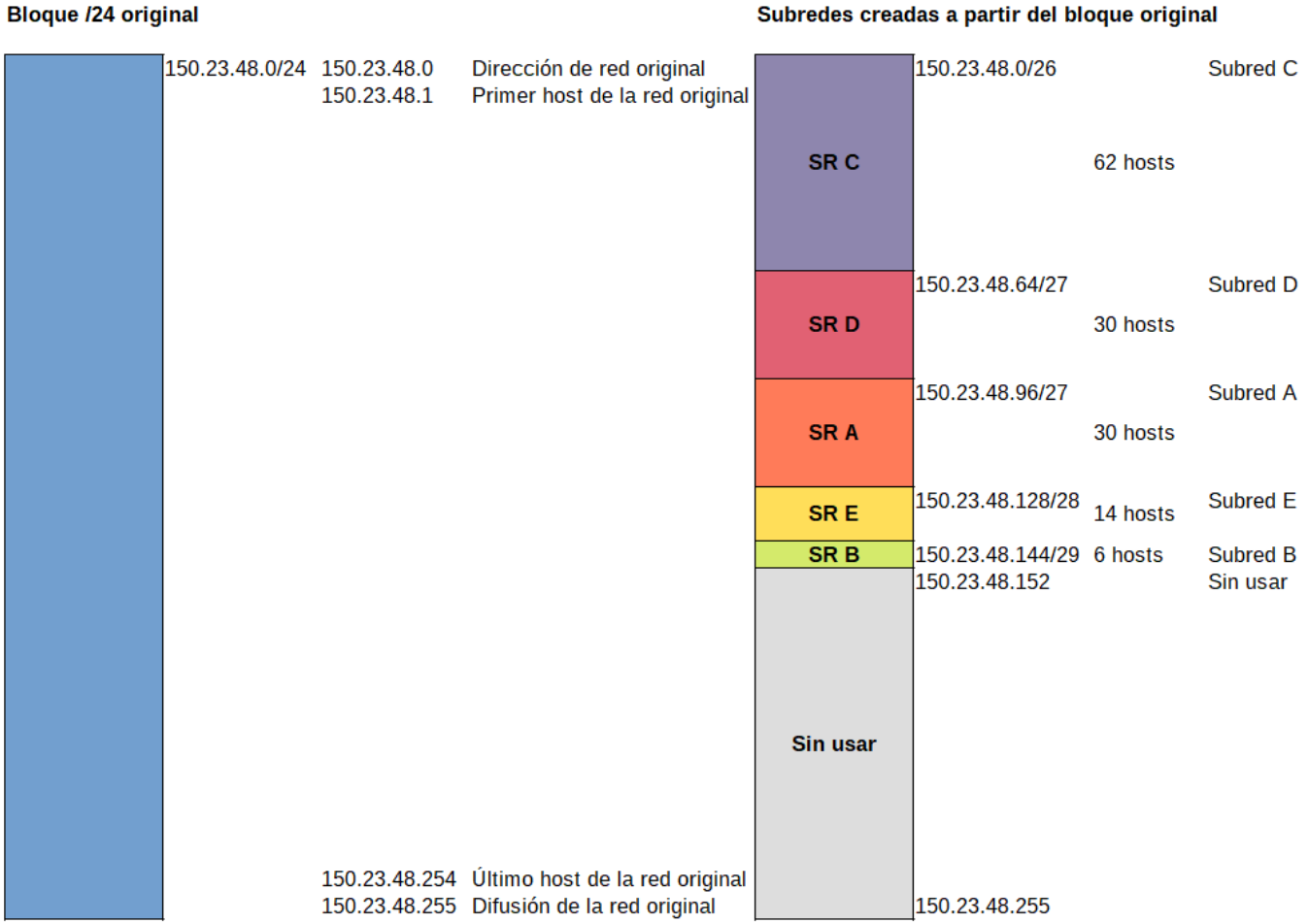
- **Comparativa subnetting CIDR vs VLSM:**
  - **CIDR:** subredes de tamaño fijo



3. PROTOCOLO IP
3.2 Segmentación de redes

Subnetting VLSM

- **Comparativa subnetting CIDR vs VLSM:**
  - **VLSM:** subredes de tamaño variable



3. PROTOCOLO IP
3.2 Segmentación de redes

- **Subnetting VLSM (*Variable Length Subnetting Mask*)**
  - **Ejemplo**: subdividir una **red original 192.168.0.0/24** de **254 equipos** en
    - 1 **Subred SRA** de **60 equipos**
    - 1 **Subred SRB** de **30 equipos**
    - 1 **Subred SRC** de **100 equipos**

3. PROTOCOLO IP
3.2 Segmentación de redes

- # Subnetting VLSM
  - ## Ejemplo
    - ### Paso 1
      - **Sumar los equipos** que necesitamos y **verificar que es inferior** al **total** de la red original
      - En nuestro caso: 60 + 30 + 100 = 190 < 254 => OK
    - ### Paso 2:
      - **Ordenar de mayor a menor** las **subredes** a obtener **por tamaño** descendente
      - En nuestro caso:
        - **SRC**: 100 equipos
        - **SRA**:60 equipos
        - **SRB**: 30 equipos

3. PROTOCOLO IP
3.2 Segmentación de redes

Subnetting VLSM

- **Ejemplo**
  - **Paso 3:**
    - **Asignar máscara de subred** más larga posible a cada una de las subredes y de forma consecutiva
    - En nuestro caso:
      - **192.168.0.0/25** => 126 equipos para la **subred SRC**
      - **192.168.0.128/26** => 62 equipos para la **subred SRA**
      - **192.168.0.192/27** => 30 equipos para la **subred SRB**

3. PROTOCOLO IP
3.3 Sumarización de redes

Sumarización de subredes/redes

- Con la misma técnica VLSM se puede realizar **agrupación de subredes**
- **Sumarización:**
  - **Agrupar subredes para identificarlas** como un **grupo único**
  - Útil para **reducir las tablas de enrutamiento** en routers
  - La técnica de sumarización utiliza la reducción de los bits de subred para agrupar subredes

3. PROTOCOLO IP
3.3 Sumarización de redes

Sumarización de subredes/redes

- **Ejemplo de Sumarización:**
  - **Subredes iniciales**:
    - 10.56.248.0/24
    - 10.56.249.0/25
    - 10.56.249.128/26
    - 10.56.249.192/26
    - 10.56.250.0./23
- Se quiere **agrupar esas subredes** (sumarizar)

3. PROTOCOLO IP
3.3 Sumarización de redes

- **Sumarización de subredes/redes**
  - **Solución al ejemplo 10.56.248.0/22** (agrupa/sumariza a todas ellas)
    - **Cuidado**: otras soluciones serían
      - 10.56.0.0/16
      - 10.0.0.0/8
    - Pero lo que se busca es **sumarizar** con la máscara más larga posible para **incluir las subredes indicadas pero sin abarcar demasiadas subredes adyacentes**
    - La **sumarización óptima** en este caso es **10.56.248.0/22**

3. PROTOCOLO IP
3.3 Sumarización de redes

Sumarización de subredes/redes

- **Técnica para obtener la sumarización óptima**:
  - **Paso 1**. Se pasan a **binario las subredes** a agrupar
    - 10.56.248.0/24 = 00001010.00111000.11111000.00000000
    - 10.56.249.0/25 = 00001010.00111000.11111001.00000000
    - 10.56.249.128/26 = 00001010.00111000.11111001.10000000
    - 10.56.249.192/26 = 00001010.00111000.11111001.11000000
    - 10.56.250.0/23 = 00001010.00111000.11111010.00000000

3. PROTOCOLO IP
3.3 Sumarización de redes

Sumarización de subredes/redes

- **Técnica para obtener la sumarización óptima**:
  - **Paso 2**. Se obtiene el nº **bits más significativos iguales** entre todas
    - **00001010.00111000.11111**00.00000000
    - **00001010.00111000.11111**01.00000000
    - **00001010.00111000.11111**01.10000000
    - **00001010.00111000.11111**01.11000000
    - **00001010.00111000.11111**10.00000000
    - En este caso vemos que hay **22 bits iguales**,
    - La **máscara óptima de sumarización** será **/22**

3. PROTOCOLO IP
3.3 Sumarización de redes

Sumarización de subredes/redes

- **Técnica para obtener la sumarización óptima**:
  - **Paso 3**. Se **aplica la máscara /22** a **cualquiera** de ellas para obtener el **ID de red sumarizada**
    - **A 10.56.248.0 le aplicamos máscara /22**
      - 00001010.00111000.11111000.00000000
      - 11111111.11111111.11111100.00000000
      - 00001010.00111000.11111000.00000000
    - O también, a **10.56.249.0 le aplicamos máscara /22**
      - 00001010.00111000.11111001.00000000
      - 11111111.11111111.11111100.00000000
      - 00001010.00111000.11111000.00000000
    - **ID de Red (sumarizada): 10.56.248.0/22 (en ambos casos)**

3. PROTOCOLO IP
3.4 IPv6

El protocolo IPv6

- Hasta ahora cuando hablamos de protocolo **IP** nos referíamos a **IPv4**, que se implementó en **1983** para su uso en la red **ARPANET**.
- En aquella época, los **4.294.967.296 (2^{32}) direcciones** únicas parecían más que suficientes, pero **después del crecimiento** que ha sufrido **Internet** desde entonces, han sido completamente **insuficientes**. Sobre todo por varios motivos:
  - Porque muchas de ellas están reservadas para redes locales
  - Porque en la actualidad cada usuario individual puede tener múltiples dispositivos conectados a Internet (ordenadores, teléfonos, tablets, etc.)
  - Porque en el futuro inmediato, con la generalización del Internet de las cosas (IoT), se prevé que los tipos de dispositivos conectados se diversifique mucho más (vehículos, televisores y otros muchos dispositivos domésticos)

