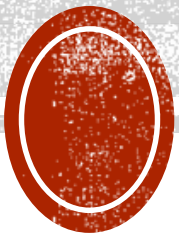
NIVEL DE TRANSPORTE



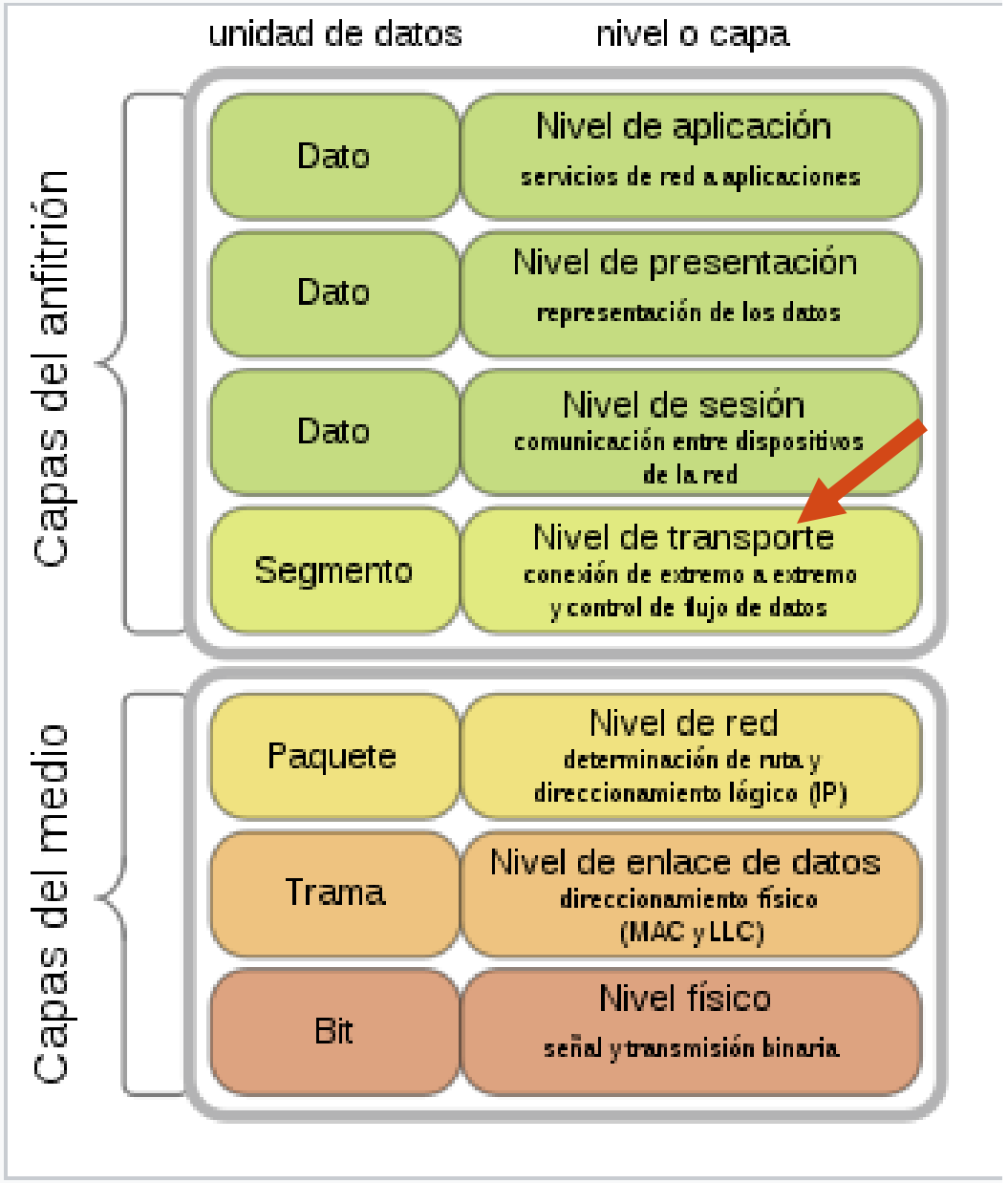
CONTENIDOS

- Introducción
  - Descripción general
  - Funciones
  - Los puertos: qué son y tipos. Ejemplo
  - Los socket: qué son. Ejemplo
  - Protocolos de capa de transporte
- Protocolo de Control de Transmisión (TCP)
  - Introducción
  - Funciones
  - Formato segmento TCP
  - Aplicaciones usan TCP
  - Establecimiento de conexión TCP
    - Fases
    - Estados de la conexión
    - Campos de bit de control
- Protocolo de Datagramas de Usuario (UDP)
  - Introducción
  - Características
  - Funcionamiento
  - Formato segmento UDP
  - Aplicaciones usan UDP
- UDP vs TCP

INTRODUCCIÓN

DONDE ESTAMOS

- Capa 4 del modelo OSI



DESCRIPCIÓN GENERAL

- En el **emisor** recibe el mensaje procedente del nivel de aplicación, lo divide en segmentos y los entrega a la capa o nivel de red, asegurándose de que lleguen al otro extremo.
- En el **receptor** se ensamblan todos los segmentos para formar de nuevo el mensaje y entregarlo libre de errores a la capa o nivel de aplicación
- El **nivel de transporte ocurre exclusivamente en los equipos terminales** y no en dispositivos de interconexión que solamente tienen niveles de red, enlace y físico.
- Los protocolos de transporte se implementan en el sistema operativo de los equipos terminales.

DESCRIPCIÓN GENERAL

- La capa de transporte es donde, como su nombre indica, los datos se transportan de un host a otro.
- La capa de transporte no tiene conocimiento del tipo de host de destino, el tipo de medio por el que deben viajar los datos, la ruta tomada por los datos, la congestión en un enlace o el tamaño de la red.
- La capa de transporte utiliza dos protocolos: TCP y UDP.
  - TCP podría asemejarse a recibir una carta certificada. Tienes que firmar antes de que el transportista de correo te la entregue. Esto ralentiza un poco el proceso, pero el remitente sabe con certeza que recibió la carta y cuándo la recibió.
  - UDP es como una carta común con un sello. El remitente no puede estar seguro de que has recibido la carta. Hay momentos en que UDP, como una carta, es el protocolo que se necesita.

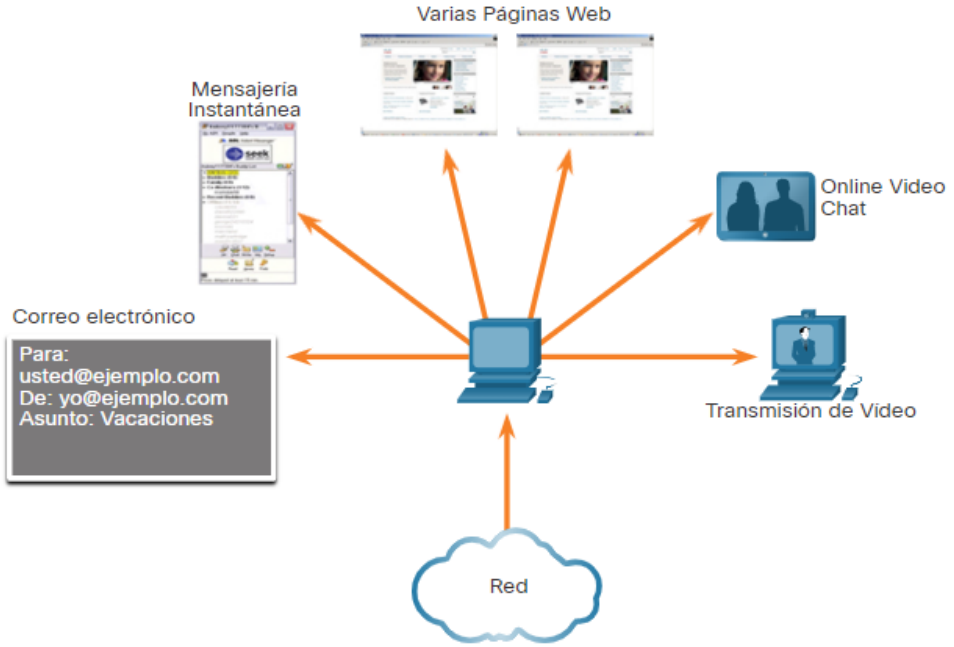
FUNCIONES

- La **capa de transporte** tiene las siguientes **responsabilidades**:

- Seguimiento de conversaciones individuales
- Segmentación de Datos y Rearmado de Segmentos
- Agregar Información de Encabezado
- Identificación de las Aplicaciones
- Multiplexión de Conversaciones
- Control de errores

SEGUIMIENTO DE CONVERSACIONES INDIVIDUALES

- Un host puede tener **múltiples aplicaciones que se comunican** a través de la red simultáneamente.
- En la capa de transporte, cada conjunto de datos que fluye entre una aplicación de origen y una aplicación de destino se conoce como una **conversación** y se rastrea por separado. Es responsabilidad de la capa de transporte mantener y hacer un seguimiento de todas estas conversaciones.



- La entrega de paquetes se realiza desde un proceso en el equipo emisor, hasta otro proceso en el quipo receptor, usando **puertos**
- Esto permite varias **conexiones simultáneas** en el mismo equipo, por eso es posible abrir varios navegadores y a su vez varias pestañas, todas funcionando a la vez

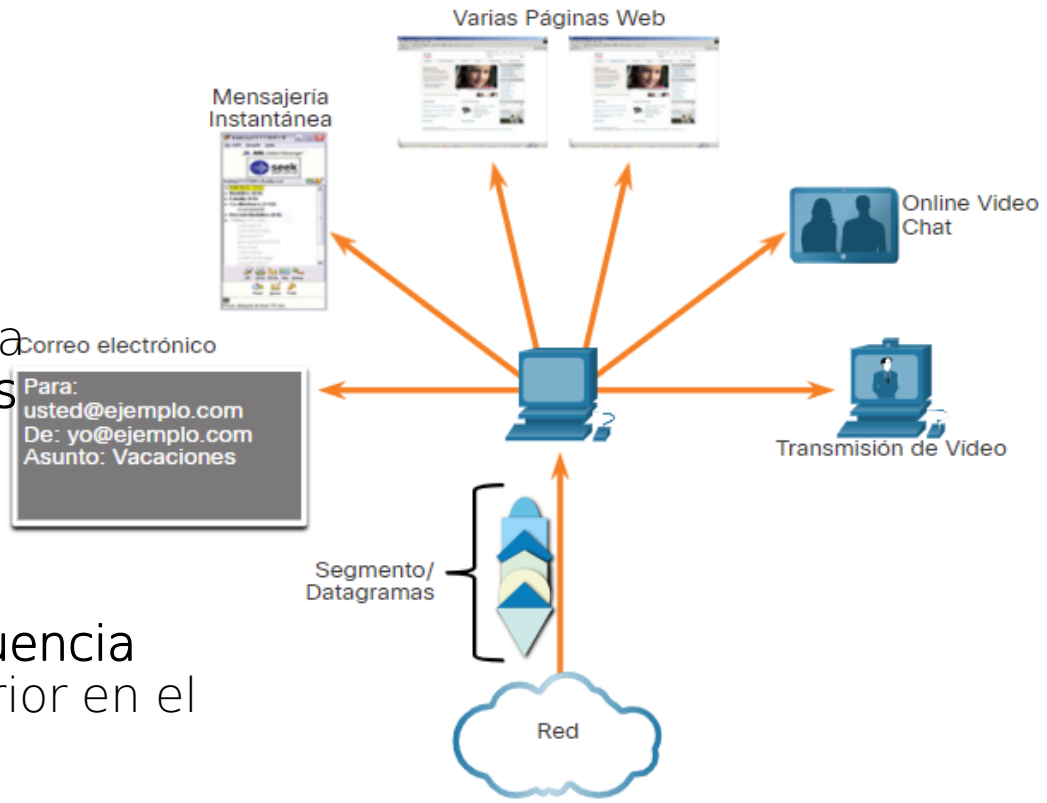
SEGMENTACIÓN DE DATOS Y REARMADO DE SEGMENTOS

- Es responsabilidad de la capa de transporte **dividir los datos** de la aplicación en **bloques** de **tamaño adecuado**.

