TEMA 4:
Capa de Transporte

UNIVERSIDAD POLITÉCNICA DE VALENCIA
Escuela Politécnica Superior de Gandia

Tema 4: Capa de transporte

4.0 Introducción

4.1 Protocolos de la capa de transporte

- Describir el propósito de la capa de transporte en la administración del transporte de datos en la comunicación de extremo a extremo.
- Describir las características de los protocolos TCP y UDP, incluidos los números de puerto y sus usos.

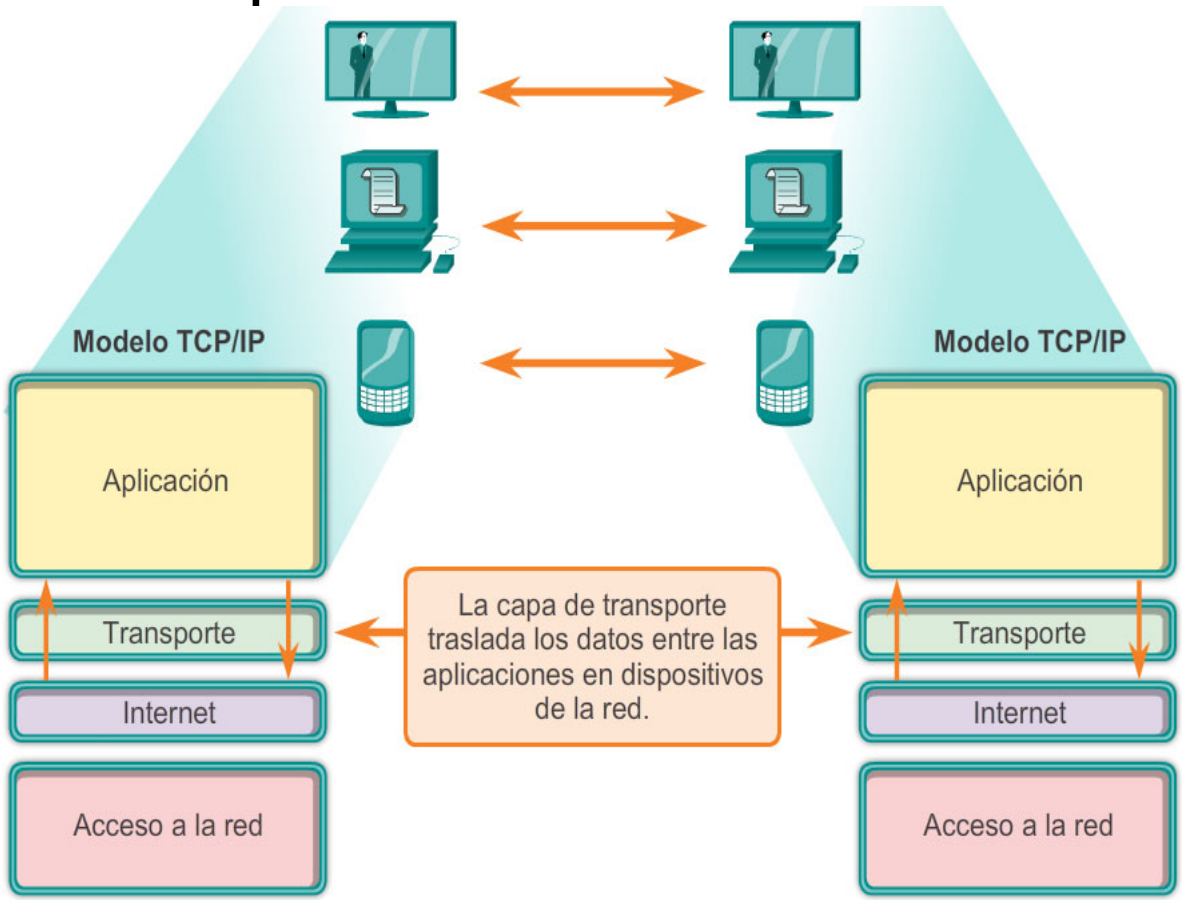
4.2 TCP y UDP

- Explicar la forma en que los procesos de establecimiento y finalización de sesión TCP promueven una comunicación confiable.
- Explicar la forma en que se transmiten y se reconocen las unidades de datos del protocolo TCP para garantizar la entrega.
- Describir los procesos de cliente UDP para establecer la comunicación con un servidor.
- Comparar UDP y TCP.