3. PROTOCOLO IP
3.4 IPv6

- **Notación simplificada direcciones IPv6:**
- Las direcciones IPv6 tienden a ser difíciles de manejar, existen formas abreviadas de escribir algunas de ellas:
  - Si en la dirección IPv6 tenemos **campos cuyo valor es cero**, podemos **representarlos** con **un único cero** en lugar de cuatro.
    - Por ejemplo, la **dirección anterior**, podríamos escribirla como **2001:0db8:ac10:0013:0:0:2b4e:0c11**
  - Incluso podemos ir más allá y **eliminar el campo** por completo: **2001:0db8:ac10:0013:::2b4e:0c11**
  - Sin embargo, esto **no podemos hacerlo dos veces en la misma dirección**.
    - Es decir, si la **dirección original** fuese **2001:0db8:ac10:0000:0000:0013:0000:0000**, podríamos eliminar completamente uno de los dos bloques de ceros, pero **no ambos**.
    - Por lo tanto, serían válidas las siguientes abreviaturas: 2001:0db8:ac10::0013:0:0 y 2001:0db8:ac10:0:0:0013::.
  - Si el **número de campos consecutivos que se encuentran a cero son más de** dos, la **abreviatura sería la misma**.
    - Es decir, si la dirección original fuese 2001:0db8:0000:0000:0000:0000:2b4e:0c11, podríamos escribir 2001:0db8::2b4e:0c11
  - También **pueden omitirse los dígitos a la izquierda de un campo cuando su valor es cero**. Además, podemos utilizarlo como complemento a cualquiera de los métodos anteriores.
    - Por ejemplo: **2001:db8::2b4e:c11**

3. PROTOCOLO IP
3.4 IPv6

- ## Estructura de direcciones IPv6
- Desde un punto de vista lógico, las direcciones IPv6 se dividen en **tres partes**:
  - **Prefijo del sitio**: Ocupa, **como máximo, los tres primeros campos (48 bits)** y forman la **parte pública de la dirección**. Suele expresarse en **notación CIDR**. Así, cuando escribimos 2001:db8:ac10:13:0:0:2b4e:c11/48 estaremos indicando que el prefijo ocupa los tres primeros campos.
    - Si queremos referirnos únicamente al prefijo, podríamos utilizar la notación con ceros comprimidos: 2001:db8:ac10::/48

3. PROTOCOLO IP
3.4 IPv6

- **Estructura de direcciones IPv6**
- Desde un punto de vista lógico, las direcciones IPv6 se dividen en **tres partes**:
  - **Prefijo de subred**: suele ocupar el **cuarto campo (16 bits)** y le permite al **enrutador (router)** identificar la **topología interna de la red**.
    - Para expresar la subred suele emplearse también **notación CIDR**. Por ejemplo: 2001:db8:ac10:13::/64
    - La suma entre los **bits del prefijo del sitio y los del prefijo de subred será siempre 64: netID**.
  - **ID de la interfaz**: Serán los **cuatro últimos campos (64 bits)**. Suele llamarse ***token.*** Su valor puede asignarse manualmente o provenir de la **dirección MAC** (*Media Access Control*) de la tarjeta de red mediante el formato **EUI-64** (*Extended Unique Identifier*).

3. PROTOCOLO IP
3.4 IPv6

Direcciones IPv6 con direcciones IPv4 incrustadas

- Es posible **combinar direcciones IPv6 y direcciones IPv4** de forma que éstas últimas se **incrusten** en las primeras.
- Para lograrlo, la dirección **IPv6 se divide en dos partes**:
  - La primera sigue utilizando **notación hexadecimal** y representa los **6 primeros campos de la dirección**.
  - La segunda (**el segmento IPv4)** tiene **cuatro campos** y utiliza **notación decimal** con valores de **8 bits**.
- Así, se **asegura la compatibilidad** de los equipos que aún funcionen con una configuración IPv4 y los que ya dispongan de configuración IPv6 dentro de la misma red. De este modo, los dispositivos de red que trabajen con IPv6 representarán las direcciones de los dispositivos IPv4 como direcciones IPv6
- Un ejemplo: 0:0:0:0:0:ffff:192.1.1.25
  - También podemos escribirlo en formato abreviado así: ::ffff:192.1.1.25/96.

4. PROTOCOLOS TRANSPORTE

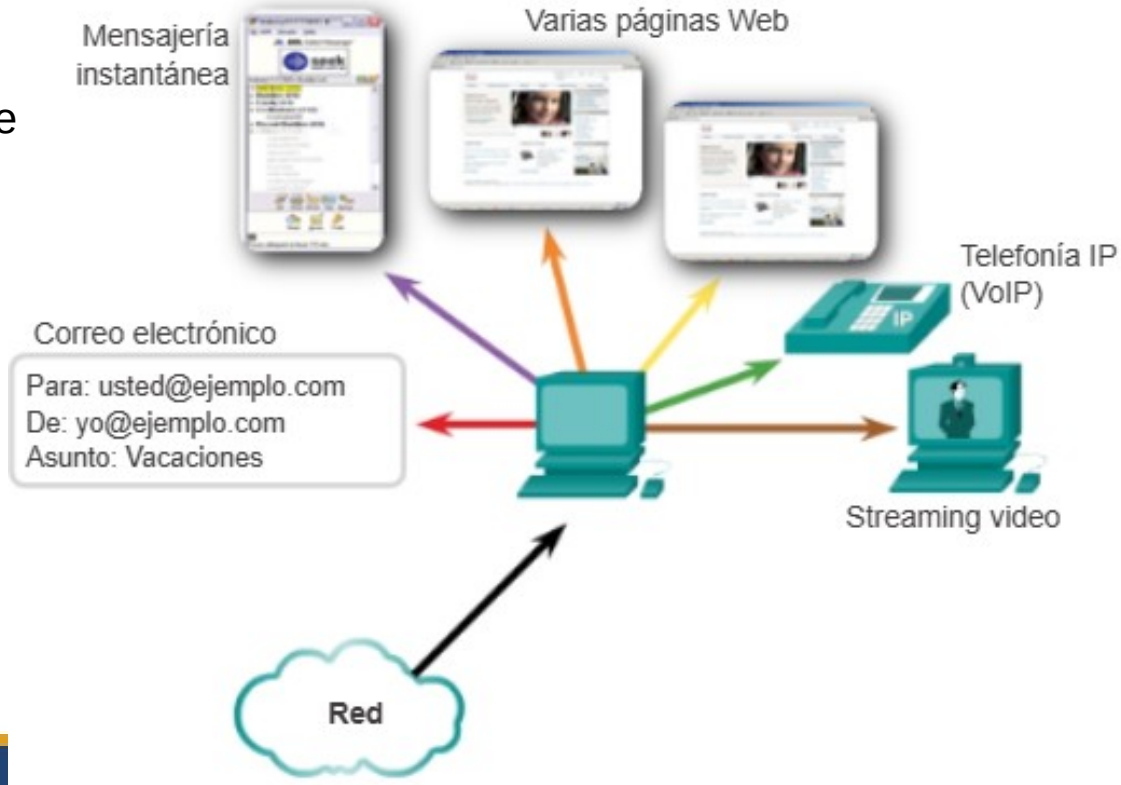
- El **protocolo TCP** corresponde al **nivel 4** o de **transporte** en la arquitectura de redes del **modelo OSI**.
- La **capa de transporte** es responsable de **establecer** una **sesión de comunicación** temporal **entre dos aplicaciones** y de **transmitir datos** entre ellas.

4. PROTOCOLOS TRANSPORTE

- La **capa de transporte** proporciona un método para **entregar datos** a través de la red de una manera que **garantiza** que estos **se puedan** volver a **unir correctamente** en el extremo **receptor**.
- La capa de transporte **permite la segmentación de datos** y proporciona el **control** necesario para **rearmar estos segmentos** en los distintos streams de comunicación.
- En el **protocolo TCP/IP**, estos procesos de segmentación y rearmado se pueden lograr utilizando **dos protocolos muy diferentes** de la **capa de transporte**:
  - El **protocolo TCP** *(Transmission Control Protocol)*
  - El **protocolo UDP** (*User Datagram Protocol*).