- **Dependiendo del protocolo** de capa de transporte utilizado, los **bloques** de capa de transporte se denominan **segmentos o datagramas**.

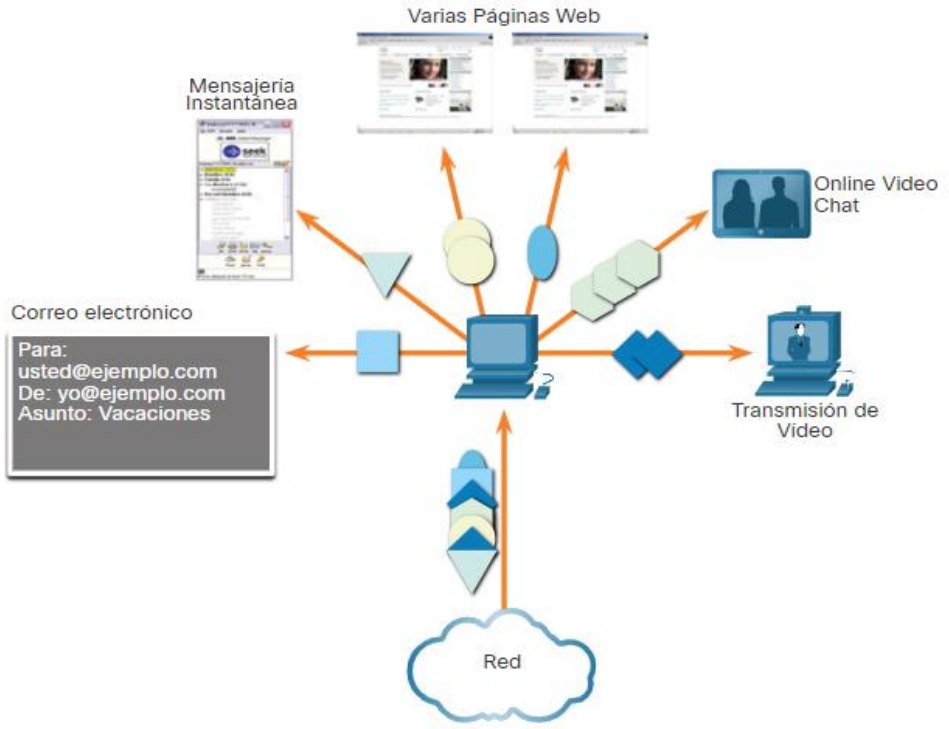
- **Cada segmento** lleva un **nº de secuencia** para permitir el ensamblado posterior en el receptor.

- El **último segmento** lleva un **indicador de fin de secuencia**



AGREGAR INFORMACIÓN DE ENCABEZADO

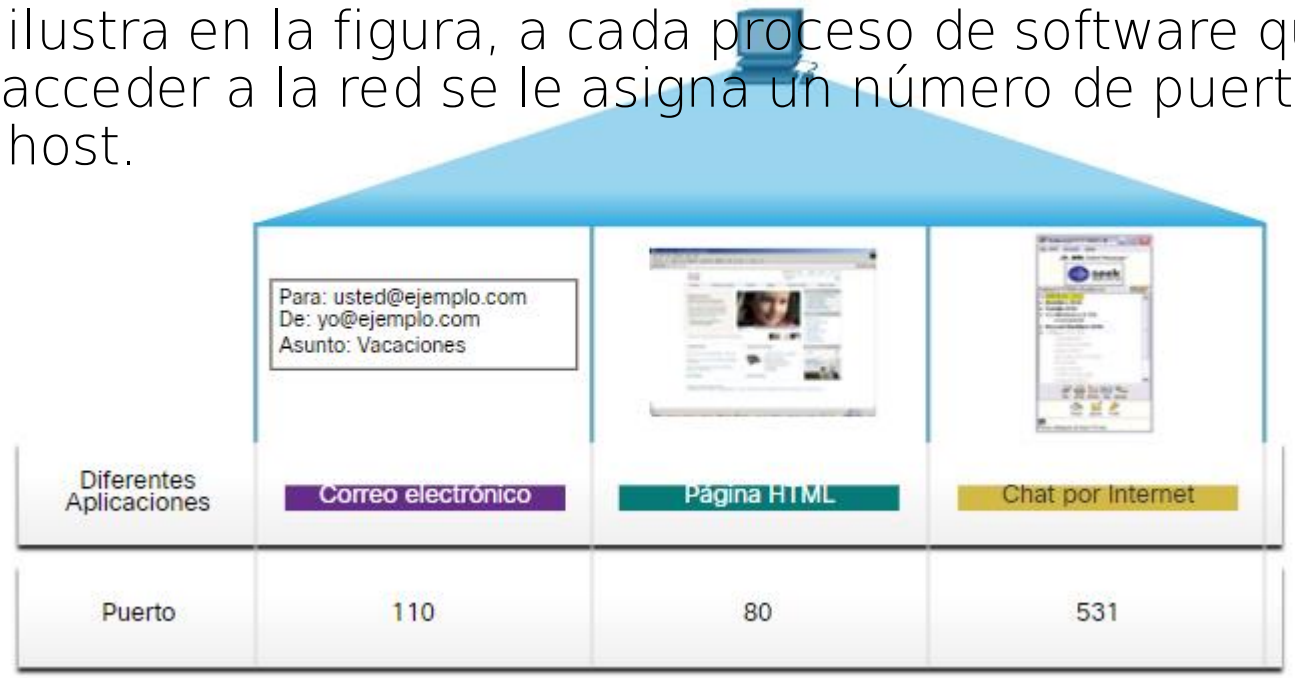
- El protocolo de capa de transporte también **agrega información de encabezado que contiene datos binarios organizados en varios campos a cada bloque de datos**. Los valores de estos campos permiten que los distintos protocolos de la capa de transporte lleven a cabo variadas funciones de administración de la comunicación de datos.
- Por ejemplo, el **host receptor utiliza la información de encabezado para volver a ensamblar los bloques de datos** en un flujo de datos completo para el programa de capa de aplicación de recepción.



- La capa de transporte garantiza que incluso con múltiples aplicaciones que se ejecutan en un dispositivo, todas las

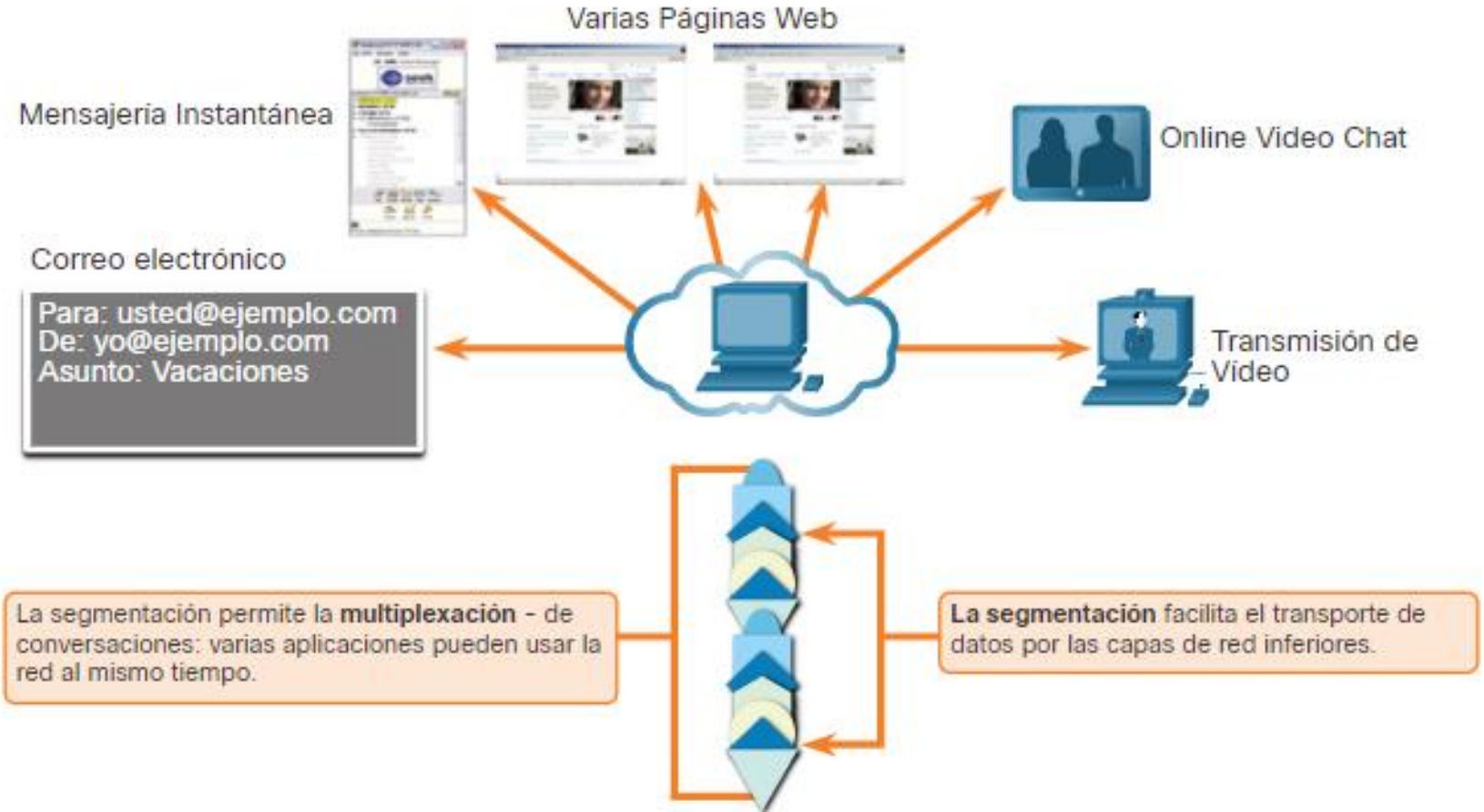
IDENTIFICACIÓN DE LAS APLICACIONES

- La capa de transporte debe poder separar y **administrar varias comunicaciones** con diferentes necesidades de requisitos de transporte.
- Para **pasar flujos de datos a las aplicaciones adecuadas**, la capa de transporte identifica la aplicación de destino utilizando un identificador llamado número de **puerto**.
- Como se ilustra en la figura, a cada proceso de software que necesita acceder a la red se le asigna un número de puerto único para ese host.



MULTIPLEXIÓN DE CONVERSACIONES

- Como se muestra en la figura, la capa de transporte utiliza segmentación y multiplexación para permitir que diferentes conversaciones de comunicación se intercalen en la misma red.