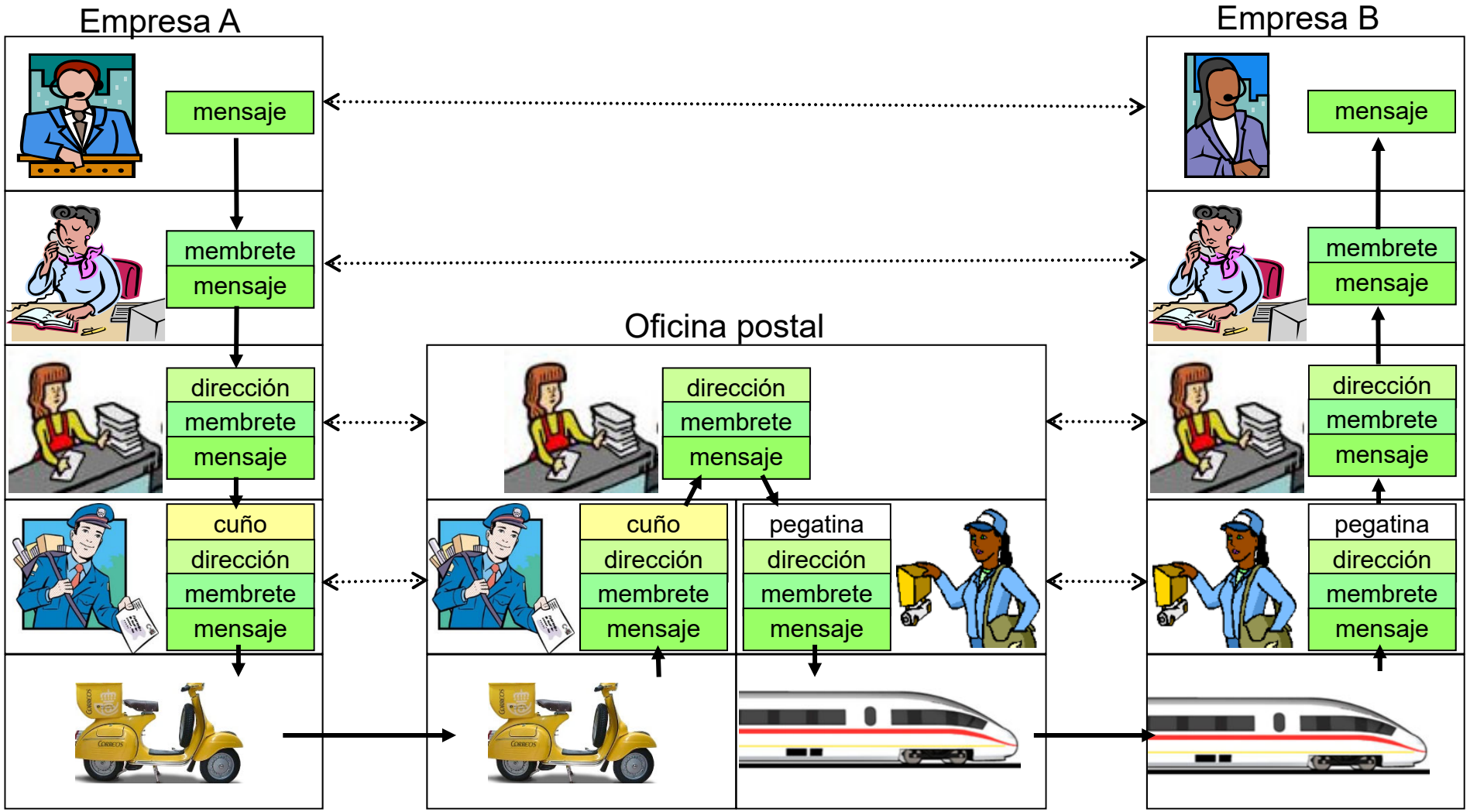
4.3 Resumen

Función de la capa de transporte

- Propósito: Conseguir una comunicación fiable entre dos aplicaciones



Símil de las empresas

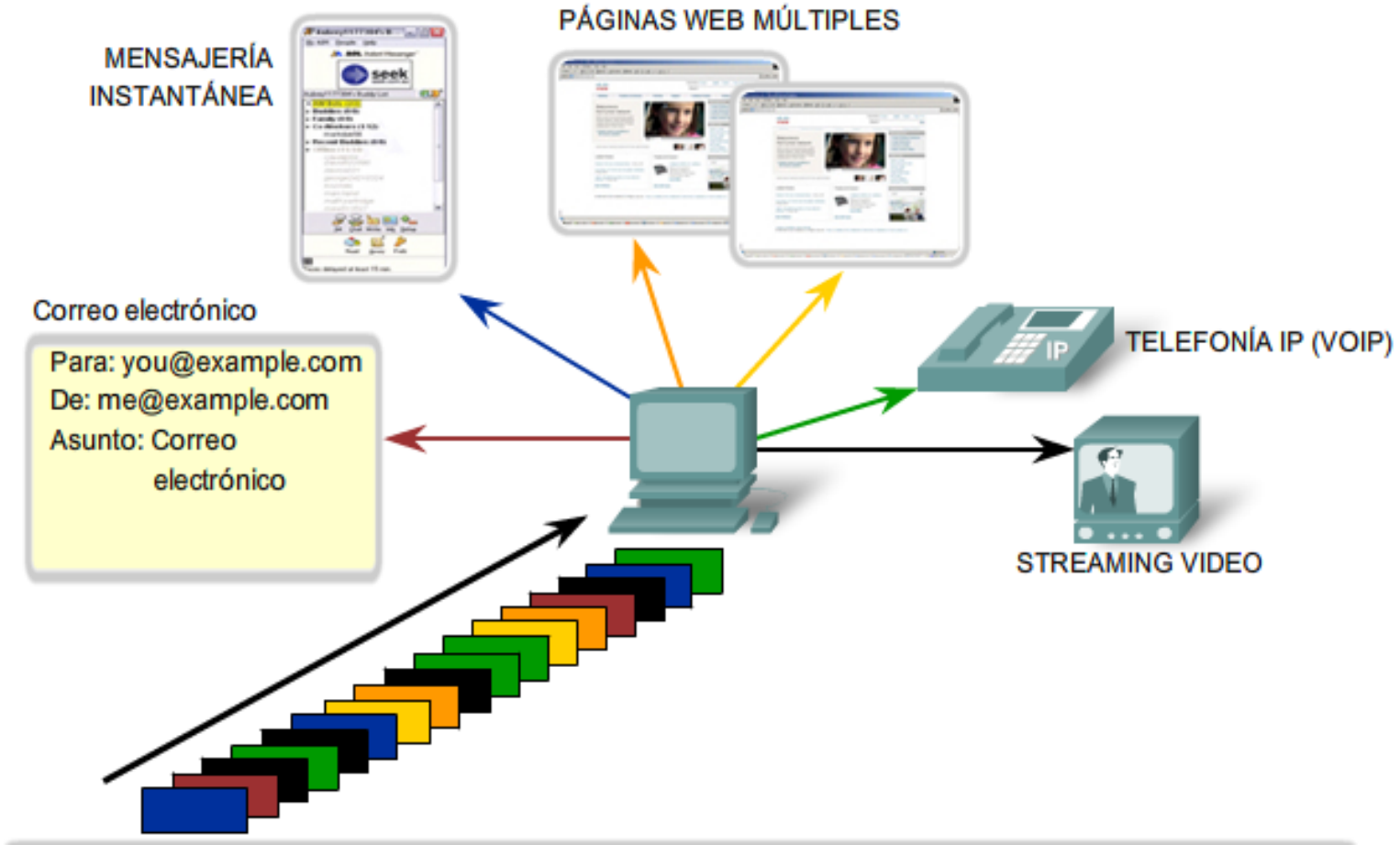


Función de la capa de transporte

- Funciones:
  - Separación de comunicaciones entre aplicaciones (puertos)
  - Segmentación de datos y reensamblado
  - Cubrir diferentes requisitos de aplicación (TCP/UDP)
  - Para comunicaciones orientadas a la conexión:
    - Entrega confiable
    - Control del flujo