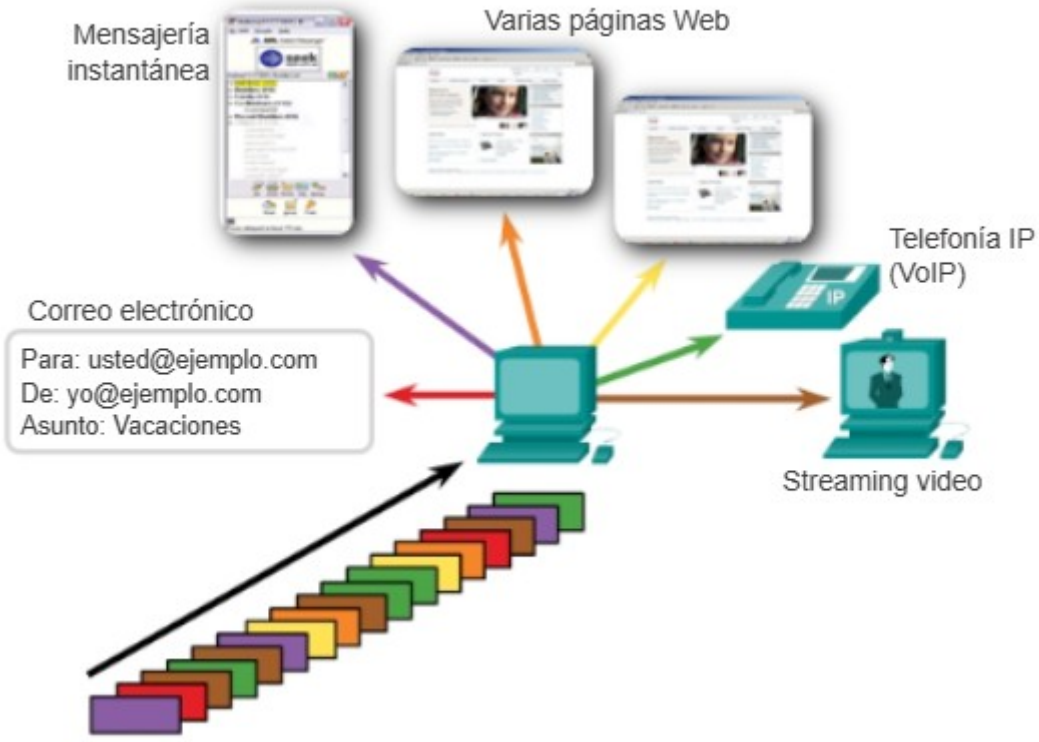
4. PROTOCOLOS TRANSPORTE

- Las **principales funciones** de los **protocolos** de la **capa de transporte** son :
  - **Rastreo de comunicación** individual **entre aplicaciones** en los hosts de origen y destino
    - En la capa de transporte, cada conjunto de datos particular que fluye entre una **aplicación de origen y una de destino** se conoce como **conversación**. Un host puede tener varias aplicaciones que se comunican a través de la red de forma simultánea, cada una de las cuales se comunica con una o más aplicaciones en uno o más hosts remotos. Es responsabilidad de la **capa de transporte mantener y hacer un seguimiento** de todas estas **conversaciones**.



4. PROTOCOLOS TRANSPORTE

- Las **principales funciones** de los **protocolos** de la **capa de transporte** son :
  - **División de los datos** en **segmentos** para su administración y reunificación de los datos segmentados en ***streams*** de datos de aplicación en el destino
    - Los protocolos de la capa de transporte tienen servicios que segmentan los datos de aplicación en **bloques de datos** de un **tamaño apropiado**. Estos servicios incluyen la **encapsulación** necesaria en cada porción de datos. Se agrega un **encabezado** a cada bloque de datos.
    - En el destino, la capa de transporte debe poder **reconstruir** las porciones de datos en un **stream de datos** completo que sea útil para la capa de aplicación, empleando la información del **encabezado**.



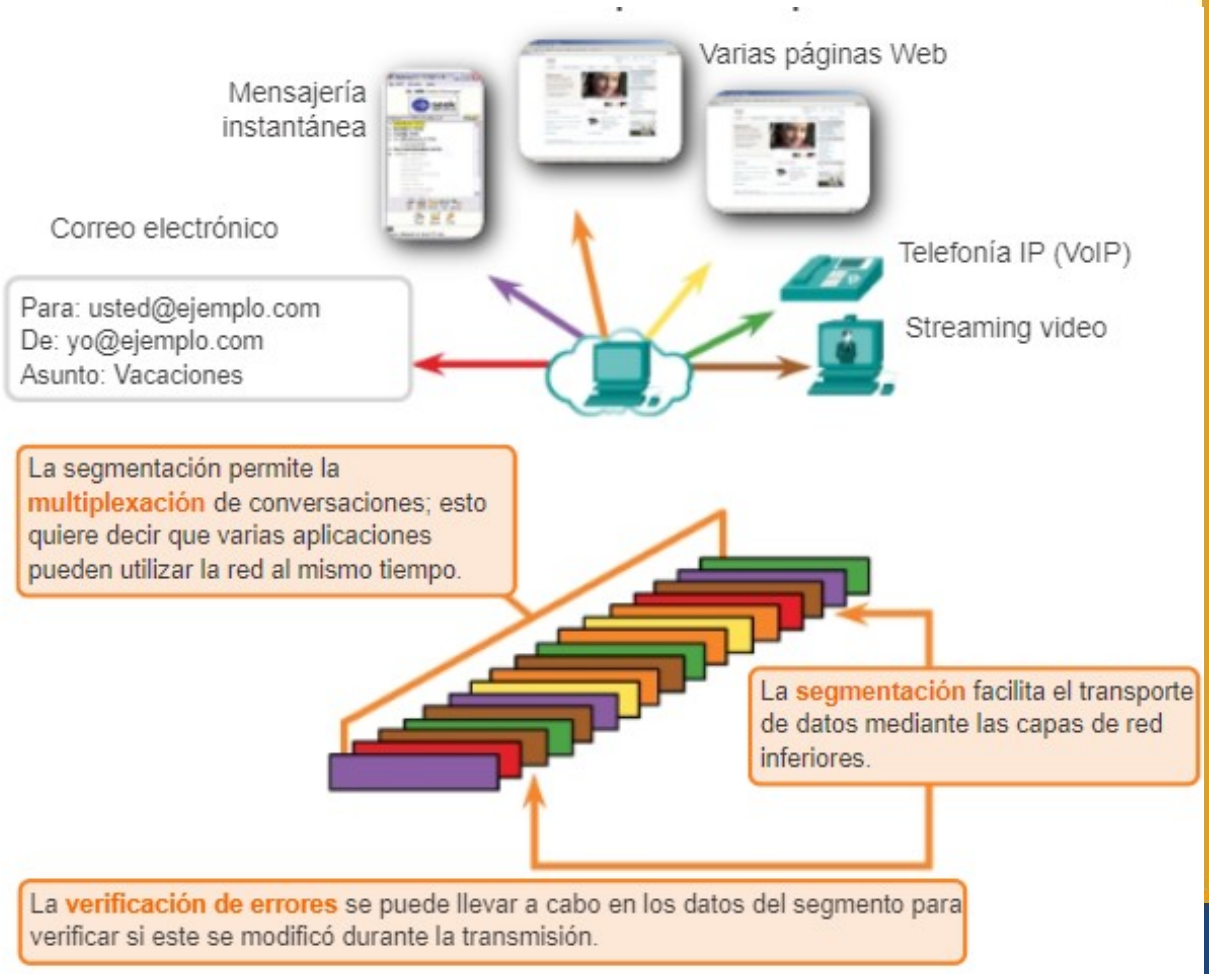
4. PROTOCOLOS TRANSPORTE

- Las **principales funciones** de los **protocolos** de la **capa de transporte** son :
  - **Identificación** de la **aplicación** correspondiente para cada stream de comunicación.
    - Puede haber muchas aplicaciones o servicios que se ejecutan en cada host de la red, las cuales deben ser dirigidas a las aplicaciones adecuadas.
    - La capa de transporte asigna un identificador, el **número de puerto,** a cada aplicación que requiera acceder a la red en ese host. La capa de transporte utiliza **puertos para identificar la aplicación o el servicio**.

4. PROTOCOLOS TRANSPORTE

Multiplexación de conversaciones

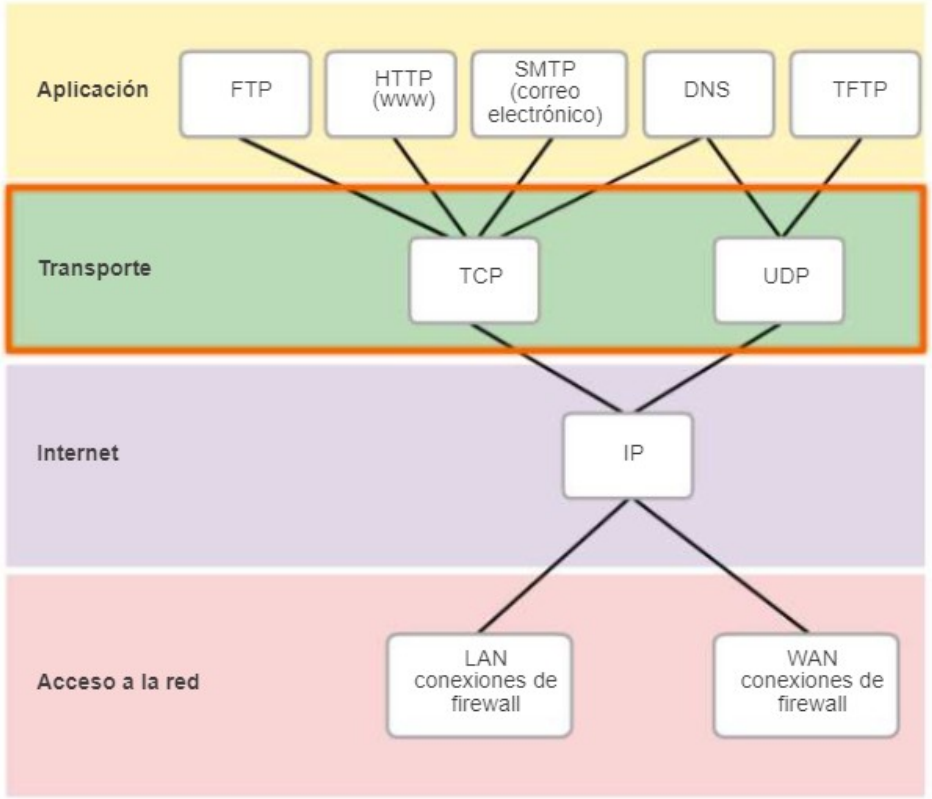
- La **segmentación** de los datos en partes más pequeñas permite que se entrelacen (**multiplexen**) **varias comunicaciones** de **distintos usuarios** en la misma red, pero tambnién sobre varias aplicaciones a la vez en un mismo PC.
- Para **identificar cada segmento** de datos, la capa de transporte agrega al segmento un **encabezado** permiten que los distintos protocolos de la capa de transporte administren la comunicación de datos.