CONTROL DE ERRORES

- La verificación de errores se puede realizar en los datos del segmento, para determinar si el segmento se modificó durante la transmisión.
- Permite realizar comunicaciones orientadas a la conexión
- En estas comunicaciones, se establece un camino al inicio y todos los segmentos del mismo mensaje recorren dicho camino
- Una vez que se han enviado todos los datos se libera la conexión

COMUNICACIONES MÚLTIPLES SEPARADAS

- El nivel de red utiliza solamente direcciones IP para identificar los paquetes que envía a través de la red.
- Sin embargo, el nivel de transporte añade el puerto, para distinguir entre los posibles procesos que pueden enviar o recibir datos dentro de un mismo host

- El número de puerto de origen está asociado con la aplicación de origen en el host local, mientras que el número de puerto de destino está asociado con la aplicación de destino en el host remoto.

¿QUÉ ES UN PUERTO?

Un puerto es un número de 16 bits que identifica un proceso local o remoto

¿Cuántos puertos puedo tener?

¿QUÉ ES UN PUERTO?

Un puerto es un número de 16 bits que identifica un proceso local o remoto

- Como $2^{16} = 65.536$ el rango de puertos va desde 0 a 65.536
- Estos puertos, a su vez están divididos en tres grupos:
  - Los puertos bien conocidos
  - Los puertos registrados
  - Los puertos dinámicos / privados

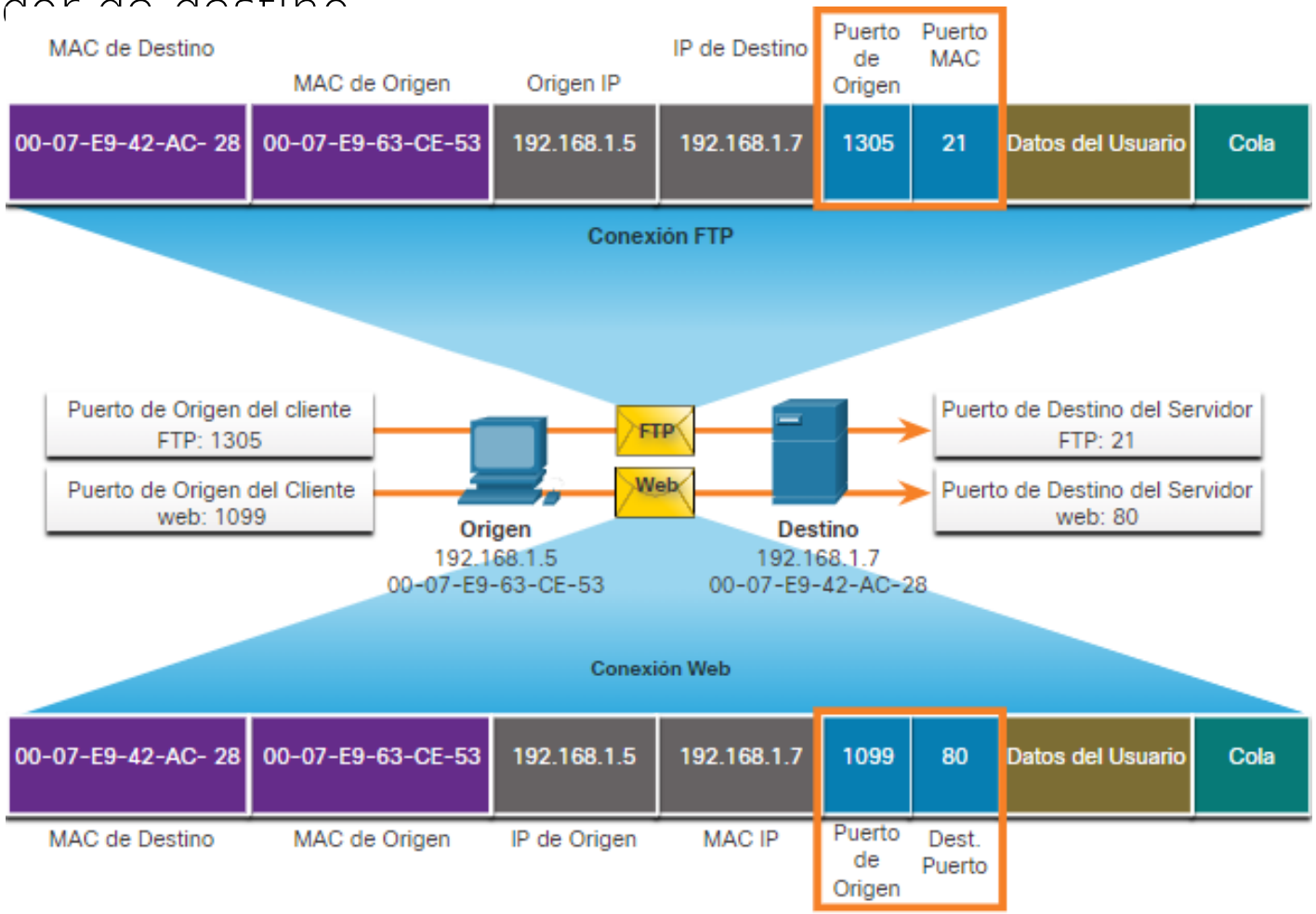
LOS PUERTOS

- Total de 65.536 puertos

Puertos bien conocidos (rango del 0 al 1023)	Puertos registrados (rango del 1024 al 49151)	Puertos dinámicos/privados (rango del 49152 al 65535)
• Servicios de red registrados por ICANN • FTP (20 y 21) • SSH (22) • TELNET (23) • SMTP (25) • DNS (53) • HTTP (80) • POP3 (110) • IMAP (143) • HTTPS(443) • …	• Empleados por aplicaciones de usuario de forma temporal • Servicios de red registrados por otras entidades • MS SQL Server (1433) • Oracle (1525) • MySQL (3306) • …	• Son empleados por aplicaciones de usuario de forma temporal. • Cuando se termina comunicación se liberan

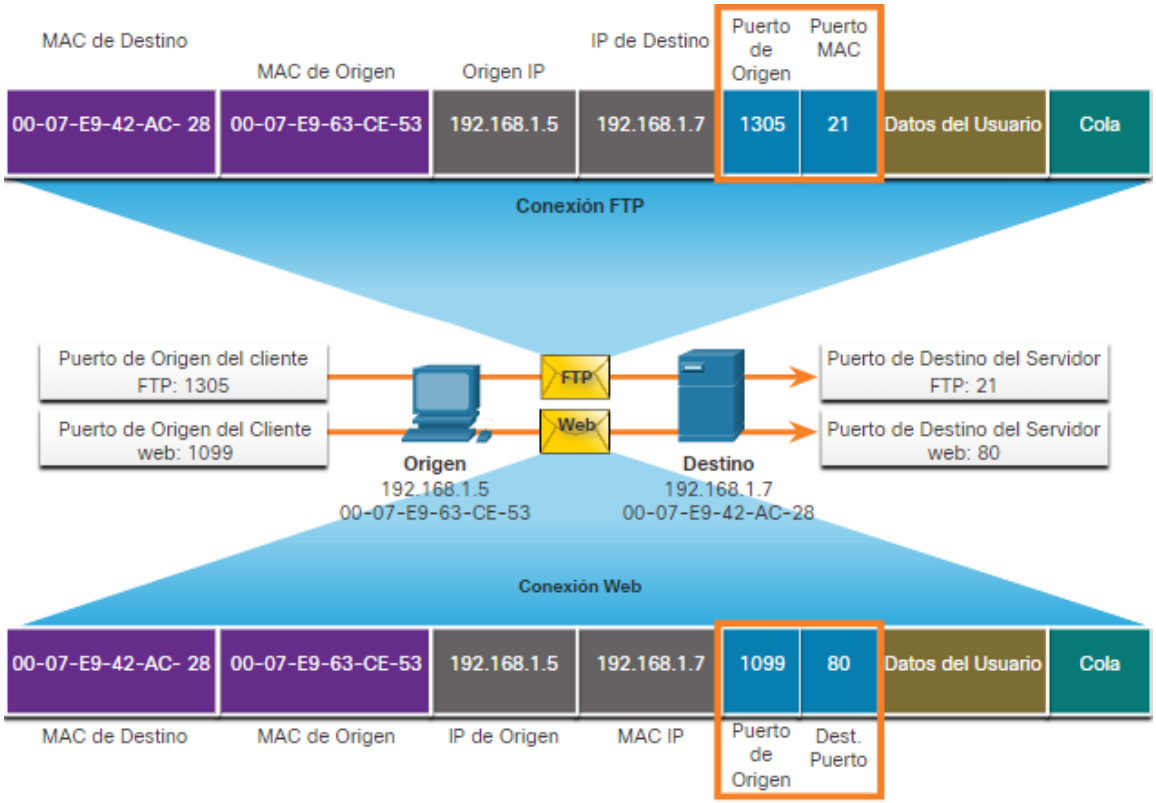
EJEMPLO

- El PC está solicitando simultáneamente servicios FTP y web desde el servidor de destino



EJEMPLO

- La solicitud FTP generada por el PC incluye las direcciones MAC de Capa 2 y las direcciones IP de Capa 3. La solicitud también identifica el puerto de origen 1305 (es decir, generado dinámicamente por el host) y el puerto de destino, identificando los servicios FTP en el puerto 21.
- El host también ha solicitado una página web del servidor utilizando las mismas direcciones de Capa 2 y Capa 3. Sin embargo, está utilizando el número de puerto de origen 1099 (es decir, generado dinámicamente por el host) y el puerto de destino que



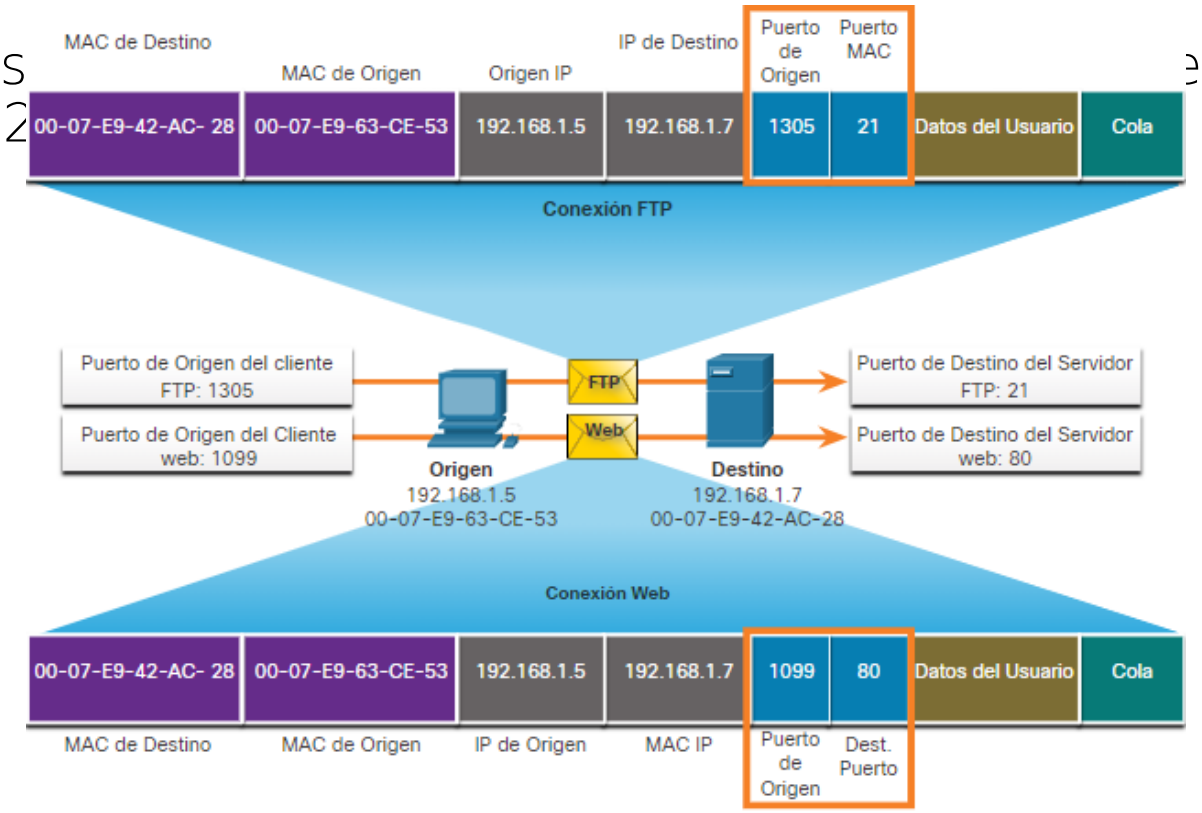
¿QUÉ ES UN SOCKET?