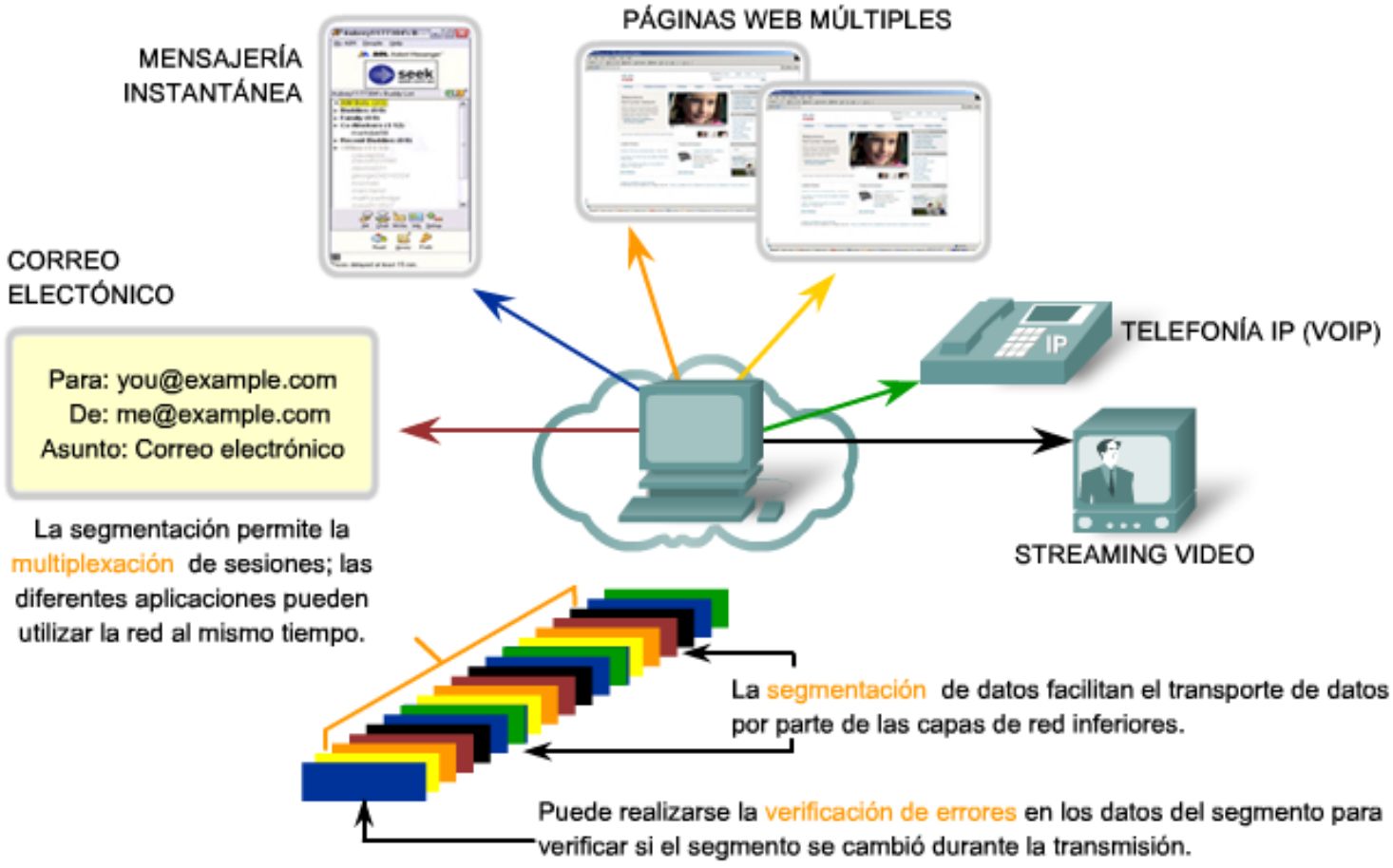
Función de la capa de transporte

- Separación de comunicaciones entre aplicaciones



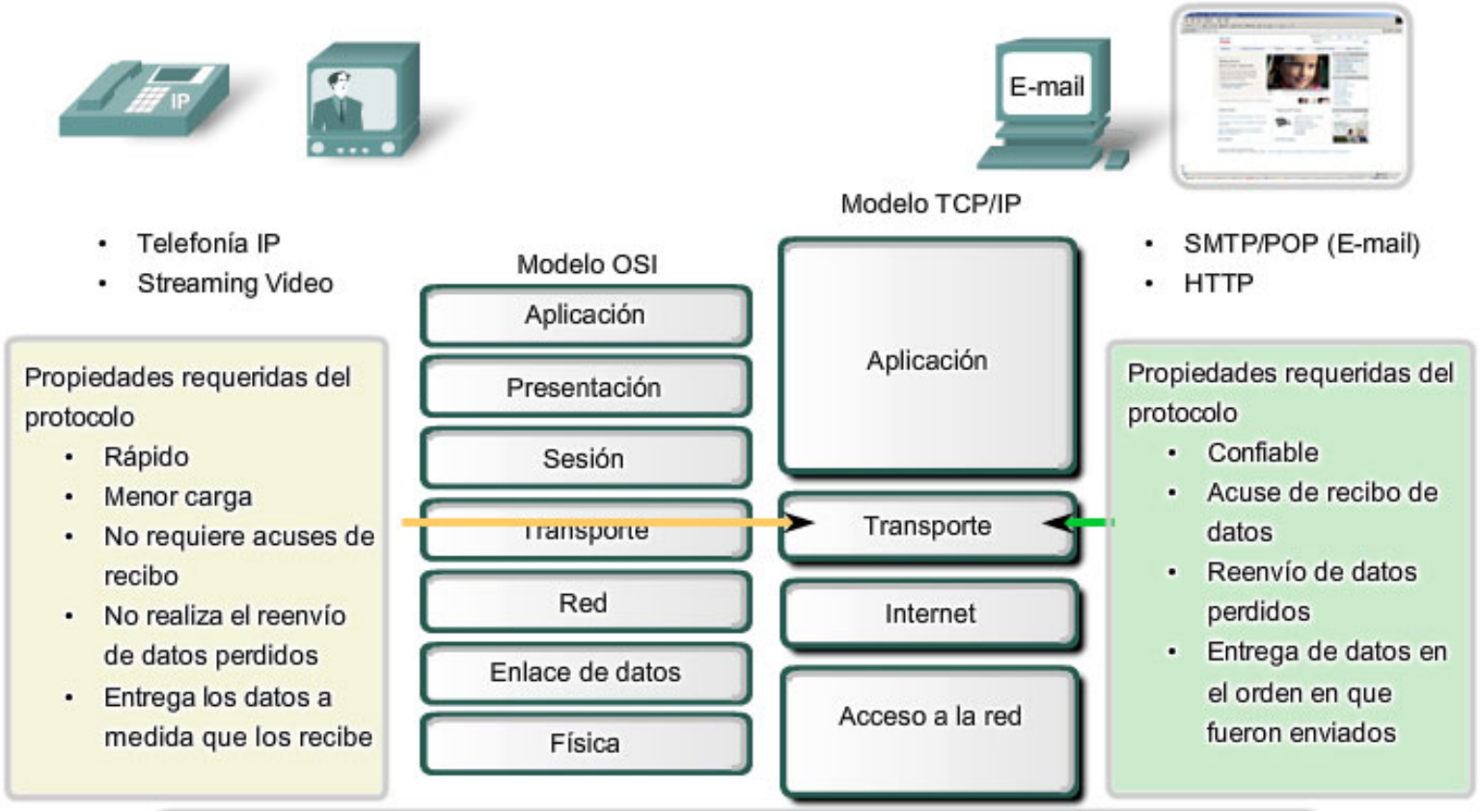
Función de la capa de transporte

- Segmentación de datos y reensamblado



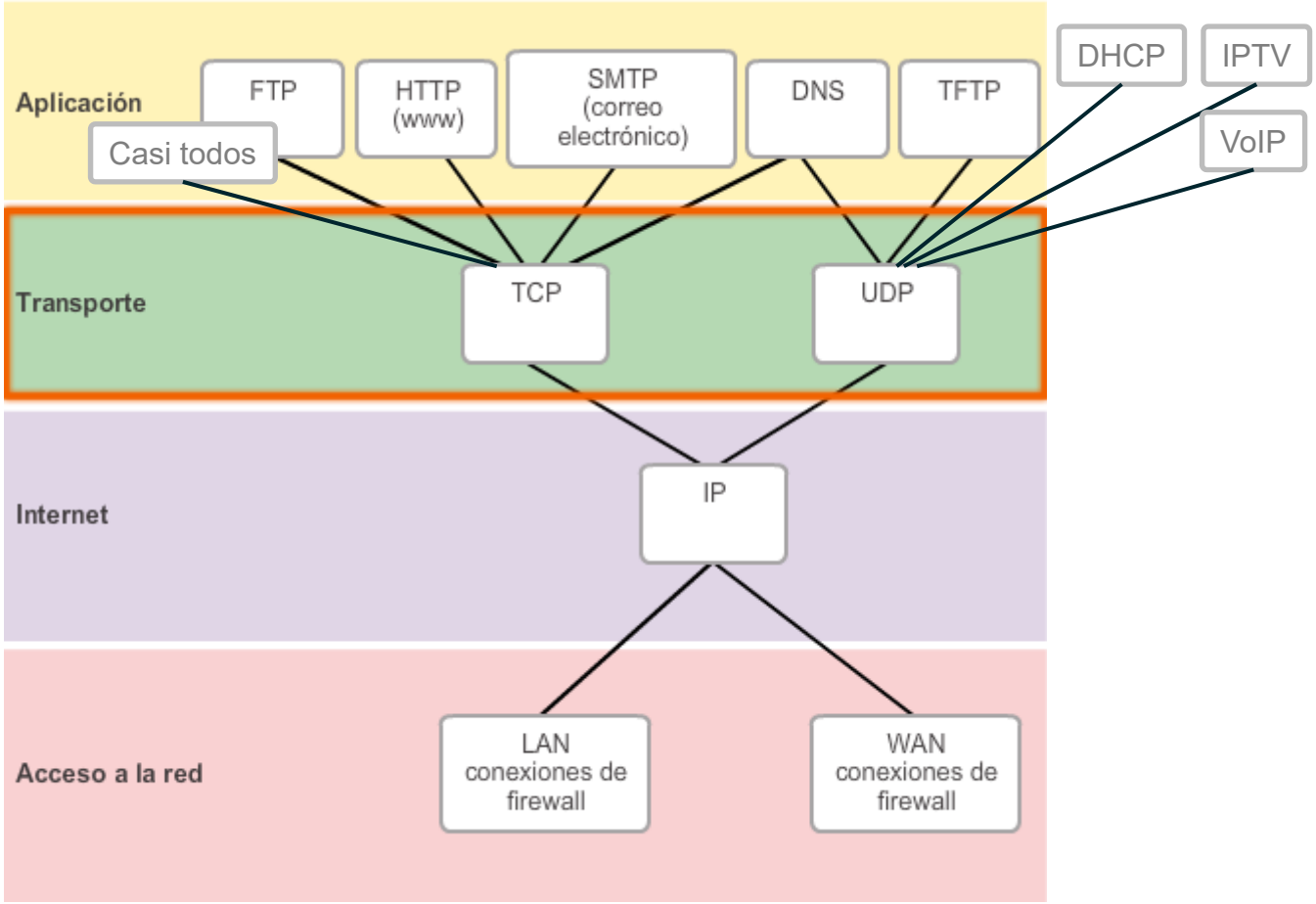
Función de la capa de transporte

- Cubrir diferentes requisitos de aplicación (TCP/UDP)