4. PROTOCOLOS TRANSPORTE

Confiablildad de la capa de transporte

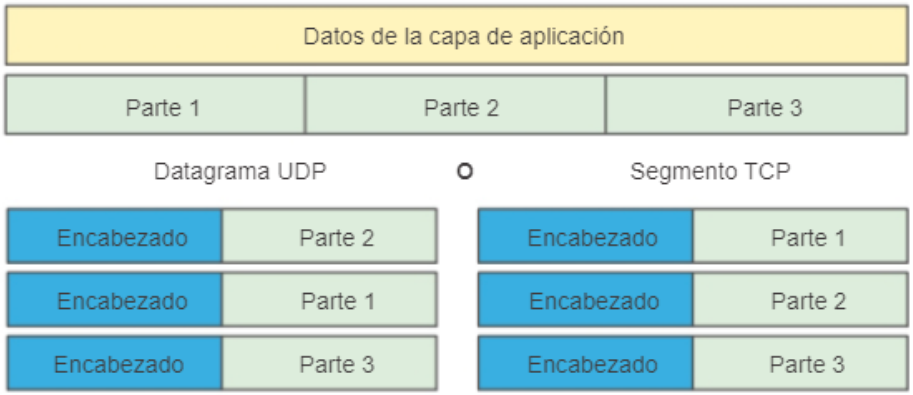
- La capa de transporte también es responsable de administrar los requisitos de **confiabilidad** de las **conversaciones requeridos** por las **aplicaciones**
- **IP** se ocupa solo de la estructura, el direccionamiento y el enrutamiento de paquetes, y **no especifica** la **manera** en que se lleva a cabo la **entrega o el transporte** de los **paquetes**.
- **Los protocolos de transporte** especifican la manera en que se transfieren los mensajes entre los host:
  - **TCP** se considera un protocolo **confiable y completo**, que garantiza que todos los datos lleguen al destino.
  - **UDP** es un protocolo muy simple que **no proporciona confiabilidad**.



4. PROTOCOLOS TRANSPORTE

TCP y UDP: manejo distinto de la segmentación

- Cada encabezado del **segmento TCP** contiene un número de secuencia que permite que las funciones de la capa de transporte en el host de destino vuelvan a armar segmentos en el orden en que se transmitieron.
- UDP rastrea también las conversaciones entre las aplicaciones,pero no se encarga del orden en que se transmite la información ni de mantener una conexión. No existe número de secuencia en el encabezado de **un datagrama UDP**. UDP es un diseño simple y genera menos carga que TCP, lo que produce una transferencia de datos más rápida.



4. PROTOCOLOS TRANSPORTE

Separación de comunicaciones

- La capa de transporte debe poder **separar y administrar** varias **comunicaciones** con diferentes necesidades de requisitos de transporte.
- Para diferenciar los segmentos y datagramas para cada aplicación, tanto TCP como UDP cuentan con **campos de encabezado** que pueden **identificar** de manera exclusiva estas **aplicaciones**. Estos identificadores únicos son **números de puertos**.

4. PROTOCOLOS TRANSPORTE

Direccionamiento de puertos

- Un puerto es un identificador numérico de cada segmento, que se utiliza para realizar un seguimiento de conversaciones específicas y de servicios de destino solicitados. Cada mensaje que envía un host contiene.
- **Puerto de destino**. El cliente coloca un **número de puerto de destino** en el segmento para informar al servidor de destino el servicio solicitado.
- **Puerto de origen**. El número de **puerto de origen** es **generado de manera aleatoria** por el dispositivo emisor para identificar una conversación entre dos dispositivos. Esto permite establecer varias conversaciones simultáneamente.
  - El seguimiento de las conversaciones por separado se basa en los puertos de origen

4. PROTOCOLOS TRANSPORTE

Direccionamiento de puertos

- Los puertos de origen y de destino se colocan dentro del segmento. Los segmentos se encapsulan dentro de un paquete IP. El paquete IP contiene la dirección IP de origen y de destino. La **combinación de las direcciones IP de origen y de destino** y de los **números de puerto de origen y de destino** se conoce como **socket**.
- Un **par de sockets**, que consiste en las direcciones IP de origen y destino y los números de puertos, también es exclusivo e **identifica la conversación** específica entre los dos hosts.
- Los **sockets** permiten que los **procesos múltiples** que se ejecutan en un cliente **se distingan** entre sí.

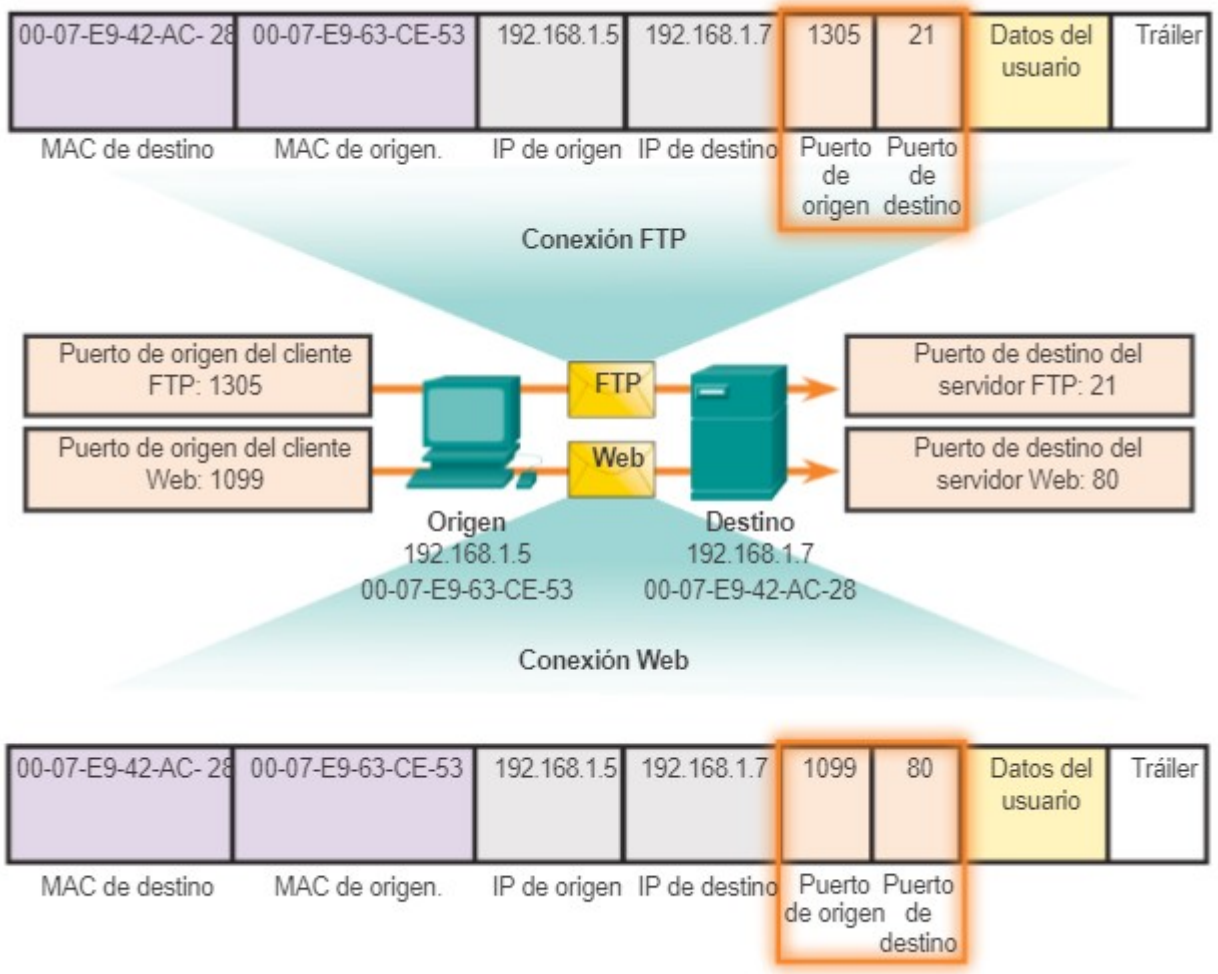
4. PROTOCOLOS TRANSPORTE

Direccionamiento de puertos

- El **puerto de origen** de la solicitud de un cliente se **genera de manera aleatoria**. El número de puerto actúa como **dirección de retorno** para la aplicación que realiza la solicitud.
- La capa de transporte hace un seguimiento de este puerto y de la aplicación que generó la solicitud de manera que cuando se devuelva una respuesta, esta se envíe a la aplicación correcta.
- El **número de puerto** de la aplicación que realiza la **solicitud** se utiliza como **número de puerto de destino** en la **respuesta** que vuelve del servidor.

4. PROTOCOLOS TRANSPORTE

Direccionamiento de puertos



4. PROTOCOLOS TRANSPORTE

Direccionamiento de puertos

- La Agencia de asignación de números por Internet (**IANA**) **asigna** números de **puerto**.
- Existen diferentes tipos de números de puerto:
  - **Puertos bien conocidos (números del 0 al 1023)**: estos números se reservan para servicios y aplicaciones.
  - **Puertos registrados (números del 1024 al 49151)**: estos números de puerto se asignan a procesos o aplicaciones del usuario.
  - **Puertos dinámicos o privados (números 49152 a 65535)**: también conocidos como puertos efímeros, generalmente se los asigna de forma dinámica a las aplicaciones cliente cuando el cliente inicia una conexión a un servicio.