Un socket es un punto de comunicación por el cual un proceso puede emitir o recibir información.

- Un socket se identifica por la pareja "dirección ip: puerto".
- Ejemplo: 40.67.252.206:443
- Cada socket tiene asociado un espacio de memoria intermedia (buffer) para almacenar los datos que se van a enviar o recibir
- Para establecer la conexión es necesario un socket local y un socket remoto para trabajar de forma conjunta

EJEMPLO

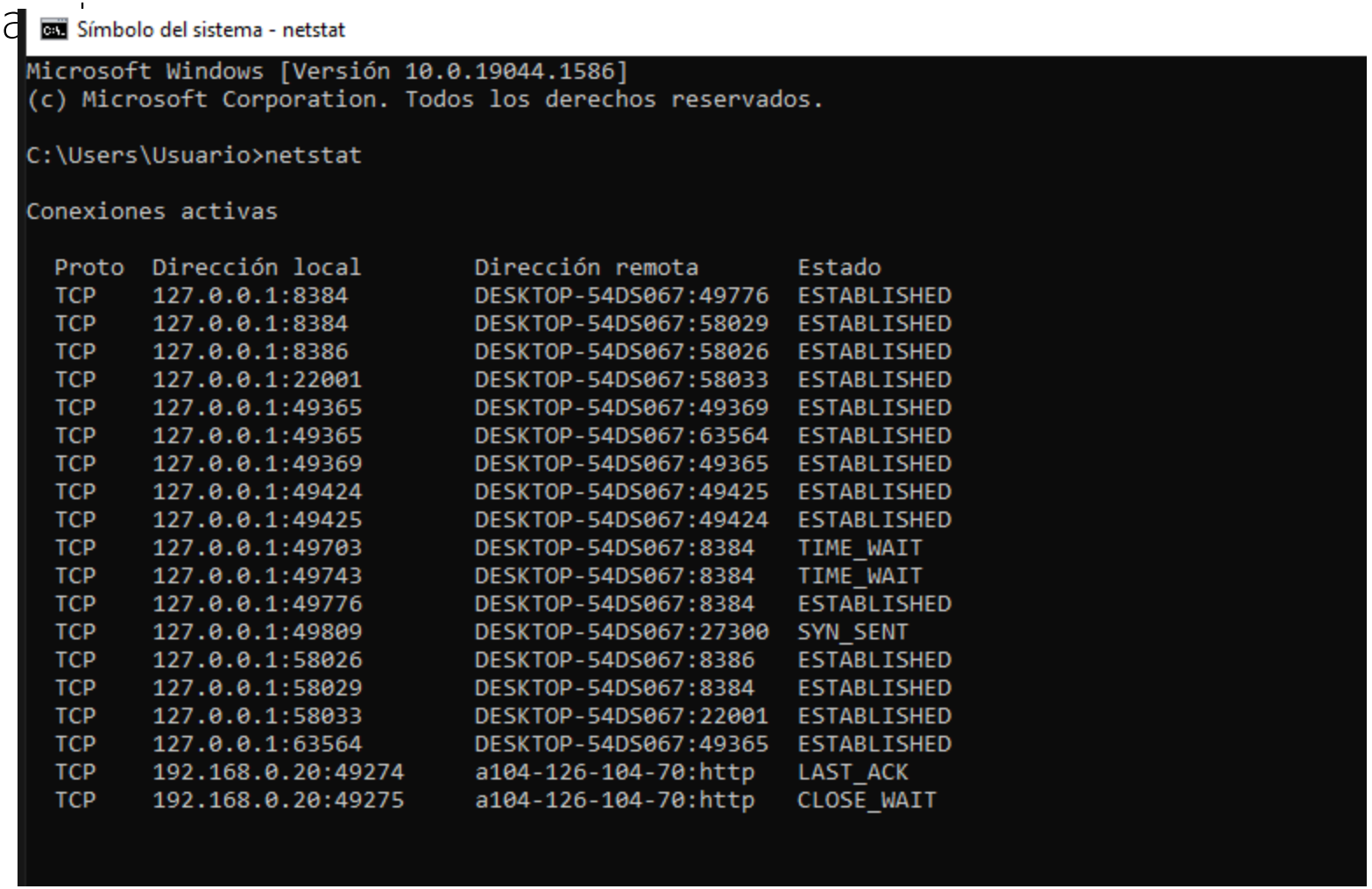
- El socket se utiliza para identificar el servidor y el servicio que solicita el cliente.
- Un socket de cliente (número de puerto de origen): 192.168.1.5:1099
- El s… estino):
  192…



Los sockets permiten que los diversos procesos que se ejecutan en un cliente se distingan entre sí. También permiten la diferenciación de diferentes conexiones a un proceso de servidor.

COMANDO NETSTAT

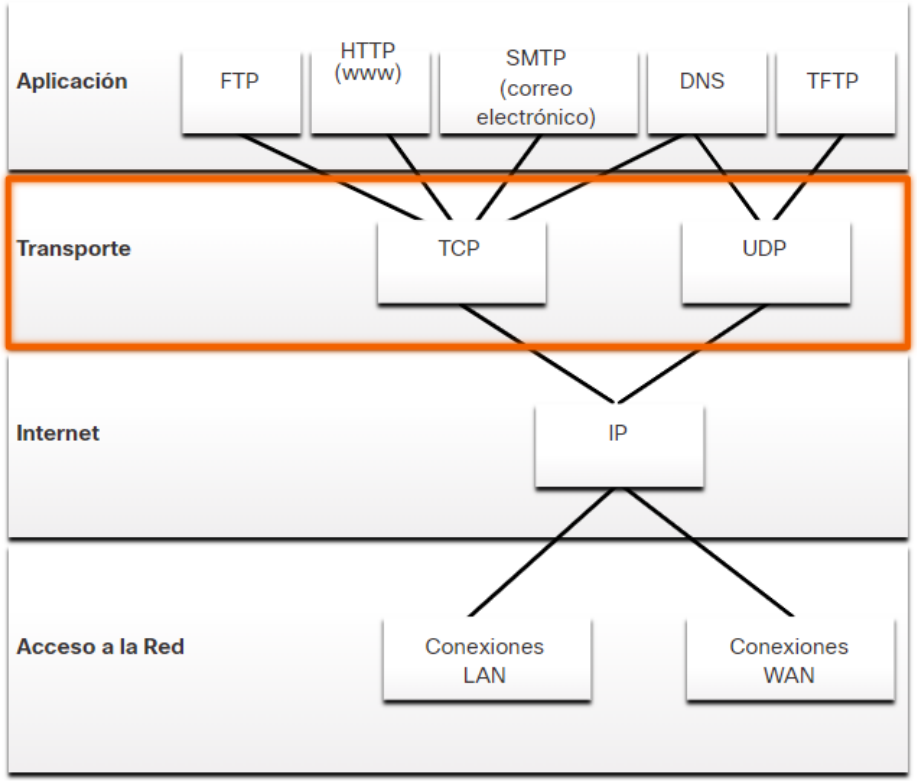
- Para ver las comunicaciones activas en nuestro equipo usamos el coma



PROTOCOLOS DE LA CAPA DE TRANSPORTE

- IP se ocupa solo de la estructura, el direccionamiento y el routing de paquetes. IP no especifica la manera en que se lleva a cabo la entrega o el transporte de los paquetes.

- Los protocolos de capa de transporte especifican cómo transferir mensajes entre hosts y son responsables de administrar los requisitos de fiabilidad de una conversación. La capa de transporte incluye los protocolos TCP y UDP.



PROTOCOLO TCP TRANSMISION CONTROL PROTOCOL

INTRODUCCIÓN

- TCP está definido en el estándar RFC 793
- TCP es **orientado a conexión y fiable** → complejo
- Cuando **TCP** recibe datos los fragmenta en trozos (llamados **segmentos**) de hasta 64Kb y los inserta en el área de datos de un datagrama IP
- Como no está garantizada la entrega de esos paquetes, ni el orden, TCP debe encargarse del control y la ordenación, utilizando contadores y números de secuencia

FUNCIONES

- La función del protocolo de transporte TCP es similar al envío de paquetes de los que se hace un rastreo de origen a destino. Si se divide un pedido de envío en varios paquetes, un cliente puede verificar en línea para ver el orden de la entrega.

- TCP proporciona confiabilidad y control de flujo mediante estas operaciones básicas:

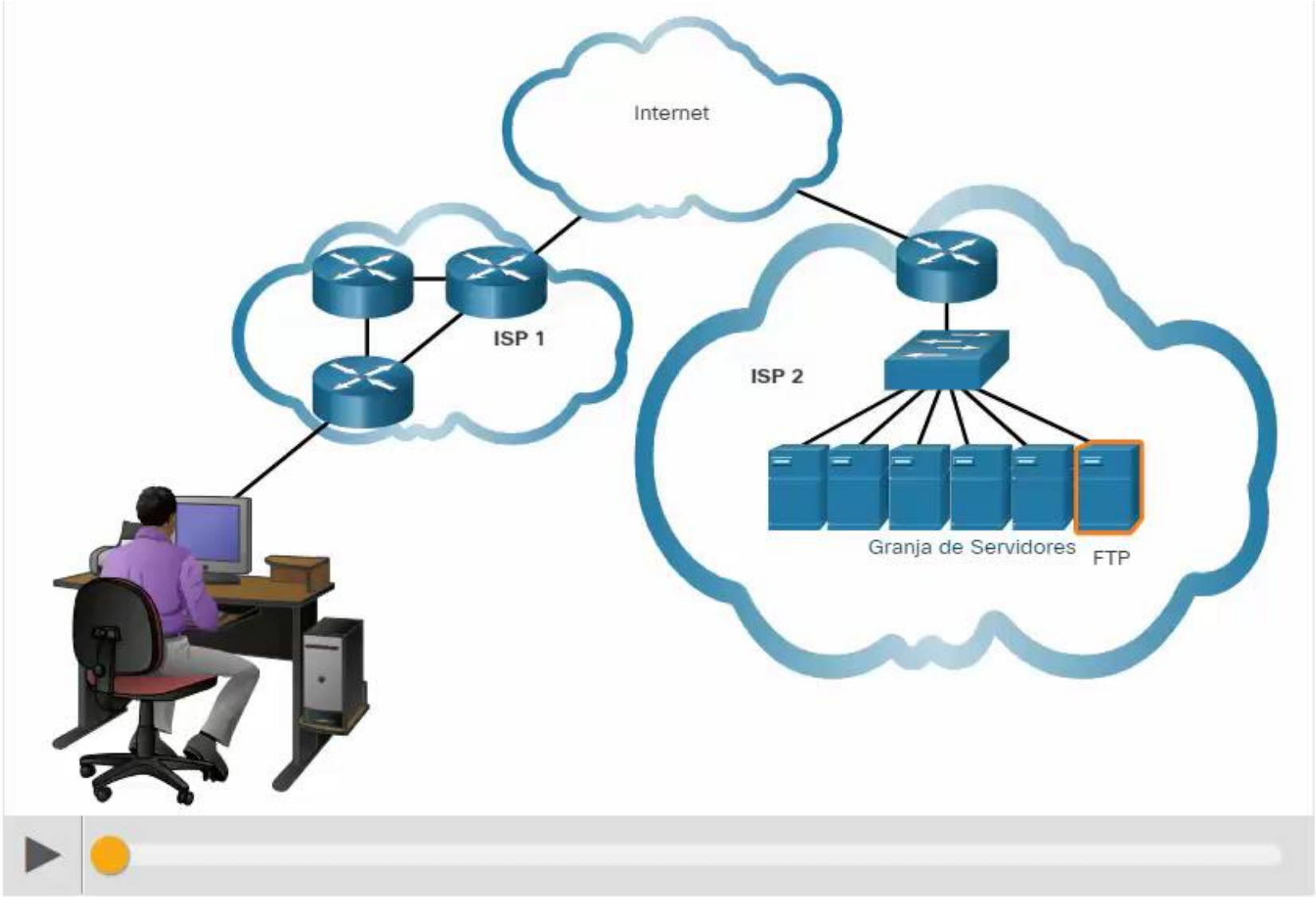
  - Enumerar y rastrear segmentos de datos transmitidos a un host específico desde una aplicación específica
  - Confirmar datos recibidos
  - Retransmitir cualquier información no reconocida después de un cierto período de tiempo
  - Secuenciar datos que pueden llegar en un orden incorrecto
  - Enviar datos a una velocidad eficiente que sea aceptable por el receptor

FUNCIONES

Además de admitir las funciones básicas de segmentación y reensamblado de datos, TCP también proporciona los siguientes servicios:

- Establece una sesión - TCP es un protocolo orientado a la conexión que negocia y establece una conexión permanente (o sesión) entre los dispositivos de origen y destino antes de reenviar cualquier tráfico. Mediante el establecimiento de sesión, los dispositivos negocian la cantidad de tráfico que se puede reenviar en un momento determinado, y los datos que se comunican entre ambos se pueden administrar detenidamente.
- Garantiza una entrega confiable - por muchas razones, es posible que un segmento se corrompa o se pierda por completo, ya que se transmite a través de la red. TCP asegura que cada segmento que envía la fuente llega al destino.
- Proporciona entrega en el mismo pedido - Debido a que las redes pueden proporcionar múltiples rutas que pueden tener diferentes velocidades de transmisión, los datos pueden llegar en el orden incorrecto. Al numerar y secuenciar los segmentos, TCP garantiza que los segmentos se vuelvan a ensamblar en el orden correcto.
- Admite control de flujo - los hosts de red tienen recursos limitados (es decir, memoria y potencia de procesamiento). Cuando TCP advierte que estos recursos están sobrecargados, puede solicitar que la aplicación emisora reduzca la velocidad del flujo de datos. Esto lo lleva a un cabo TCP, que regula la cantidad de datos que transmite el origen. El control de flujo puede evitar la necesidad de retransmitir los

EJEMPLO



FORMATO SEGMENTO TCP

Offsets Octeto		0								1								2								3							
Octeto	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Puerto de origen																Puerto de destino															
4	32	Número de secuencia																															
8	64	Número de acuse de recibo (si ACK es establecido)																															
12	96	Longitud de Cabecera				Reservado				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Tamaño de Ventana														
16	128	Suma de verificación																Puntero urgente (si URG es establecido)															
20 ...	160 ...	Opciones (Si la Longitud de Cabecera > 5, relleno al final con "0" bytes si es necesario) ...																															

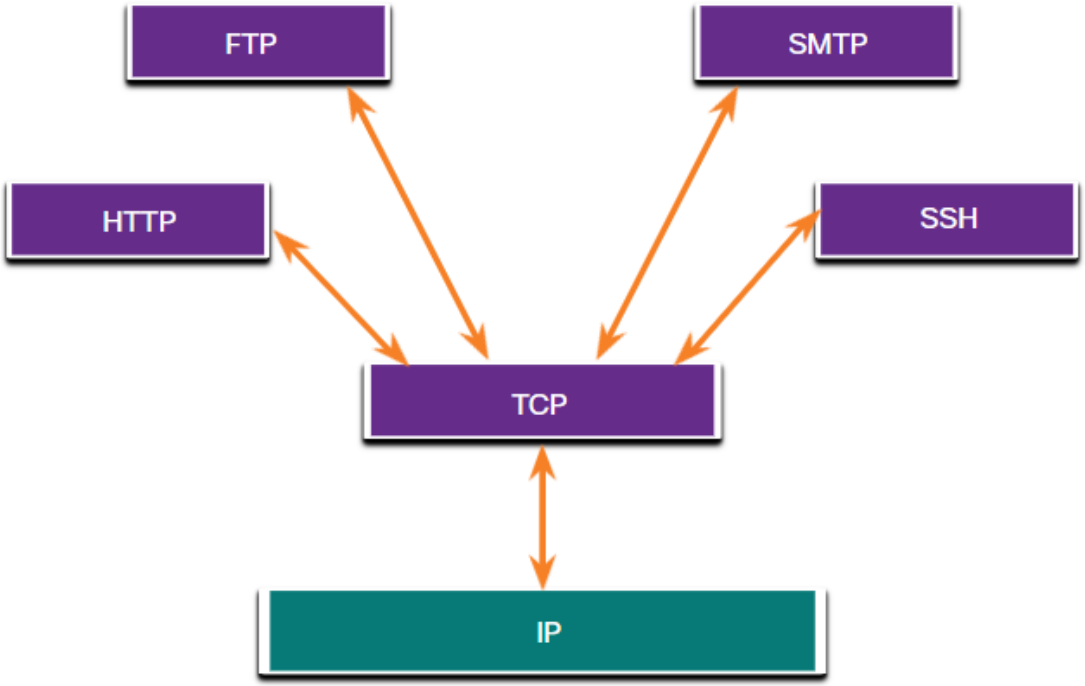
FORMATO SEGMENTO TCP

- La tabla identifica y describe los diez campos de un encabezado TCP

Campo de Encabezado TCP	Descripción
Puerto de Origen	Campo de 16 bits utilizado para identificar la aplicación de origen por número de puerto.
Puerto de Destino	Un campo de 16 bits utilizado para identificar la aplicación de destino por puerto número.
Secuencia de Números	Campo de 32 bits utilizado para reensamblar datos.
Número de Acuse de Recibo	Un campo de 32 bits utilizado para indicar que se han recibido datos y el siguiente byte esperado de la fuente.
Longitud del Encabezado	Un campo de 4 bits conocido como «desplazamiento de datos» que indica la propiedad longitud del encabezado del segmento TCP.
Reservado	Un campo de 6 bits que está reservado para uso futuro.
Bits de Control	Un campo de 6 bits utilizado que incluye códigos de bits, o indicadores, que indican el propósito y función del segmento TCP.
Tamaño de la ventana	Un campo de 16 bits utilizado para indicar el número de bytes que se pueden aceptar a la vez.
Suma de Comprobación	A 16-bit field used for error checking of the segment header and data.
Urgente	Campo de 16 bits utilizado para indicar si los datos contenidos son urgentes.

APLICACIONES QUE UTILIZAN TCP

- TCP maneja todas las tareas asociadas con la división del flujo de datos en segmentos, proporcionando confiabilidad, controlando el flujo de datos y reordenando segmentos.
- TCP libera la aplicación de tener que administrar estas tareas. Las aplicaciones, como las que se muestran en la figura, simplemente puede enviar el flujo de datos a la capa de transporte y utilizar los servicios d



ESTABLECIMIENTO DE CONEXIONES TCP

- En algunas culturas, cuando dos personas se conocen, generalmente se saludan dándose la mano. Ambas partes entienden el acto de estrechar la mano como una señal de saludo amistoso.
- Las conexiones en la red son similares.

FUNCIONAMIENTO

- Las conexiones TCP se componen de tres fases:

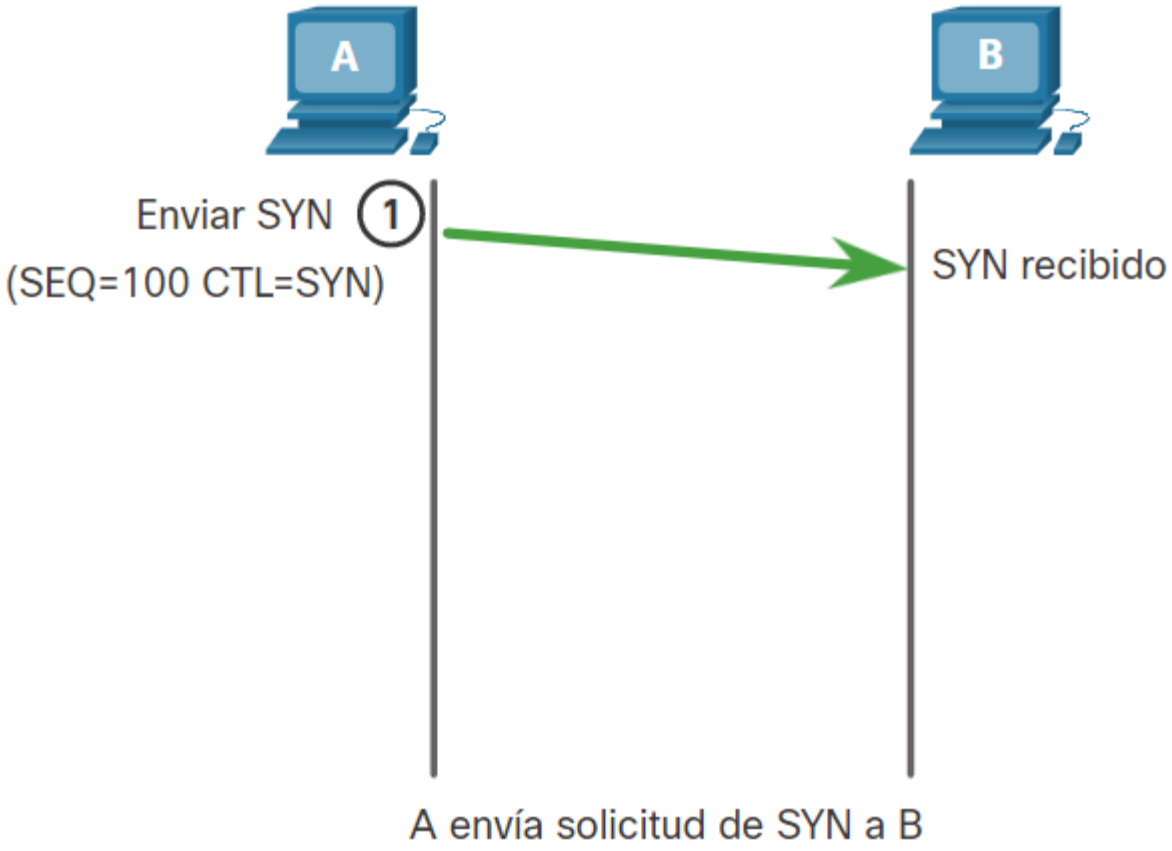
- En las conexiones TCP, el cliente host establece la conexión con el servidor mediante el proceso de "negociación en tres pasos".
- Para liberar la conexión se usa una "negociación en cuatro pasos"
- A continuación, se explicará cada una de las fases