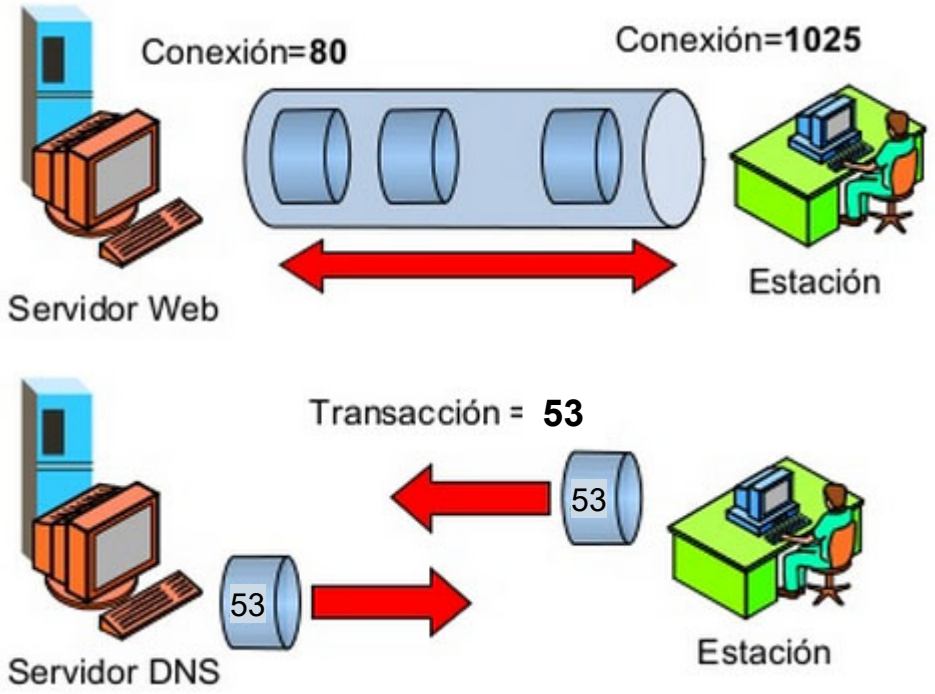
Función de la capa de transporte

- Cubrir diferentes requisitos de aplicación (TCP/UDP)



Función de la capa de transporte

- **Conversaciones orientadas a la conexión**
  - En una comunicación con conexión ambas partes se ponen de acuerdo antes de iniciar una conversación
  - En las redes de conmutación de paquetes, hay que “montar” esta conexión
- Las **conversaciones sin conexión** no necesitan estos requisitos



Función de la capa de transporte

- Funciones adicionales para conversaciones orientadas a la conexión:
  - **Entrega confiable**
    - Numeración de los segmentos transmitidos
    - Acuse de recibo de cada segmento
    - Retransmisión de cualquier segmento sin acuse de recibo
  - **Control del flujo**
    - Cuando el receptor está sobrecargado, el emisor ha de reducir la velocidad

Función de la capa de transporte

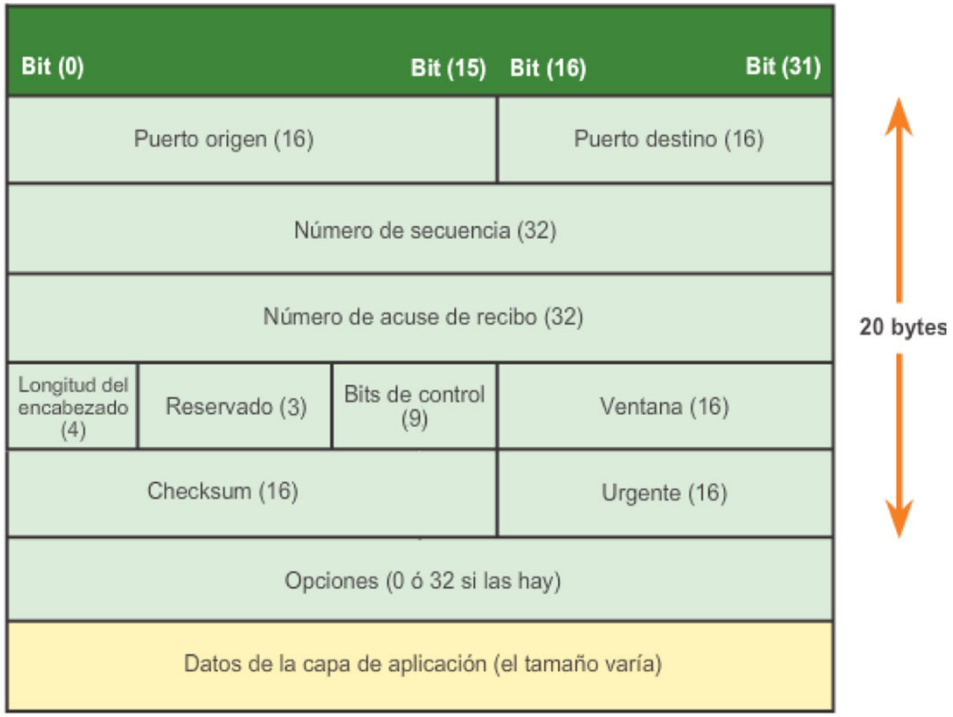
- Protocolo UDP
  - Muy simple y rápido (sólo separa tráfico entre aplicaciones)
  - Sin conexión
  - Aplicaciones: DNS, DHCP, TFTP, Streaming de vídeo (IPTV), Voz sobre IP (VoIP), juegos online,…
- Protocolo TCP
  - Más complejo, puede introducir retardos
  - Con conexión
  - Todas las funciones:
    - Segmentación y reensamblado
    - Entrega confiable
    - Control de flujo
  - Aplicaciones: la mayoría

Protocolos de la capa de transporte

Descripción general de TCP y UDP

- Encabezado TCP
  - TCP es un protocolo con información de estado.
  - TCP agrega 20 bytes de sobrecarga en el encabezado del segmento.

Segmento TCP

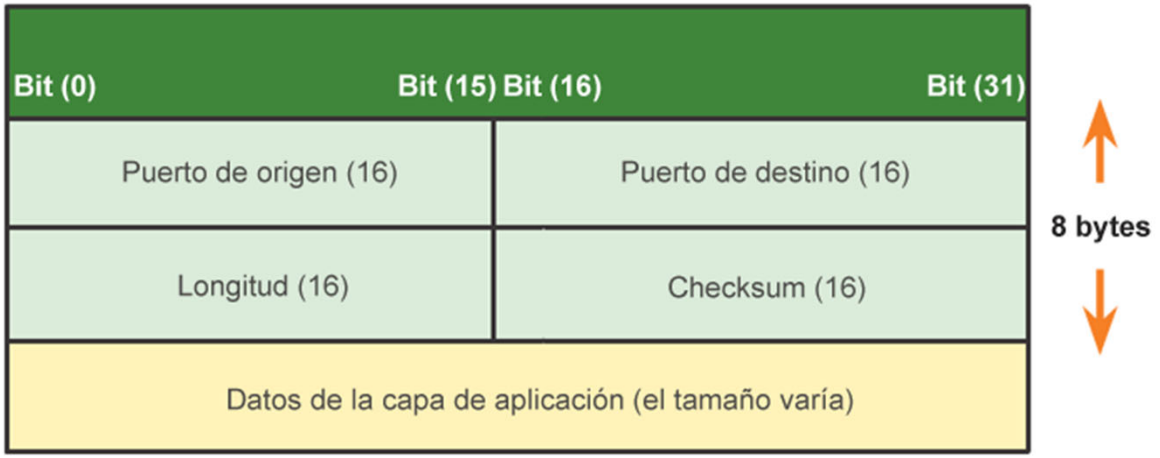


Protocolos de la capa de transporte

Descripción general de TCP y UDP

- Encabezado UDP
  - UDP es un protocolo sin información de estado.
  - Es la aplicación la que debe manejar la confiabilidad.
  - Las porciones de comunicación en UDP se denominan datagramas.
  - UDP agrega solo 8 bytes de sobrecarga.