4. PROTOCOLOS TRANSPORTE

Direccionamiento de puertos

- **Algunas aplicaciones** pueden **utilizar** tanto **TCP** como **UDP**.
  - Por ejemplo, el bajo gasto de UDP permite que DNS atienda rápidamente varias solicitudes de clientes. Sin embargo, a veces el envío de la información solicitada puede requerir la confiabilidad de TCP. En este caso, el número de puerto bien conocido (53) lo utilizan ambos protocolos con este servicio.

Rango de números de puerto	Grupo de puertos
Entre 0 y 1023	Puertos bien conocidos
de 1024 a 49151	Puertos registrados
de 49152 a 65535	Puertos privados y/o dinámicos

4. PROTOCOLOS TRANSPORTE

Elección del protocolo de transporte

- Tanto **TCP como UDP** son protocolos de transporte **válidos**. Según los requisitos de la aplicación, se puede utilizar uno u otro, o se pueden utilizar ambos.
- Para algunas aplicaciones, los **segmentos** deben llegar en una **secuencia muy específica** para que se puedan procesar correctamente, o puede que todos los **datos** se deben **recibir** en **forma completa** para poder considerarse útiles. En estos casos, se utiliza **TCP.**
  - Por ejemplo, las aplicaciones, como las bases de datos, los exploradores Web y los clientes de correo electrónico.

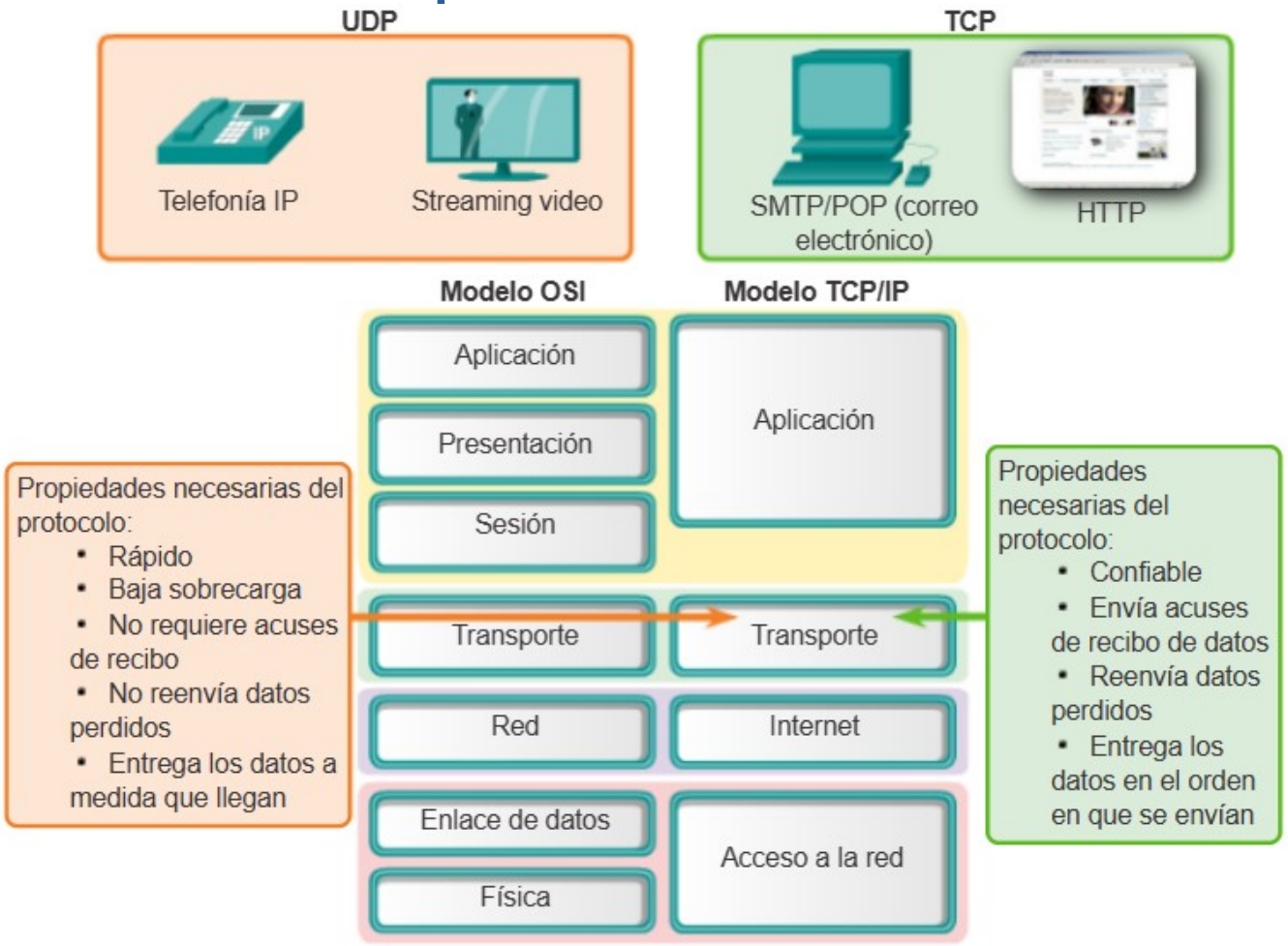
4. PROTOCOLOS TRANSPORTE

Elección del protocolo de transporte

- En otros casos, una aplicación puede **tolerar cierta pérdida de datos** durante la transmisión a través de la red, pero **no se admiten retrasos** en la transmisión. **UDP** es la mejor opción para estas aplicaciones, ya que se requiere menos sobrecarga de red. Los acuses de recibo reducirían la velocidad de la entrega, y las retransmisiones no son recomendables.
  - Con aplicaciones como streaming audio, video y voz sobre IP (VoIP), es preferible utilizar UDP.

4. PROTOCOLOS TRANSPORTE

- **Elección del protocolo de transporte**



4. PROTOCOLO TRANSPORTE
4.1 TCP

- Además de admitir **funciones básicas de segmentación y rearmado** de datos, **TCP** proporciona además
  - **Conversaciones orientadas a la conexión** mediante el establecimiento de **sesiones.**
    - TCP **negocia y establece** una conexión (o **sesión**) permanente entre los dispositivos de origen y de destino antes de reenviar tráfico, se **termina** solo cuando se completa toda la comunicación.
  - **Entrega confiable.**
    - TCP puede asegurar que todas las partes lleguen a destino al hacer que el dispositivo de origen **retransmita** los **datos perdidos o dañados**.
  - **Reconstrucción de datos ordenada**
    - Los datos pueden llegar en el orden equivocado, debido a que las redes pueden proporcionar varias rutas. Al **numerar y secuenciar los segmentos**, TCP puede asegurar que estos se rearmen en el orden correcto.
  - **Control del flujo**
    - Los hosts de la red cuentan con recursos limitados, como memoria o ancho de banda. Cuando TCP advierte que estos recursos están sobrecargados, puede solicitar que la aplicación emisora reduzca la velocidad del flujo de datos. El **control de flujo** puede evitar la pérdida de segmentos en la red y evitar la necesitad de la retransmisión.