ESTABLECIMIENTO DE LA CONEXIÓN

Establecimiento de la conexión → Transferencia de datos → Fin de la conexión

PASO 1. SYN

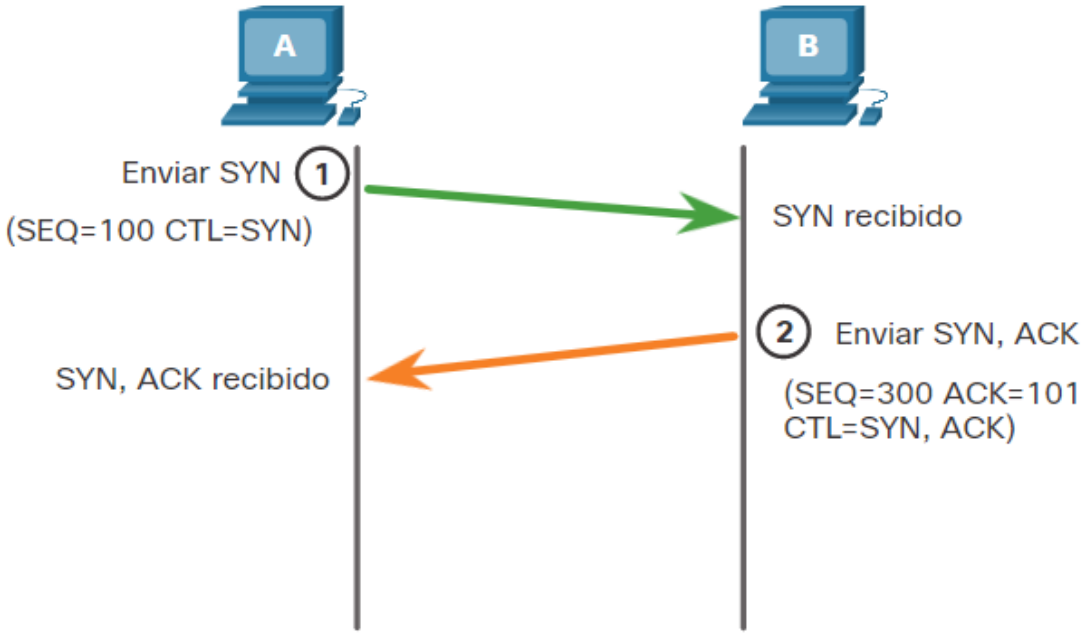
- El cliente solicita una sesión de comunicación con el servidor y realiza e SYN al servic



ESTABLECIMIENTO DE LA CONEXIÓN

Paso 2. ACK y SYN

- El servidor comprueba si el puerto está abierto (es decir, si existe un proceso escuchando en dicho puerto)
  - Si no lo está se envía al cliente un paquete de respuesta con el bit RST activado. Rechazo de conexión
  - Si está abierto, se envía un paquete SYN-ACK



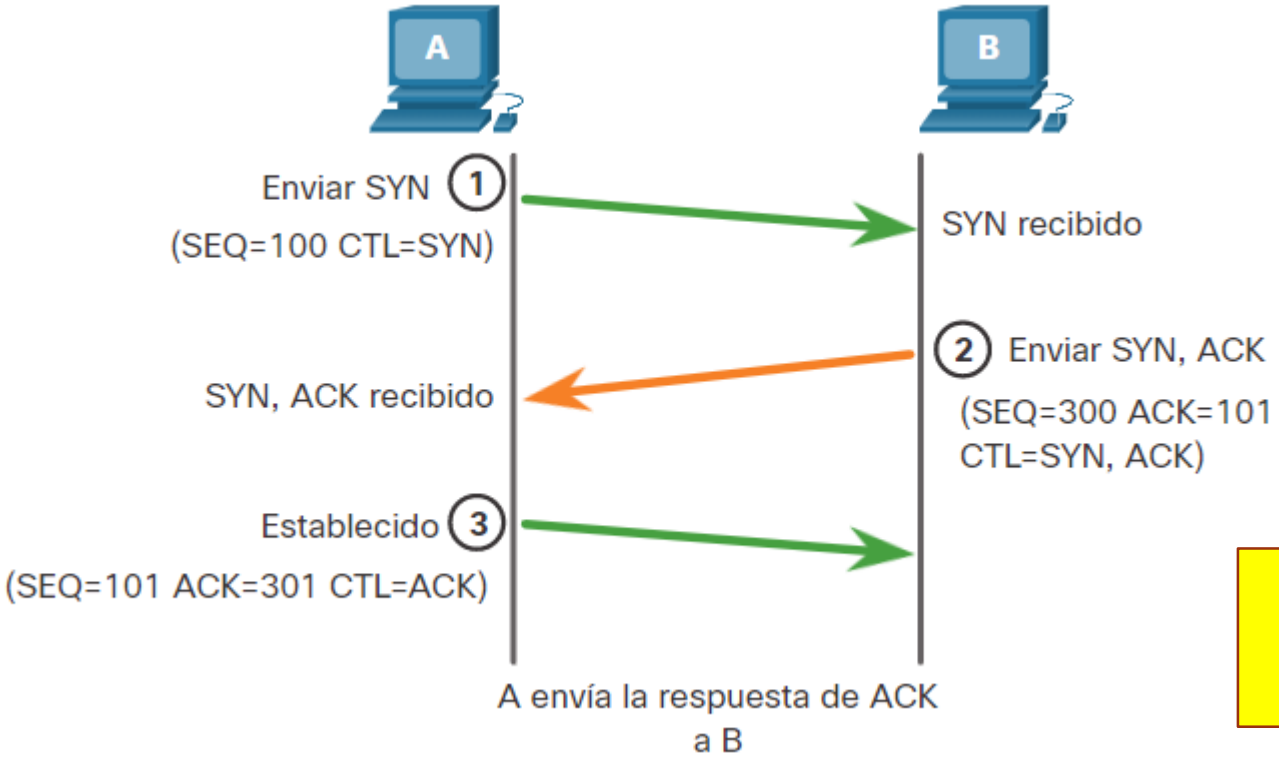
B envía la respuesta de ACK y la solicitud de SYN a A

ESTABLECIMIENTO DE LA CONEXIÓN

Establecimiento de la conexión → Transferencia de datos → Fin de la conexión

PASO 3. ACK

- El cliente debe responder con un ACK completando la negociación en tres pasos



Establecimiento conexión
Negociación en tres pasos

TRANSFERENCIA DE DATOS

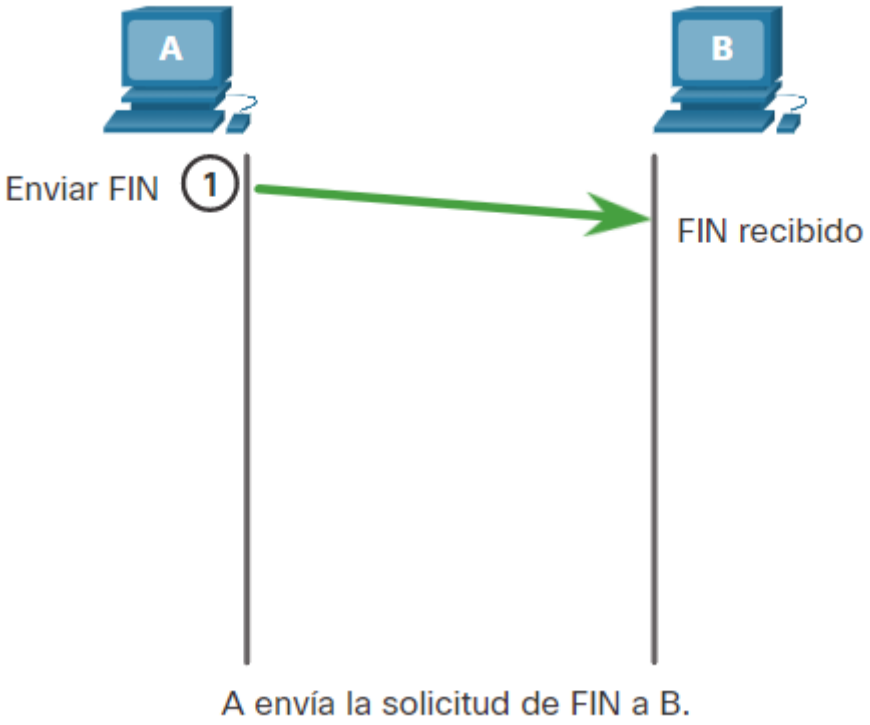
- En esta etapa se utilizan una serie de **mecanismos para asegurar la fiabilidad y robustez del protocolo**.
- Entre ellos están:
  - el uso del nº de secuencia para ordenar los segmentos TCP recibidos y detectar paquetes duplicados,
  - *checksum* para detectar errores
  - temporizadores para detectar perdidas y retrasos.

FIN DE LA CONEXIÓN

Establecimiento de la conexión → Transferencia de datos → Fin de la conexión

Paso 1. FIN

- Cuando el cliente no tiene más datos para enviar en la transmisión, envía un segmento con el indicador FIN establecido.

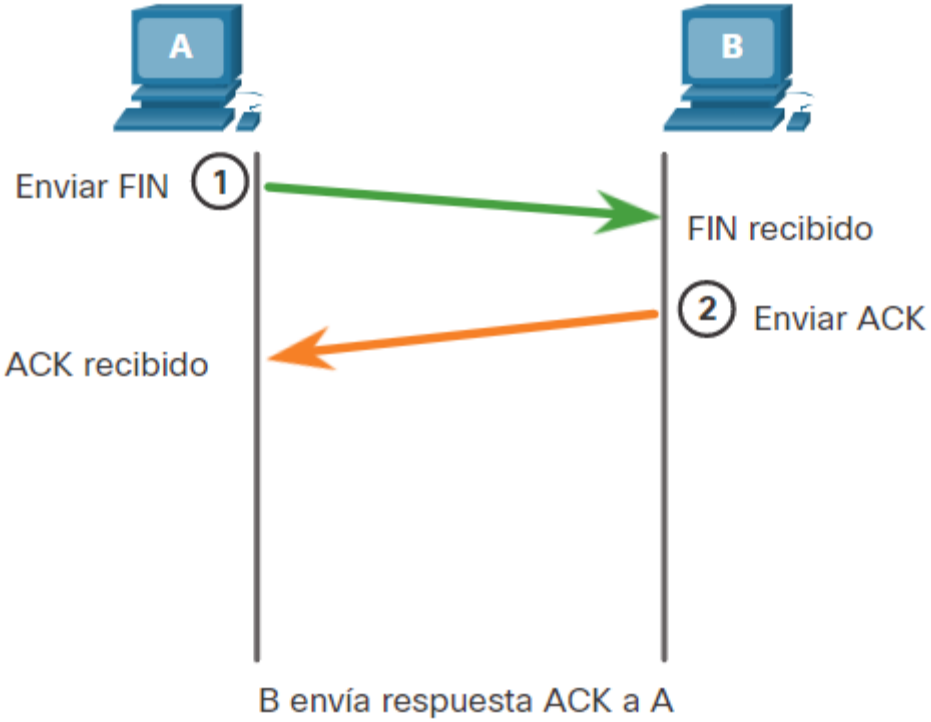


FIN DE LA CONEXIÓN

Establecimiento de la conexión → Transferencia de datos → Fin de la conexión

Paso 2. ACK

El servidor envía un ACK para acusar recibo del FIN para terminar la sesión de cliente a servidor.