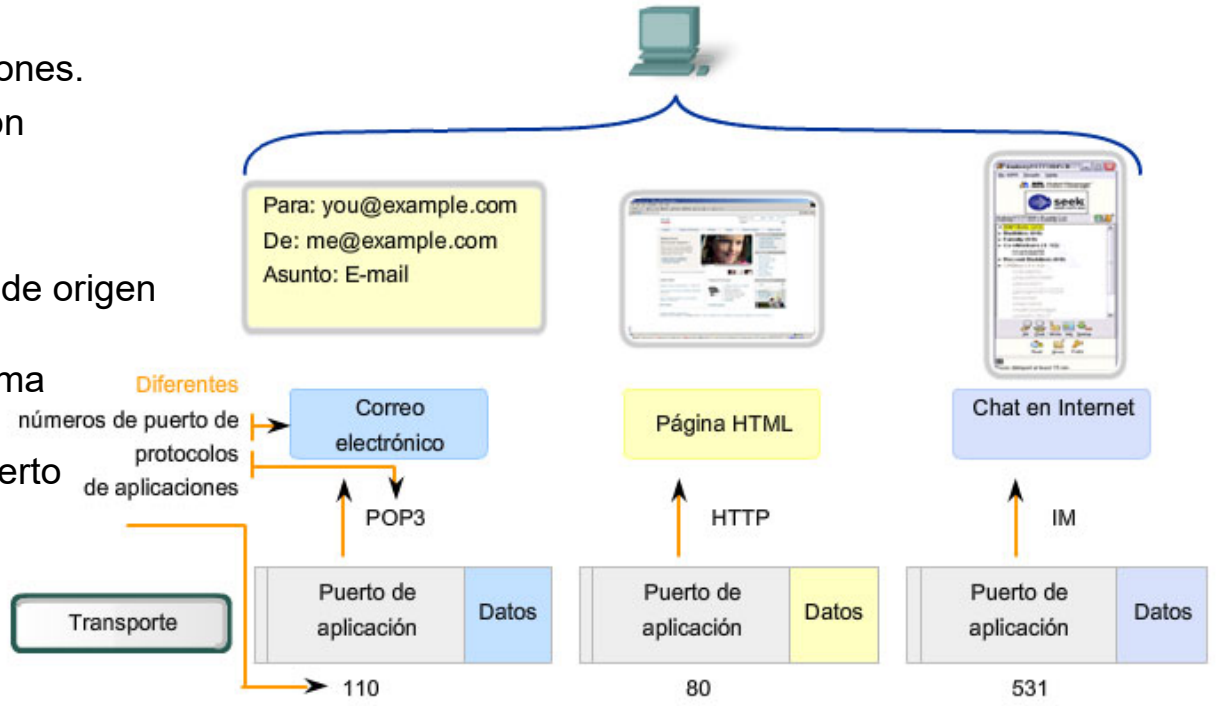
Datagrama UDP



Protocolos de la capa de transporte

Descripción general de TCP y UDP

- Varias conversaciones por separado
  - La capa de transporte separa y administra varias comunicaciones con diferentes requisitos de transporte.
  - Diferentes aplicaciones envían y reciben datos en la red de manera simultánea.
  - Los valores únicos de encabezado permiten que TCP y UDP administren estas distintas conversaciones simultáneas por medio de la identificación de estas aplicaciones.
  - Estos identificadores únicos son números de puertos.
- Números de puerto
  - Suelen verse en pares: puerto de origen y puerto de destino.
  - El cliente elige el puerto en forma dinámica a partir del 1024.
  - El puerto del servidor es un puerto conocido que se utiliza para identificar la aplicación.



Grupos de número de puerto (IANA)

- Puertos bien conocidos (0-1023):
  - Para servidores bien conocidos (80, 25, …)
- Puertos registrados (1024-49151)
  - Servidores menos conocidos (WhatsApp: 5223)
  - Servidores que el usuario decide instalar en otro puerto (8080)
  - Para clientes (cada vez menos), asignados de forma dinámica
- Puertos dinámicos o privados (49152-65535):
  - Para clientes, asignados de forma dinámica

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

Puertos TCP registrados:
1863 MSN Messenger
2000 Cisco SCCP (VoIP)
8008 HTTP alternativo
8080 HTTP alternativo

Puertos TCP bien conocidos:
21 FTP
23 Telnet
25 SMTP
80 HTTP
110 POP3
194 Internet Relay Chat (IRC)
443 HTTP seguro (HTTPS)

Grupos de número de puerto (IANA)

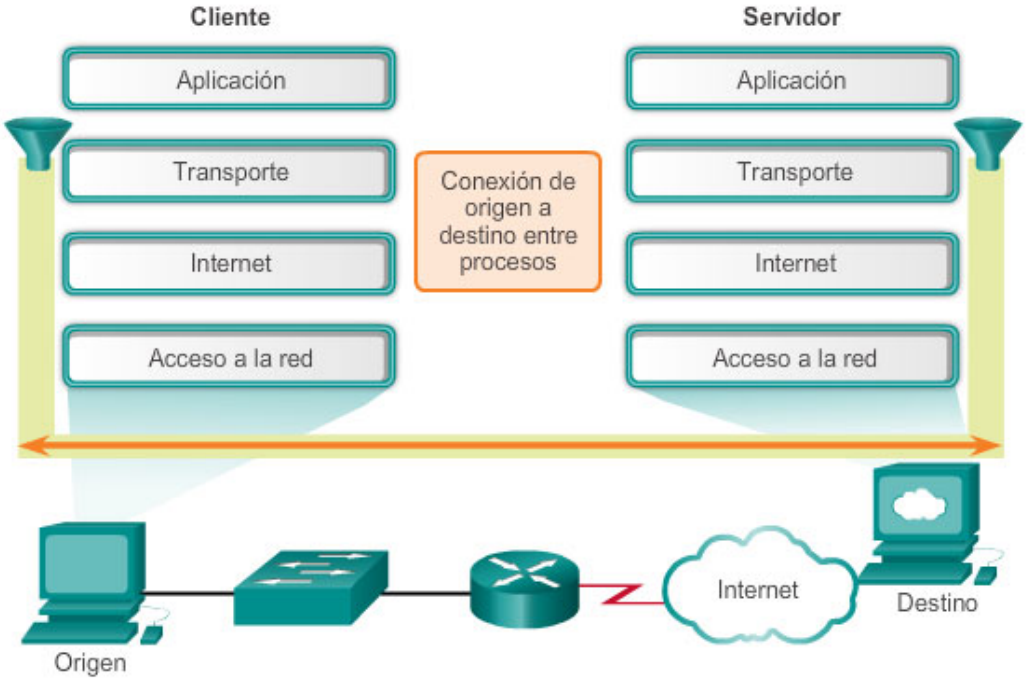
- Listado completo de la IANA:
  - https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