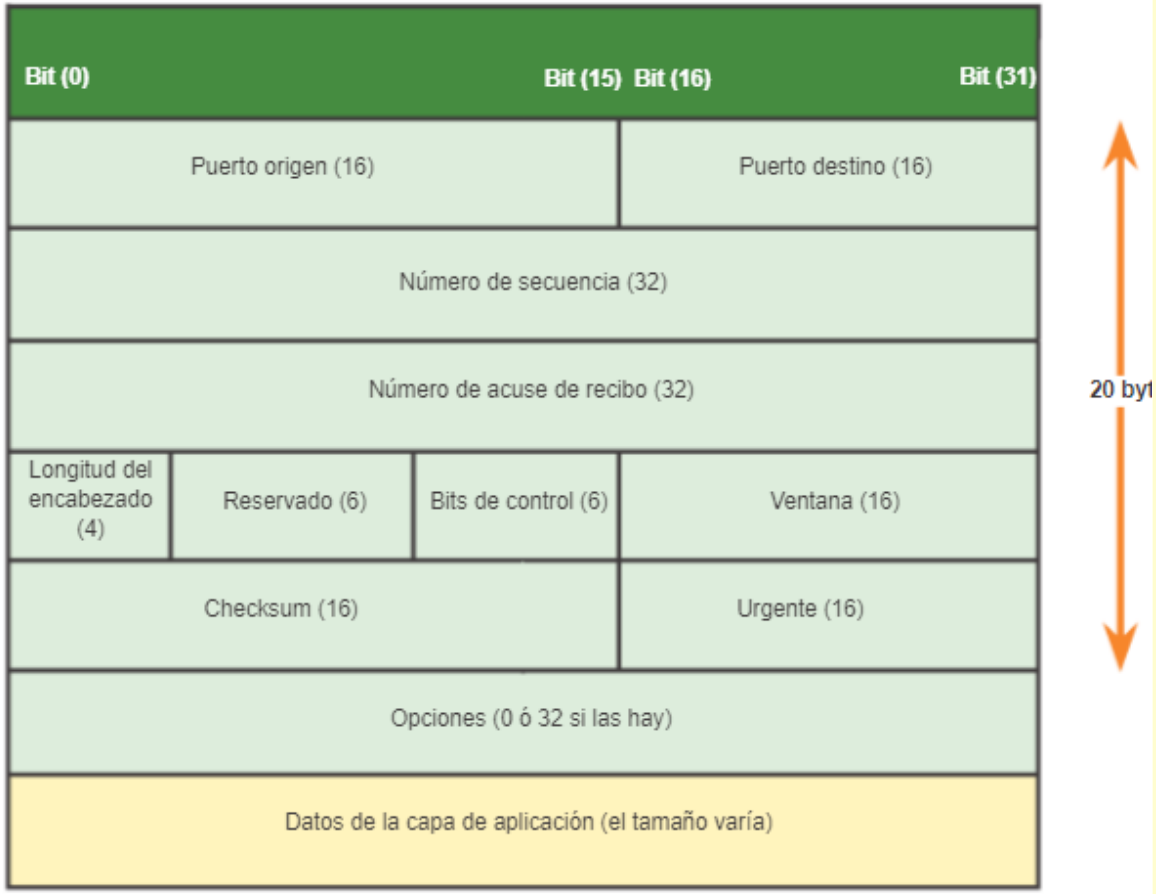
4. PROTOCOLO TRANSPORTE
4.1 TCP

Confiabilidad TCP

- **TCP** se considera un **protocolo de transporte confiable**, lo que significa que incluye procesos para **garantizar la entrega confiable** entre aplicaciones.
- Con TCP, las tres **operaciones básicas de confiabilidad** son las siguientes:
  - **Seguimiento de segmentos** de datos transmitidos.
    - TCP realiza un seguimiento del número de segmentos que se enviaron a un host específico desde una aplicación específica.
  - **Acuse de recibo** de datos.
    - Garantiza la entrega confiable entre aplicaciones mediante el uso de entrega con acuse de recibo.
  - **Retransmisión** de cualquier dato **sin acuse de recibo**
    - Si el emisor no recibe un acuse de recibo antes del transcurso de un período determinado, supone que los segmentos se perdieron y los vuelve a transmitir.

4. PROTOCOLO TRANSPORTE
4.1 TCP

- **Estructura segmento TCP**
  - **TCP** genera **sobrecarga de datos adicional** para obtener estas funciones. Cada segmento **TCP** tiene **20 Bytes** de sobrecarga en el **encabezado** que encapsula los datos de la capa de aplicación.
  - Este tipo de segmento es **mucho más largo** que un segmento **UDP**, que solo tiene 8 B de sobrecarga.



4. PROTOCOLO TRANSPORTE
4.1 TCP

Comunicación TCP

- La **diferencia** clave entre **TCP** y **UDP** es la **confiabilidad**. La **confiabilidad** de la comunicación TCP se obtiene con el uso de **sesiones orientadas a la conexión**.
  - Antes de que un host que utiliza TCP envíe datos a otro host, TCP inicia un proceso para crear una conexión con el destino.
- Una vez que se establece una sesión y que comienza la **transferencia de datos**, el **destino envía acuses de recibo al origen** por los segmentos que recibe, los cuales forman la base de la confiabilidad dentro de la sesión TCP.
  - Cuando el origen recibe un acuse de recibo, reconoce que los datos se entregaron correctamente y puede dejar de rastrearlos. Si el origen no recibe el acuse de recibo dentro de un tiempo predeterminado, retransmite esos datos al destino.

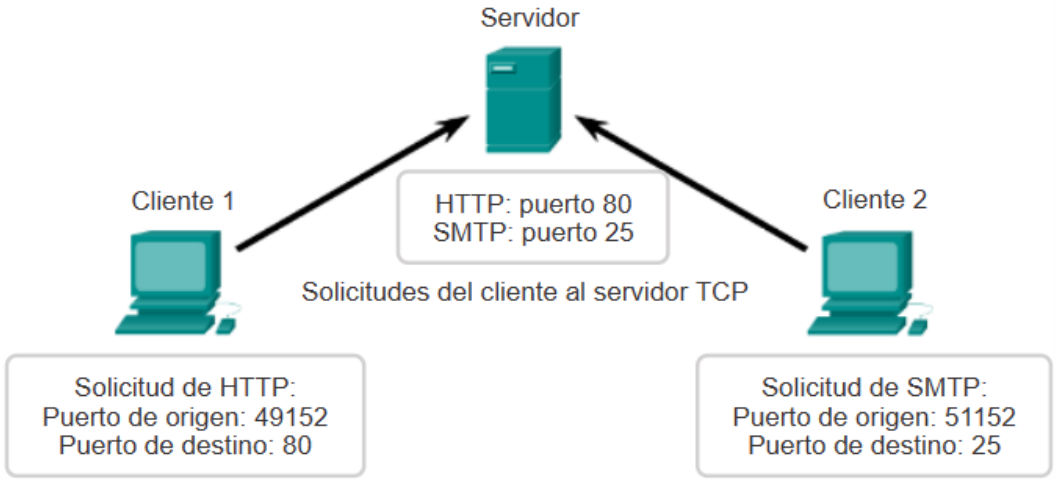
4. PROTOCOLO TRANSPORTE
4.1 TCP

Comunicación TCP

- Parte de la **carga adicional** que genera el uso de **TCP** es el tráfico de red generado por los **acuses de recibo** y las **retransmisiones**.
  - El establecimiento de las sesiones genera sobrecarga en forma de segmentos adicionales que se intercambian. Hay también sobrecarga en los hosts individuales creada por la necesidad de mantener un registro de los segmentos que esperan un acuse de recibo y por el proceso de retransmisión.

Comunicación TCP

- Los **procesos de las aplicaciones** se ejecutan en los **servidores**. Un único servidor puede ejecutar varios procesos de aplicaciones al mismo tiempo.
- Cada **proceso de aplicación** que se ejecuta en el servidor se configura para utilizar un **número de puerto**, ya sea predeterminado o de forma manual por el administrador del sistema.
- Una manera de **mejorar la seguridad** en un servidor es **restringir** el acceso al servidor únicamente a aquellos **puertos relacionados** con los servicios y las aplicaciones a los que deben poder acceder los solicitantes autorizados.



Comunicación TCP

- Cuando **dos hosts se comunican** utilizando **TCP**, se **establece** una **conexión** antes de que puedan intercambiarse los datos. Luego de que se completa la comunicación, se cierran las sesiones y la conexión finaliza.
- Para establecer la conexión los hosts realizan un **protocolo de enlace de tres vías**.
- Los hosts hacen un **seguimiento de cada segmento** de datos dentro de una sesión e intercambian información sobre qué datos se reciben mediante la información en os **bits de control** del **encabezado TCP**.

4. PROTOCOLO TRANSPORTE
4.1 TCP

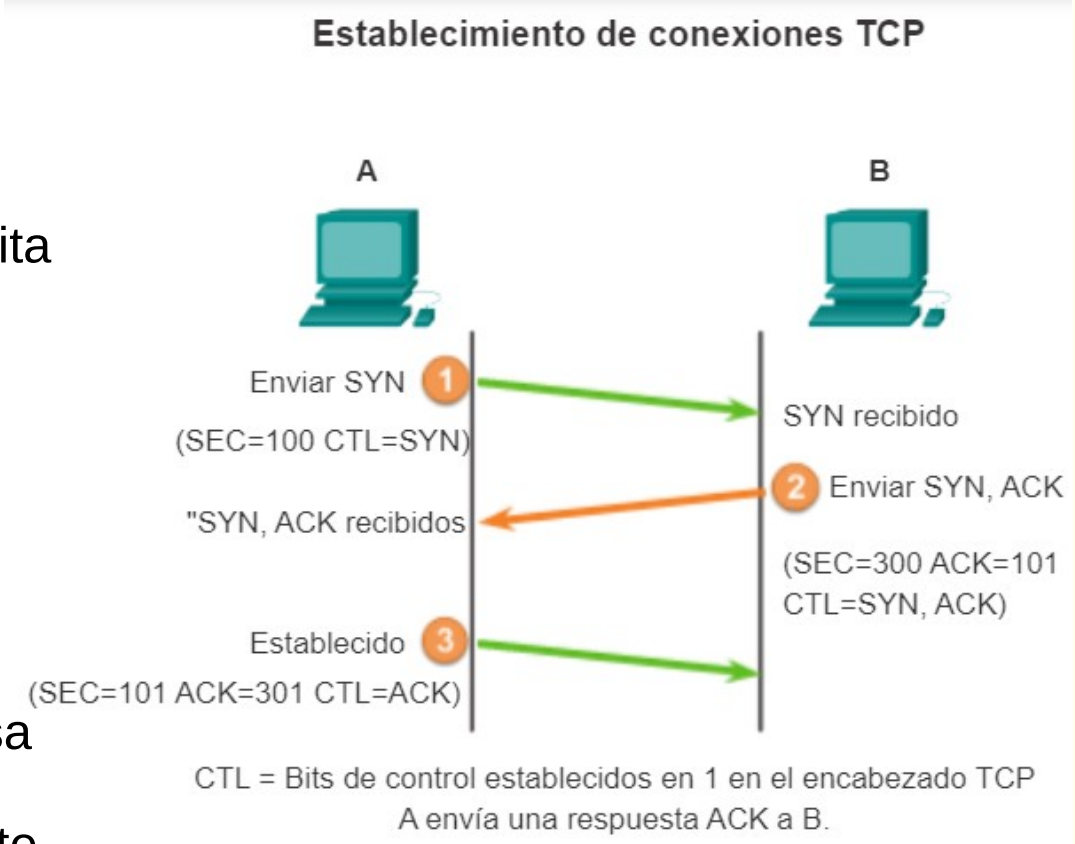
Comunicación TCP

- Dentro del **encabezado del segmento TCP**, existen **seis campos de 1** bit que contienen **información de control** utilizada para gestionar los procesos de TCP. Estos campos son los siguientes:
  - **URG**: campo indicador urgente importante
  - **ACK**: campo de acuse de recibo importante
  - **PSH**: función de empuje
  - **RST**: restablecer la conexión
  - **SYN**: sincronizar números de secuencia
  - **FIN**: no hay más datos del emisor
- Los campos **ACK y SYN** son importantes para el análisis del **protocolo de enlace de tres vías**.