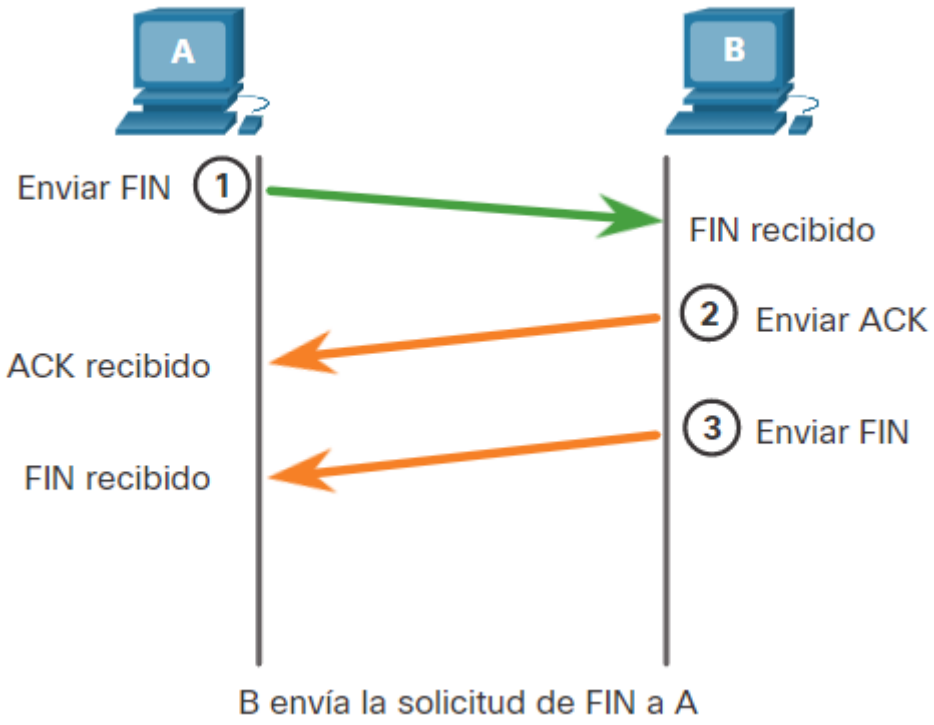


FIN DE LA CONEXIÓN

Establecimiento de la conexión → Transferencia de datos → Fin de la conexión

Paso 3. FIN

El servidor envía un FIN al cliente para terminar la sesión de servidor a cliente.

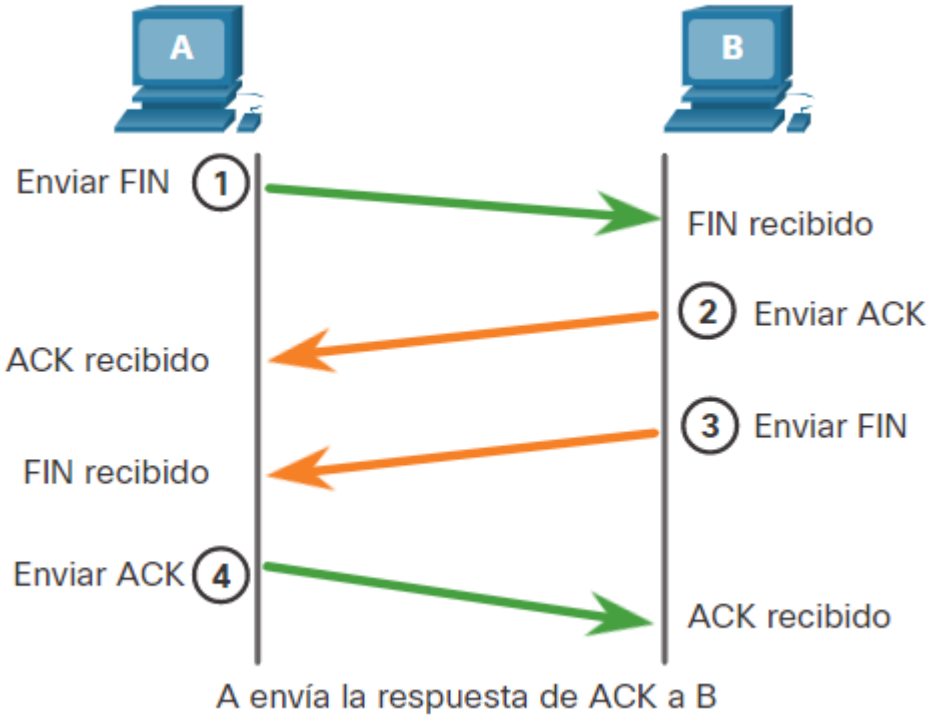


FIN DE LA CONEXIÓN

Establecimiento de la conexión → Transferencia de datos → Fin de la conexión

Paso 4. ACK

El cliente responde con un ACK para dar acuse de recibo del FIN desde el servidor.



Fin conexión
Negociación en cuatro pasos

ESTADOS DE CONEXIÓN TCP

Estado	Lado	Significado
LISTENING	Servidor	El servidor está a la espera de recibir peticiones SYN
SYN_SENT	Cliente	El cliente ha enviado una petición SYN al servidor
SYN_RCVD	Servidor	El servidor ha recibido una petición SYN desde un cliente
ESTABLISHED	Ambos	Conexión establecida El cliente ha recibido un reconocimiento SYN-ACK El servidor ha recibido un reconocimiento ACK
Cierre pasivo		
CLOSE_WAIT	Ambos	FIN recibido y ACK enviado desde ESTABLISHED
LAST_ACK	Ambos	FIN enviado y esperando ACK desde CLOSE_WAIT
CLOSED	Ambos	Conexión terminada. ACK recibido desde LAST_ACK
Cierre activo o local		
FIN_WAIT_1	Ambos	FIN enviado y esperando ACK desde ESTABLISHED
FIN_WAIT_2	Ambos	ACK recibido desde FIN_WAIT_1
TIME_WAIT	Solo uno	FIN recibido y ACK enviado desde FIN_WAIT_2
CLOSED	Ambos	Conexión terminada.

ESTADOS DE UNA CONEXIÓN TCP

- Los primeros estados (*listening, syn_sent, syn_rcvd y established*) están relacionados con el establecimiento de una conexión mediante negociación en tres pasos.
- Una vez establecida la conexión se intercambian los datos y ambos sockets están en el estado *established*
- Por último, se cierra la conexión donde hay dos casos
  - Cierre pasivo
    - El socket remoto cierra su mitad enviando un FIN, de modo que el socket local pasa al estado *close_wait*, es decir, a la espera de que el programa local que inició la comunicación ejecute un close() para cerrar el socket local
  - Cierre activo
    - El socket local cierra su mitad enviando un FIN y pasa al estado *fin_wait_1*, si el socket remoto confirma entonces pasa al estado *fin_wait_2*, y cuando el socket remoto cierra su parte, el socket local pasa al estado *time_wait*, es decir, a la espera de un tiempo predeterminado para cerrarse automáticamente

CAMPO DE BITS DE CONTROL

- Los seis indicadores de bits de control son los siguientes:
  - URG - campo de puntero urgente significativo
  - ACK - Indicador de acuse de recibo utilizado en el establecimiento de la conexión y la terminación de la sesión
  - PSH - Función de empuje
  - RST - Restablece la conexión cuando se produce un error o un tiempo de espera
  - SYN - Sincroniza números de secuencia utilizados en el establecimiento de conexión
  - FIN - No más datos del remitente y se utilizan en la terminación de la sesión

PROTOCOLO UDP
USER DATAGRAM PROTOCOL

INTRODUCCIÓN

- UDP está definido en el estándar RFC 768
- UDP es un protocolo de transporte liviano que ofrece la misma segmentación y rearmado de datos que TCP, pero sin la confiabilidad y el control del flujo de TCP.

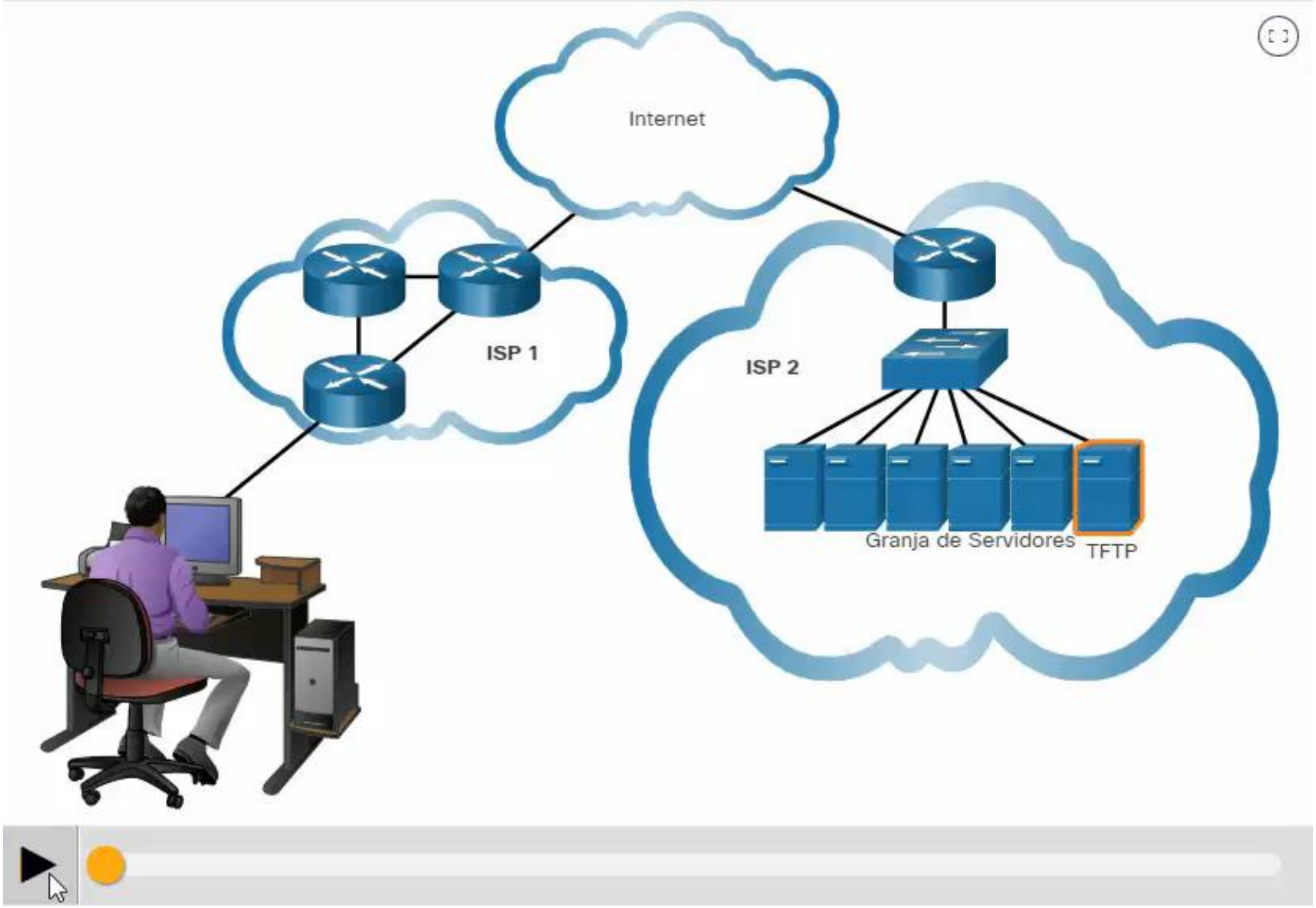
- Las características UDP incluyen lo siguiente:

  - Los datos se reconstruyen en el orden en que se recibieron.
  - Los segmentos perdidos no se vuelven a enviar.
  - No hay establecimiento de sesión.
  - El envío no está informado sobre la disponibilidad de recursos.