Número de puerto	Protocolo	Aplicación	Acrónimo
20	TCP	Protocolo de transferencia de archivos (datos)	FTP
21	TCP	Protocolo de transferencia de archivos (control)	FTP
22	TCP	Shell Seguro	SSH
23	TCP	Telnet	–
25	TCP	otocolo simple de transferencia de correo (Simple Mail Transfer Protocol)	SMTP
53	UDP, TCP	Servicio de nombres de dominios	DNS
67	UDP	Protocolo de configuración dinámica de host (servidor)	DHCP
68	UDP	Protocolo de configuración dinámica de host (cliente)	DHCP
69	UDP	Protocolo de transferencia de archivos trivial	TFTP
80	TCP	Protocolo de transferencia de hipertexto	HTTP
110	TCP	Protocolo de oficina de correos versión 3 (Post Office Protocol version 3)	POP3
143	TCP	Protocolo de acceso a mensajes de Internet (Internet Message Access Protocol)	IMAP
161	UDP	Simple Network Management Protocol	SNMP
443	TCP	Protocolo seguro de transferencia de hipertexto	HTTPS

Función de la capa de transporte

- Pares de sockets:
  - La combinación de la dirección IP de origen y el número de puerto de origen, o la dirección IP de destino y el número de puerto de destino, se conoce como “socket”.
  - Se combinan dos sockets para forma un **par de sockets**
  - Los sockets permiten que varios procesos que se ejecutan en un cliente y que varias conexiones a un proceso de servidor se distingan entre sí.

158.42.3.1 : 51123



158.42.4.23 : 80

Función de la capa de transporte

- El comando **netstat -a** permite mostrar un listado de las conexiones TCP/UDP abiertas

```
C:\>netstat

Active Connections

Proto     Local Address          Foreign Address            State
TCP       kenpc:3126             192.168.0.2:netbios-ssn    ESTABLISHED
TCP       kenpc:3158             207.138.126.152:http       ESTABLISHED
TCP       kenpc:3159             207.138.126.169:http       ESTABLISHED
TCP       kenpc:3160             207.138.126.169:http       ESTABLISHED
TCP       kenpc:3161             sc.msn.com:http            ESTABLISHED
TCP       kenpc:3166             www.cisco.com:http         ESTABLISHED

C:\>
```

- Las conexiones desconocidas pueden delatar una amenaza a la seguridad
- Práctica 4: Monitorización de puertos con netstat

Protocolo TCP: Comunicación confiable

- Confiabilidad en el protocolo TCP
  - Se establece una conexión inicial entre las partes para preparar el protocolo (enlace de tres vías)
  - Número de secuencia: da información sobre el número del primer byte que se está enviando
  - Acuse de recibo: siguiente byte esperado
  - Tamaño de ventana: cuántos bytes puedo enviar sin acuse de recibo

Protocolo TCP: Comunicación confiable

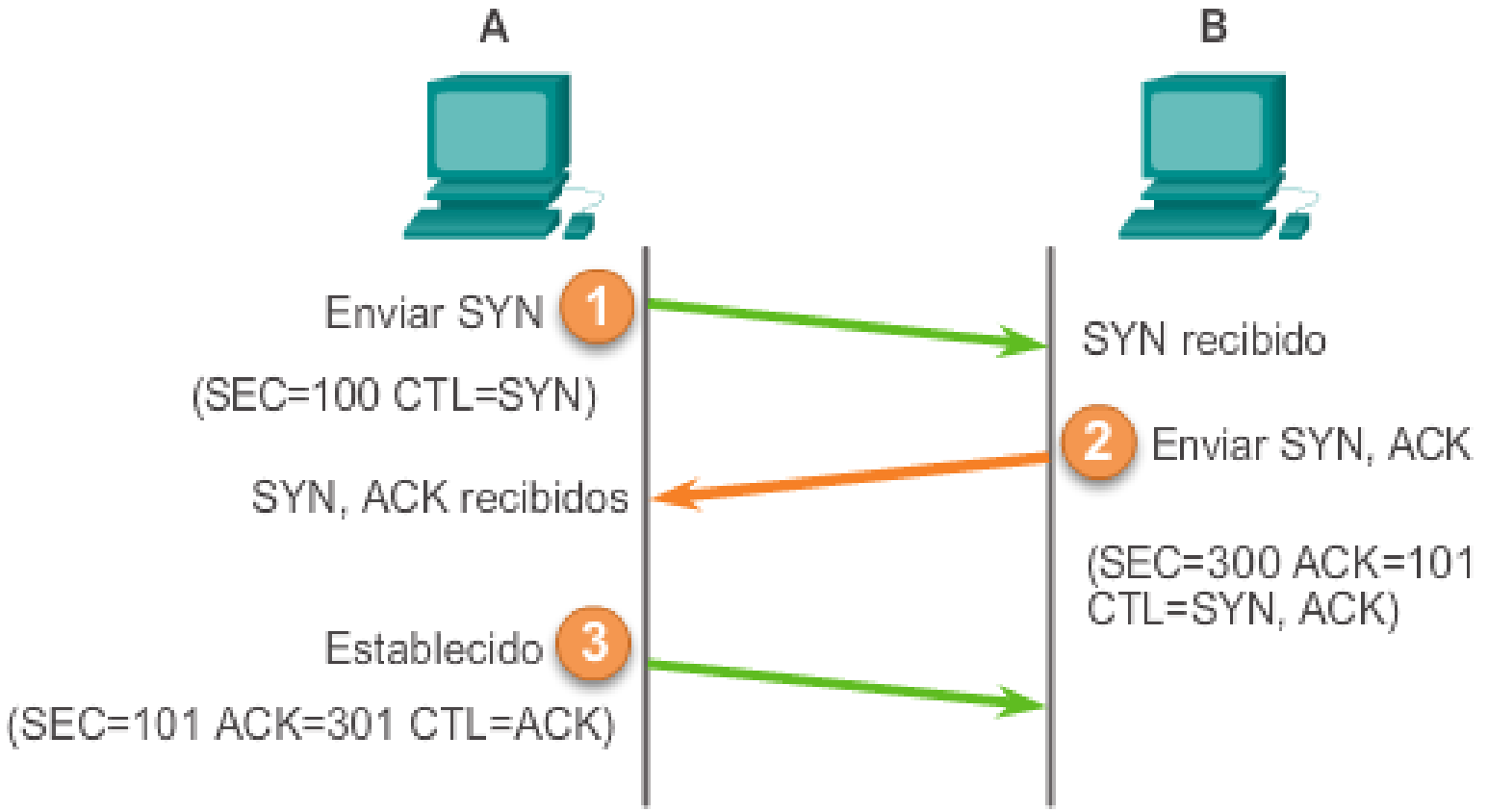
- Campos del encabezado TCP

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)																															
...	...	...																															

Protocolo TCP: conexión

- Establecimiento de conexiones TCP (**enlace de tres vías**, o de tres pasos, o *3-way handshake*):
  - El cliente de origen solicita una sesión de comunicación de cliente a servidor con el servidor.
  - El servidor acusa recibo de la sesión de comunicación de cliente a servidor y solicita una sesión de comunicación de servidor a cliente.
  - El cliente de origen acusa recibo de la sesión de comunicación de servidor a cliente.