4. PROTOCOLO TRANSPORTE
4.1 TCP

Comunicación TCP

- Los **tres pasos** en el **establecimiento** de una **conexión TCP** son:
  - **Paso 1**. El cliente de origen solicita una sesión de comunicación de cliente a servidor con el servidor.
  - **Paso 2**. El servidor acusa recibo de la sesión de comunicación de cliente a servidor y solicita una sesión de comunicación de servidor a cliente.
  - **Paso 3**. El cliente de origen acusa recibo de la sesión de comunicación de servidor a cliente.



4. PROTOCOLO TRANSPORTE
4.1 TCP

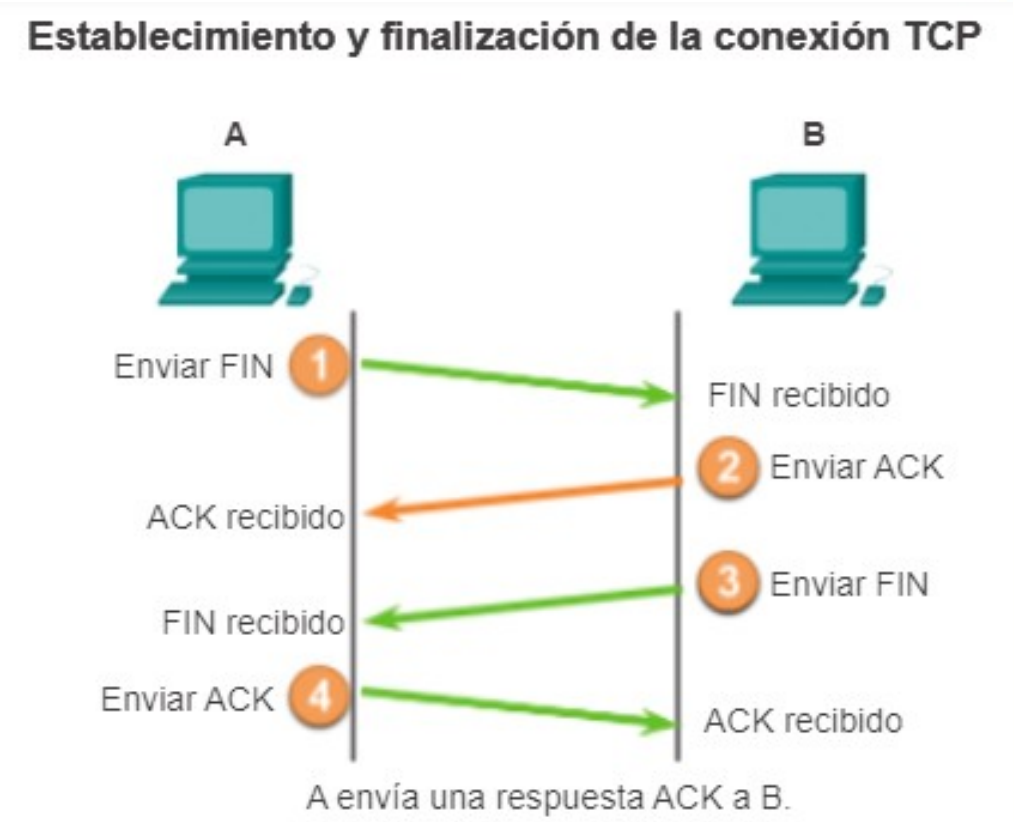
Comunicación TCP

- Para **cerrar una conexión**, se debe establecer el indicador de control **finalizar** (**FIN**) en el **encabezado del segmento**. Para finalizar todas las sesiones TCP de una vía, se utiliza un enlace de dos vías, que consta de un **segmento FIN** y un **segmento ACK**.
- Por lo tanto, para **terminar una única conversación** que admite TCP, se requieren **cuatro intercambios** para finalizar ambas sesiones: de cliente a servidor y de servidor a cliente.

4. PROTOCOLO TRANSPORTE
4.1 TCP

Comunicación TCP

- Por lo tanto, para **terminar una única conversación** que admite **TCP**, se requieren **cuatro intercambios** para finalizar ambas sesiones
  - **Paso 1**: cuando el cliente no tiene más datos para enviar en el stream, envía un segmento con el indicador FIN establecido.
  - **Paso 2**: el servidor envía un ACK para acusar recibo del FIN y terminar la sesión de cliente a servidor.
  - **Paso 3**: el servidor envía un FIN al cliente para terminar la sesión de servidor a cliente.
  - **Paso 4**: el cliente responde con un ACK para dar acuse de recibo del FIN desde el servidor.



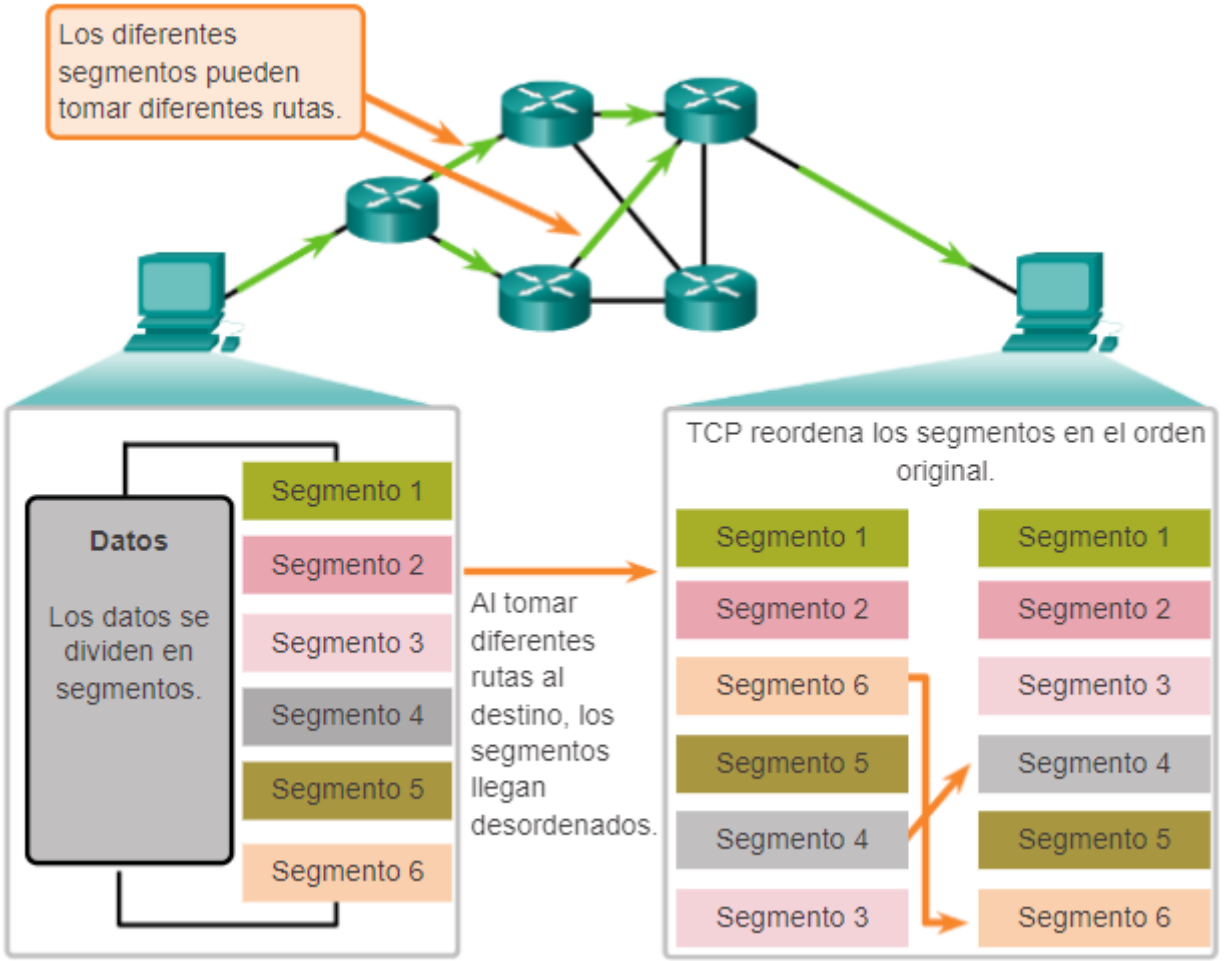
4. PROTOCOLO TRANSPORTE
4.1 TCP

Reordenamiento de segmentos

- Cuando los servicios envían datos mediante el TCP, los **segmentos** pueden llegar a su **destino** en **desorden**. En el **receptor**, estos segmentos se **reensamblan en el orden original**. Para lograr esto, se asignan **números de secuencia** en el **encabezado** de cada paquete.
- Durante la **configuración de la sesión**, se establece un **número de secuencia inicial (ISN)**. A medida que se transmiten los datos durante la sesión, el número de secuencia se incrementa en el número de bytes que se han transmitido.
- Este seguimiento de bytes de datos permite **identificar y dar acuse de recibo** de cada segmento de manera exclusiva. Se pueden **identificar segmentos perdidos**.