CARACTERÍSTICAS

- UDP es un protocolo del nivel de transporte basado en el intercambio de **datagramas**
- Es:
  - No orientado a conexión
    - Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera
  - No fiable
    - No se sabe si los paquetes han llegado correctamente, ya que no hay confirmación de entrega o recepción. No tiene control de flujo, por lo que los paquetes pueden adelantarse unos a otros y llegar desordenados

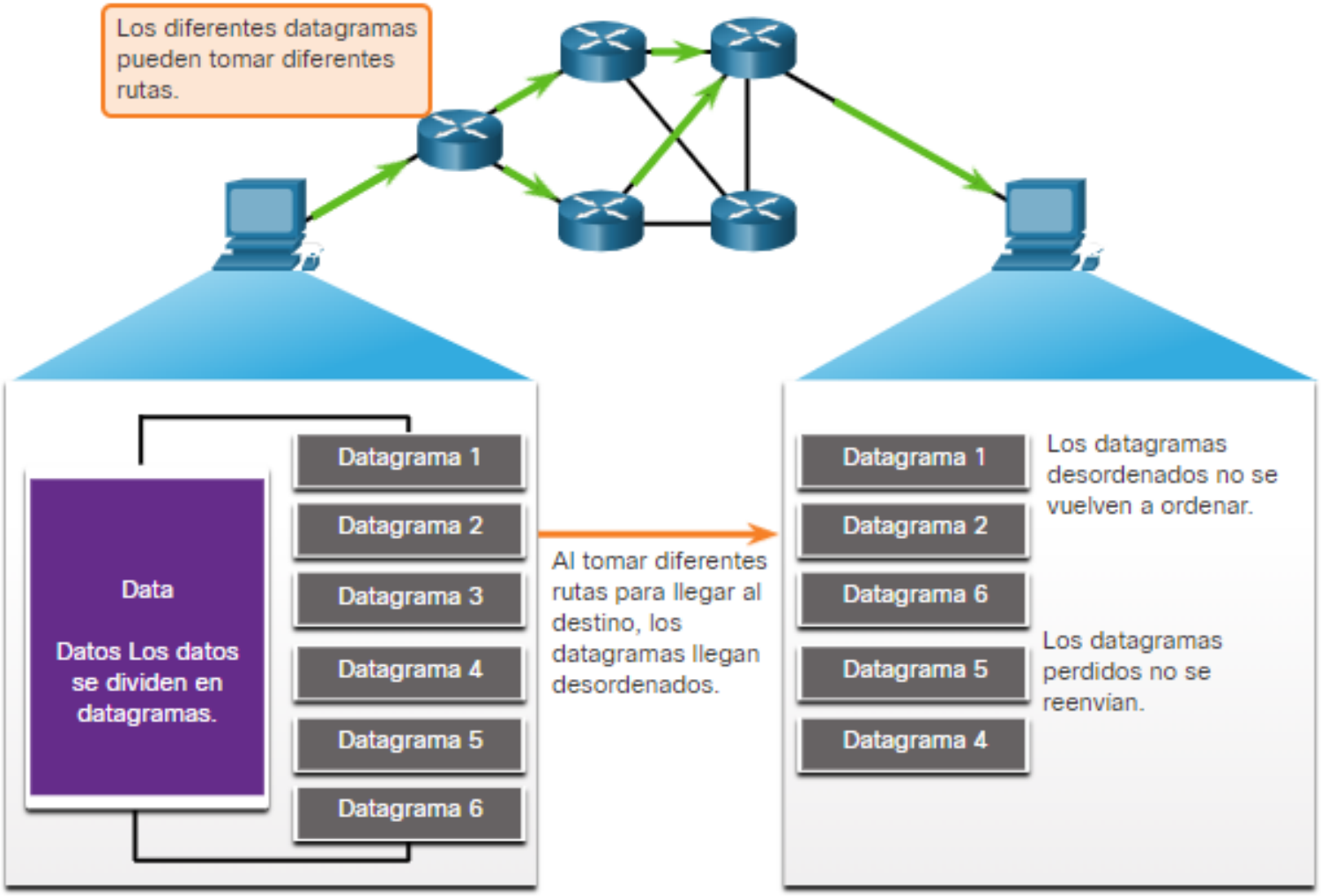
EJEMPLO



FUNCIONAMIENTO

- Cuando se envían datagramas UDP a un destino, a menudo toman diferentes rutas y llegan en el orden equivocado.
- UDP no realiza un seguimiento de los números de secuencia de la manera en que lo hace TCP. UDP no tiene forma de reordenar datagramas en el orden en que se transmiten
- Por lo tanto, UDP simplemente reensambla los datos en el orden en que se recibieron y los envía a la aplicación. Si la secuencia de datos es importante para la aplicación, esta debe identificar la secuencia adecuada y determinar cómo se deben procesar los datos.

FUNCIONAMIENTO



FORMATO SEGMENTO UDP

- Los bloques de comunicación en UDP se denominan datagramas.
- El encabezado UDP es mucho más simple que el encabezado TCP porque solo tiene cuatro campos y requiere 8 bytes (es decir, 64 bits).
- La cabecera UDP consta de 4 campos de los cuales 2 son opcionales (con fondo rojo en la tabla).

Bits	0 - 15	16 - 31
0	Puerto origen	Puerto destino
32	Longitud del Mensaje	Suma de verificación
64	Datos	

FORMATO SEGMENTO UDP

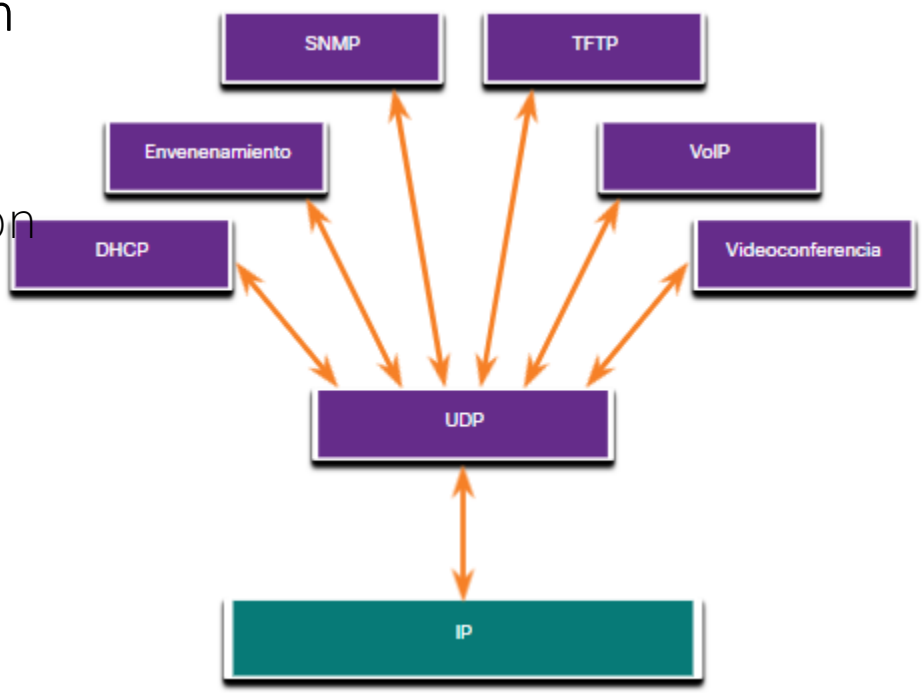
- La tabla identifica y describe los cuatro campos de un encabezado UDP.

Campo de Encabezado UDP	Descripción
Puerto de Origen	Campo de 16 bits utilizado para identificar la aplicación de origen por número de puerto.
Puerto de Destino	Un campo de 16 bits utilizado para identificar la aplicación de destino por puerto número.
Longitud	Campo de 16 bits que indica la longitud del encabezado del datagrama UDP.
Suma de comprobación	Campo de 16 bits utilizado para la comprobación de errores del encabezado y los datos del datagrama.

APLICACIONES USAN UDP

Existen tres tipos de aplicaciones que son las más adecuadas para UDP:

- Aplicaciones de video y multimedia en vivo: - estas aplicaciones pueden tolerar cierta pérdida de datos, pero requieren poco o ningún retraso. Los ejemplos incluyen VoIP y la transmisión de video en vivo.
- Solicitudes simples de solicitud y respuesta: - aplicaciones con transacciones simples en las que un host envía una solicitud y puede o no recibir una respuesta. Los ejemplos incluyen DNS y DHCP.
- Aplicaciones que manejan la confiabilidad por sí mismas: - comunicaciones unidireccionales donde el control de flujo, la detección de errores, los reconocimientos y la recuperación de errores no son



UDP VS TCP

PROTOCOLO DE LA CAPA DE TRANSPORTE CORRECTO PARA LA APLICACIÓN ADECUADA

- Algunas aplicaciones pueden tolerar cierta pérdida de datos durante la transmisión a través de la red, pero los retrasos en la transmisión son inaceptables. Para estas aplicaciones, UDP es la mejor opción porque requiere menos sobrecarga de red.
  - UDP es preferible para aplicaciones como Voz sobre IP (VoIP). Los reconocimientos y la retransmisión retrasarían la entrega y harían inaceptable la conversación de voz.
- Para otras aplicaciones es importante que todos los datos lleguen y que puedan ser procesados en su secuencia adecuada. Para estos tipos de aplicaciones, TCP se utiliza como protocolo de transporte.
  - Por ejemplo, las aplicaciones como las bases de datos, los navegadores web y los clientes de correo electrónico, requieren que todos los datos que se envían lleguen a destino en su formato original. Cualquier dato faltante podría corromper una comunicación, haciéndola incompleta o ilegible. Por ejemplo, cuando se accede a la información bancaria a través de la web, es importante asegurarse de que toda la información se envía y recibe correctamente.

UDP VS TCP

UDP

VoIP
(telefonía IP)

DNS
(Domain Nameresolución de nombre de dominio Resolution)

Propiedades de protocolo requeridas:

- Rápido
- Baja sobrecarga
- No requiere reconocimiento
- No reenvía los datos perdidos
- Entrega los datos a medida que llegan

TCP

SMTP/IMAP
(Correo electrónico)

HTTP/HTTPS
(World Wide Web)

Propiedades de protocolo requeridas:

- Confiable
- Reconoce los datos
- Reenvía los datos perdidos
- Entrega los datos en orden secuencial

BIBLIOGRAFÍA

- Capítulo 12. Nivel de transporte. Protocolos
  Planificación y Administración de Redes. Editorial Garceta 2ª Edición
- Módulo 14. Capa de transporte
  Curso CCNA 1 Introduction To Networks v7

NIVEL TRANSPORTE