Protocolo TCP: conexión



CTL = Bits de control establecidos en 1 en el encabezado TCP

Protocolo TCP: conexión

- Paso 1: Un navegador inicia una conexión con un servidor Web:
  - Puerto origen: 1069. Puerto destino: 80
  - Escoge un número de secuencia al azar (aunque en Wireshark se visualiza valor relativo: 0)
  - Activa el flag SYN
  - El encabezado tiene 28 bytes (en lugar de 20) porque en opciones se añade indicación del tamaño máximo de segmento,…


```
    13 6.201109   192.168.254.254   10.1.1.1          DNS    Standard query r
    14 6.202100   10.1.1.1          192.168.254.254   TCP    1069 > http [SYN
    15 6.202513   192.168.254.254   10.1.1.1          TCP    http > 1069 [SYN
    16 6.202543   10.1.1.1          192.168.254.254   TCP    1069 > http [ACK
    17 6.202651   10.1.1.1          192.168.254.254   HTTP   GET / HTTP/1.1

⊞ Frame 14 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: QuantaCo_bd:0c:7c (00:c0:9f:bd:0c:7c), Dst: Cisco_cf:66:40
⊞ Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 192.168.254.254 (192.168.2
⊟ Transmission Control Protocol, Src Port: 1069 (1069), Dst Port: http (80), S
    Source port: 1069 (1069)
    Destination port: http (80)
    Sequence number: 0    (relative sequence number)
    Header length: 28 bytes
  ⊟ Flags: 0x02 (SYN)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
```

Protocolo TCP: conexión

- Paso 2: El servidor acepta la conexión
  - Puerto origen: 80. Puerto destino: 1069
  - Escoge un número de secuencia al azar (aunque se visualiza valor relativo: 0)
  - Como reconocimiento: último número de secuencia recibido + 1 (se visualiza 1)
  - Activa los flags SYN y ACK


```
13 6.201109   192.168.254.254   10.1.1.1          DNS    Standard query
14 6.202100   10.1.1.1          192.168.254.254   TCP    1069 > http [S
15 6.202513   192.168.254.254   10.1.1.1          TCP    http > 1069 [S
16 6.202543   10.1.1.1          192.168.254.254   TCP    1069 > http [A
17 6.202651   10.1.1.1          192.168.254.254   HTTP   GET / HTTP/1.1

⊞ Frame 15 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: Cisco_cf:66:40 (00:0c:85:cf:66:40), Dst: QuantaCo_bd:0c:
⊞ Internet Protocol, Src: 192.168.254.254 (192.168.254.254), Dst: 10.1.1.1 (
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 1069 (1069),
    Source port: http (80)
    Destination port: 1069 (1069)
    Sequence number: 0    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 28 bytes
  ⊟ Flags: 0x12 (SYN, ACK)
```

Protocolo TCP: conexión

- Paso 3: El cliente acepta la conexión
  - Puerto origen: 1069 puerto destino: 80
  - Mismo número de secuencia que el reconocimiento recibido (se visualiza 1)
  - Como reconocimiento: último número de secuencia recibido +1 (se visualiza 1)
  - Activa el flag ACK

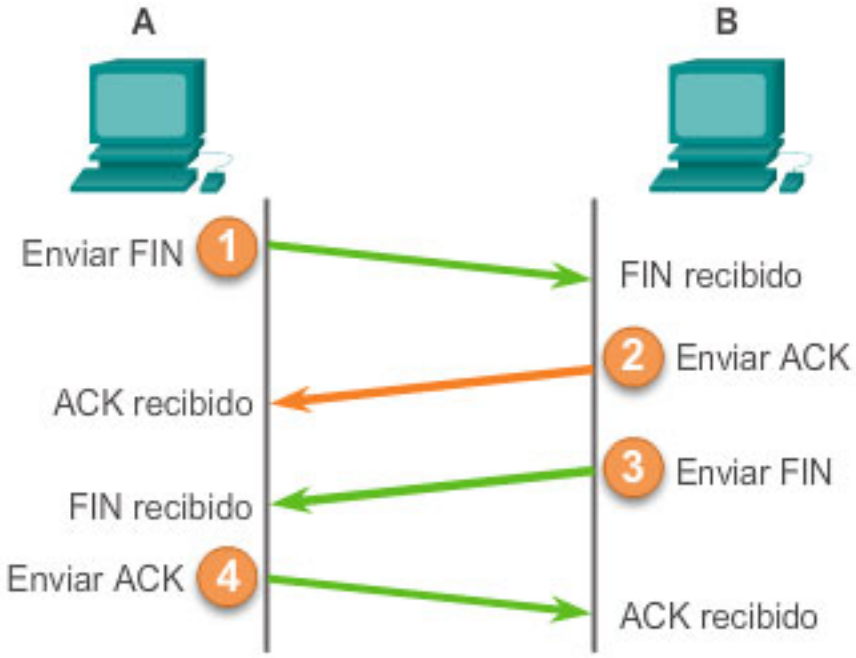
```
   13 6.201109   192.168.254.254   10.1.1.1          DNS    Standard query re
   14 6.202100   10.1.1.1          192.168.254.254   TCP    1069 > http [SYN]
   15 6.202513   192.168.254.254   10.1.1.1          TCP    http > 1069 [SYN,
   16 6.202543   10.1.1.1          192.168.254.254   TCP    1069 > http [ACK]
   17 6.202651   10.1.1.1          192.168.254.254   HTTP   GET / HTTP/1.1

⊞ Frame 16 (54 bytes on wire, 54 bytes captured)
⊞ Ethernet II, Src: QuantaCo_bd:0c:7c (00:c0:9f:bd:0c:7c), Dst: Cisco_cf:66:40
⊞ Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 192.168.254.254 (192.168.25
⊟ Transmission Control Protocol, Src Port: 1069 (1069), Dst Port: http (80), Se
    Source port: 1069 (1069)
    Destination port: http (80)
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x10 (ACK)
```

Protocolo TCP: conexión

- Análisis del enlace de tres vías de TCP:
  - Permite comprobar que el dispositivo de destino está presente en la red.
  - Verifica que el dispositivo de destino tenga un servicio activo y que acepte solicitudes en el número de puerto de destino que el cliente de origen desea utilizar.
  - Informa al dispositivo de destino que el cliente de origen intenta establecer una sesión de comunicación en dicho número de puerto

Protocolo TCP: desconexión

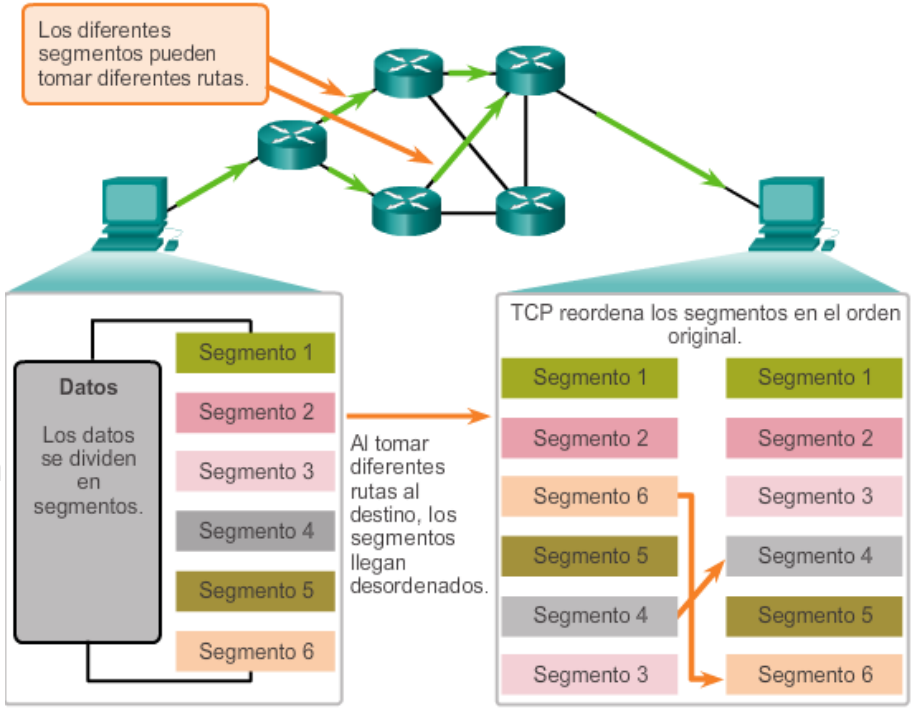
- Intercambio de señales para la finalización de conexión
  - A no tiene más datos: envía segmento con el flag FIN
    - B contesta con el flag ACK y en reconocimiento indica el número de secuencia recibido +1
    - B puede continuar enviando datos
  - B no tiene más datos: envía FIN
    - A contesta con …
  - Una vez reconocidos todos los segmentos, la sesión se cierra



Protocolos de la capa de transporte

Confiabilidad y control de flujo

- Confiabilidad de TCP: entrega ordenada
  - Los segmentos TCP utilizan números de secuencia para identificar exclusivamente a cada segmento y dar acuse de recibo de ellos, hacer un seguimiento del orden de segmentos e indicar la forma en que se vuelven a reordenar los segmentos recibidos.
  - Durante la configuración de la sesión TCP, se elige un número de secuencia inicial (ISN) al azar. Después, el ISN **se incrementa con el número de bytes transmitidos**.
  - El proceso de TCP receptor reúne los datos de los segmentos en el búfer hasta que se reciban y se vuelvan a armar todos los datos.
  - Los segmentos que no se reciben en el orden correcto se conservan para su posterior procesamiento.
  - Los datos se entregan a la capa de aplicación sólo cuando se hayan recibido y vuelto a reordenar por completo.