4. PROTOCOLO TRANSPORTE
4.1 TCP

- **Reordenamiento de segmentos**

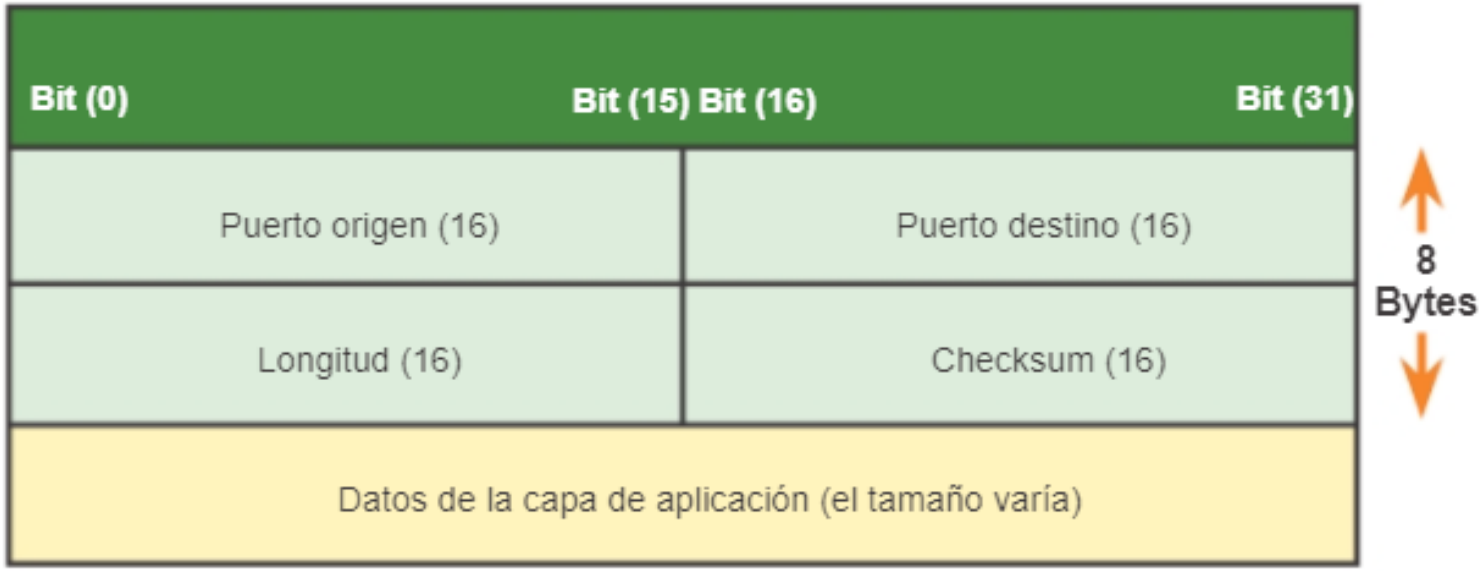


4. PROTOCOLO TRANSPORTE
4.2 UDP

- Aunque **UDP no incluye** la **confiabilidad** y los mecanismos de **control del flujo** de TCP, la entrega de datos de **baja sobrecarga** de UDP lo convierte en un protocolo de transporte ideal para las aplicaciones que pueden tolerar cierta pérdida de datos.
- Las porciones de comunicación en **UDP** se llaman **datagramas**.
  - Algunas aplicaciones que utilizan UDP son DNS o VoIP.
- **UDP** es un protocolo **sin estado**, lo cual significa que ni el cliente ni el servidor están obligados a hacer un seguimiento del estado de la sesión de comunicación

4. PROTOCOLOS TRANSPORTE
4.2 UDP

- **Estructura datagrama UDP**
  - UDP no se ocupa de la confiabilidad ni del control del flujo. Los datos se pueden perder o recibir fuera de secuencia sin ningún mecanismo de UDP que pueda recuperarlos o reordenarlos



4. PROTOCOLO TCP
4.2 UDP

Comunicación UDP

- Tiene una **sobrecarga mucho menor** que TCP, ya que **no está orientado a la conexión** y **no proporciona** los mecanismos sofisticados de **retransmisión, secuenciación y control del flujo** que ofrecen confiabilidad.
- La baja sobrecarga del UDP es deseada por dichas **aplicaciones**:
  - Sistema de nombres de dominio (**DNS**)
  - Protocolo simple de administración de red (**SNMP**, Simple Network Management Protocol)
  - Protocolo de configuración dinámica de host (**DHCP**)
  - Protocolo de información de enrutamiento (**RIP**)
  - Protocolo de transferencia de archivos trivial (**TFTP**)
  - Telefonía IP o voz sobre IP (**VoIP**)
  - Juegos en línea
  - etc.

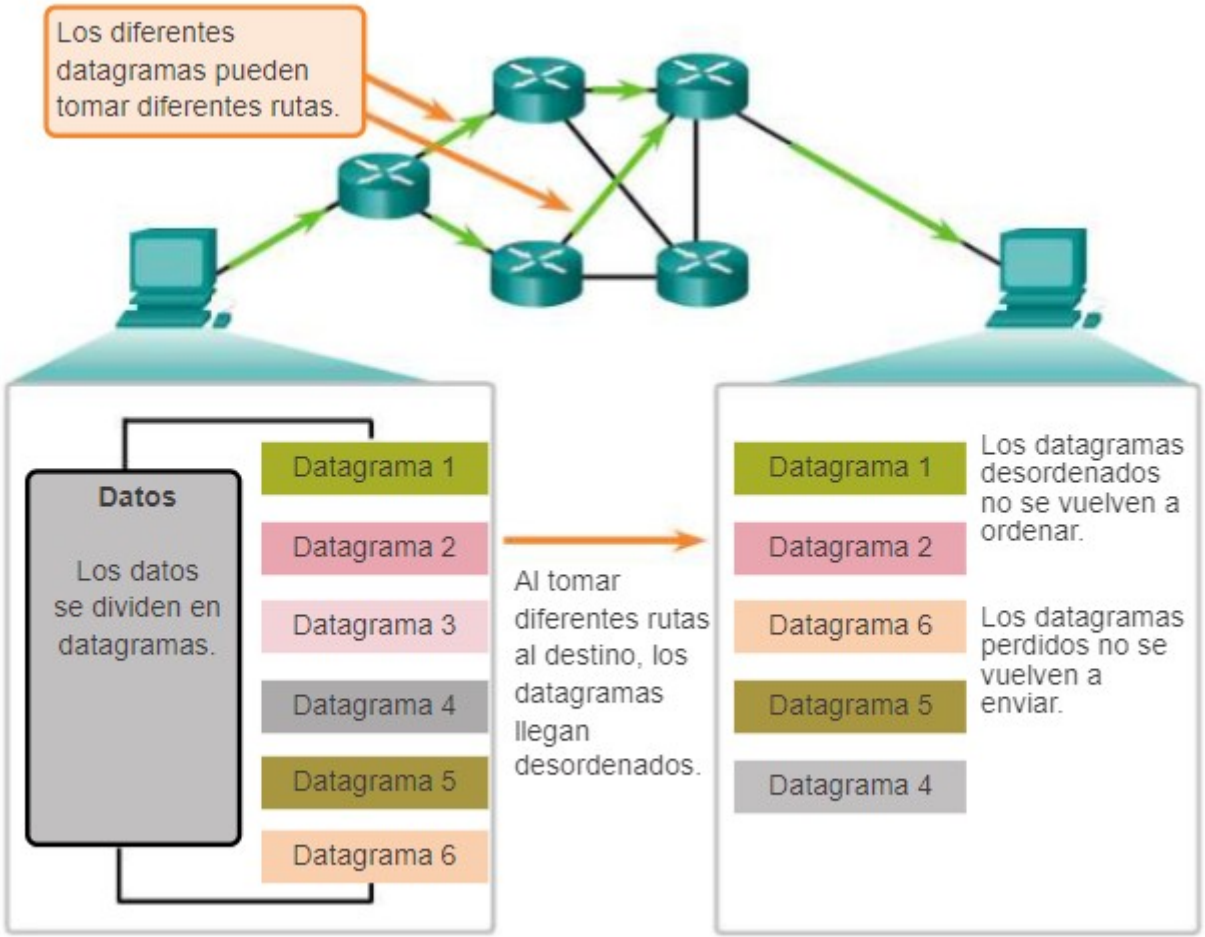
4. PROTOCOLO TCP
4.2 UDP

Comunicación UDP

- Ya que **UDP** **opera** **sin conexión**, las sesiones no se establecen antes de que se lleve a cabo la comunicación, como sucede con TCP. Se dice que UDP está **basado en** las **transacciones**; es decir, cuando una aplicación tiene datos para enviar, simplemente los envía.
- Cuando se envían **datagramas múltiples** a un destino, pueden tomar diferentes rutas y llegar en el orden equivocado. **UDP no realiza un seguimiento** de los números de secuencia de la manera en que lo hace TCP, por lo que **UDP no tiene forma de reordenarlos**.
- Por lo tanto, UDP simplemente **reensambla los datos** en el orden en que se recibieron y **los envía a la aplicación**.

4. PROTOCOLO TCP
4.2 UDP

- **Comunicación UDP**



5. PROTOCOLOS ACCESO A RED
4.2 UDP

Comunicación UDP

- Ya que **UDP** **opera** **sin conexión**, las sesiones no se establecen antes de que se lleve a cabo la comunicación, como sucede con TCP. Se dice que UDP está **basado en** las **transacciones**; es decir, cuando una aplicación tiene datos para enviar, simplemente los envía.
- Cuando se envían **datagramas múltiples** a un destino, pueden tomar diferentes rutas y llegar en el orden equivocado. **UDP no realiza un seguimiento** de los números de secuencia de la manera en que lo hace TCP, por lo que **UDP no tiene forma de reordenarlos**.
- Por lo tanto, UDP simplemente **reensambla los datos** en el orden en que se recibieron y **los envía a la aplicación**.