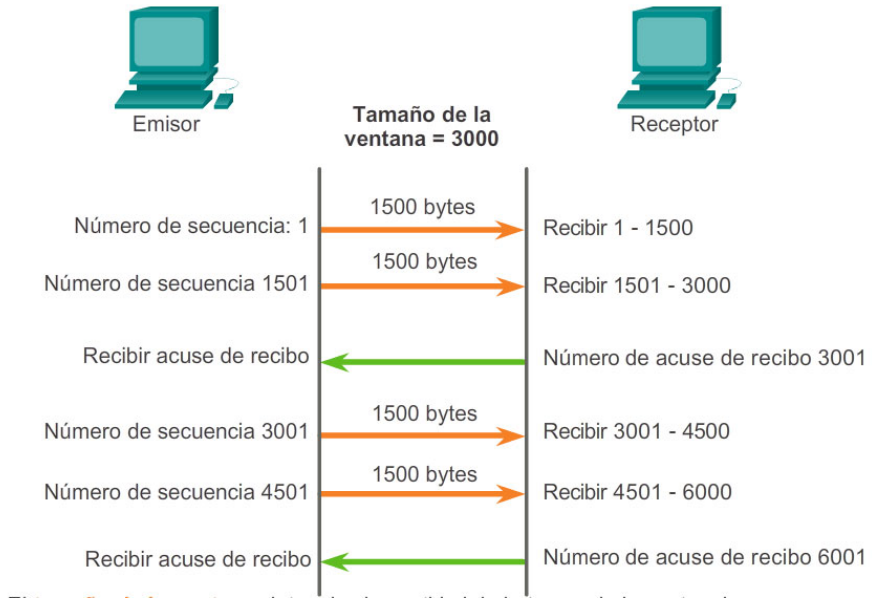
Tamaño Máximo de Segmento (*Maximum Segment Size* o MSS)

Tamaño más grande de datos (especificado en bytes) que un dispositivo puede recibir en un único trozo, sin fragmentar

Protocolos de la capa de transporte

Confiabilidad y control de flujo

- Control del flujo en TCP: tamaño de la ventana y acuses de recibo
  - TCP también proporciona mecanismos para el control del flujo.
  - El control del flujo asegura que las terminales TCP puedan recibir y procesar datos de manera confiable.
  - TCP ajusta la velocidad del flujo de datos entre el origen y el destino en una sesión determinada.
  - La función de control del flujo en TCP depende de un campo del encabezado TCP de 16 bits denominado **Tamaño de ventana**. El tamaño de ventana es la cantidad de bytes que el dispositivo de destino de una sesión TCP puede aceptar y procesar al mismo tiempo.
  - El origen y el destino TCP acuerdan el tamaño de ventana inicial cuando se establece la sesión TCP.
  - De ser necesario, los terminales TCP pueden ajustar el tamaño de ventana durante una sesión.

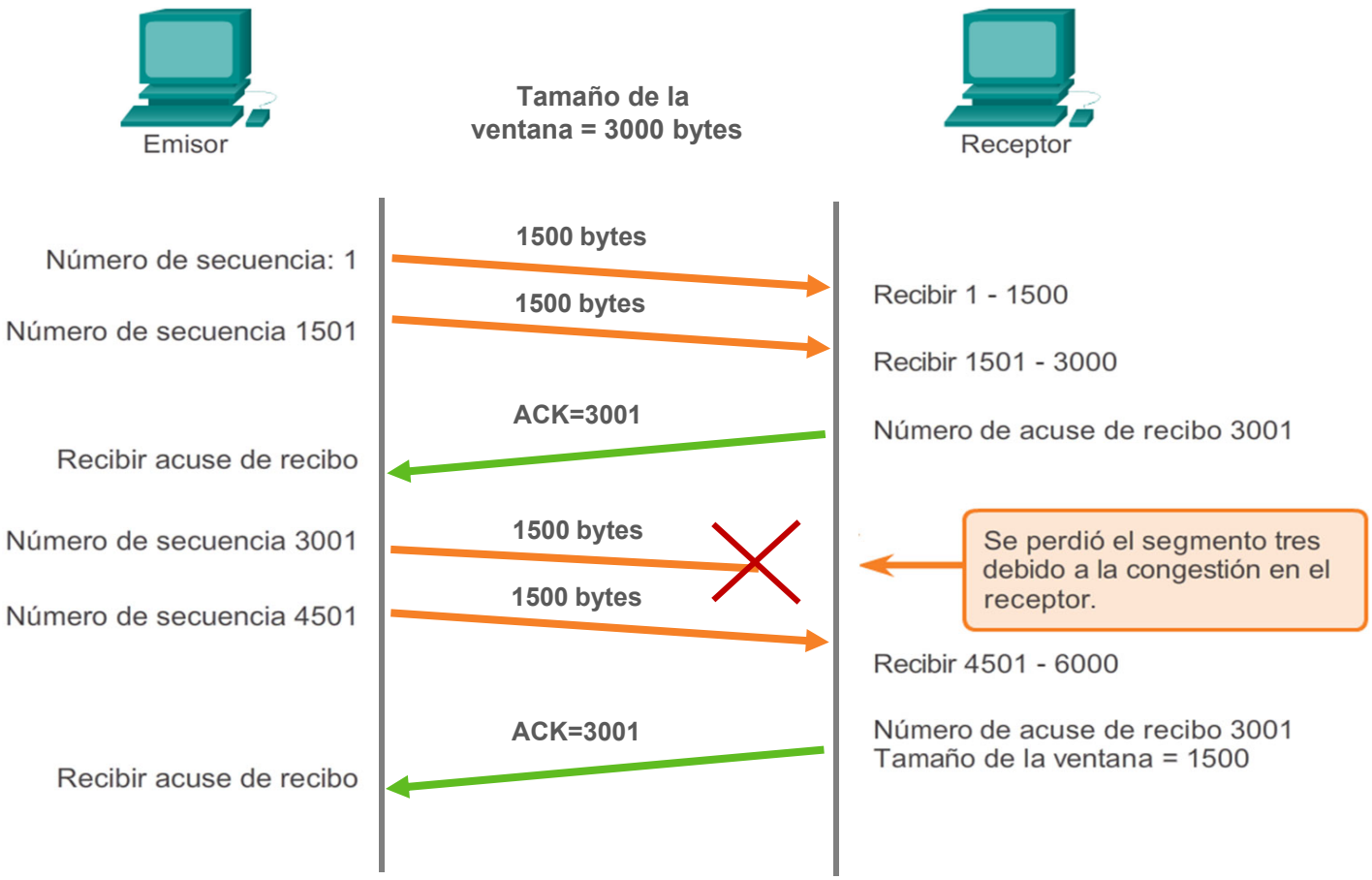


Unidad máxima de transferencia (MTU)

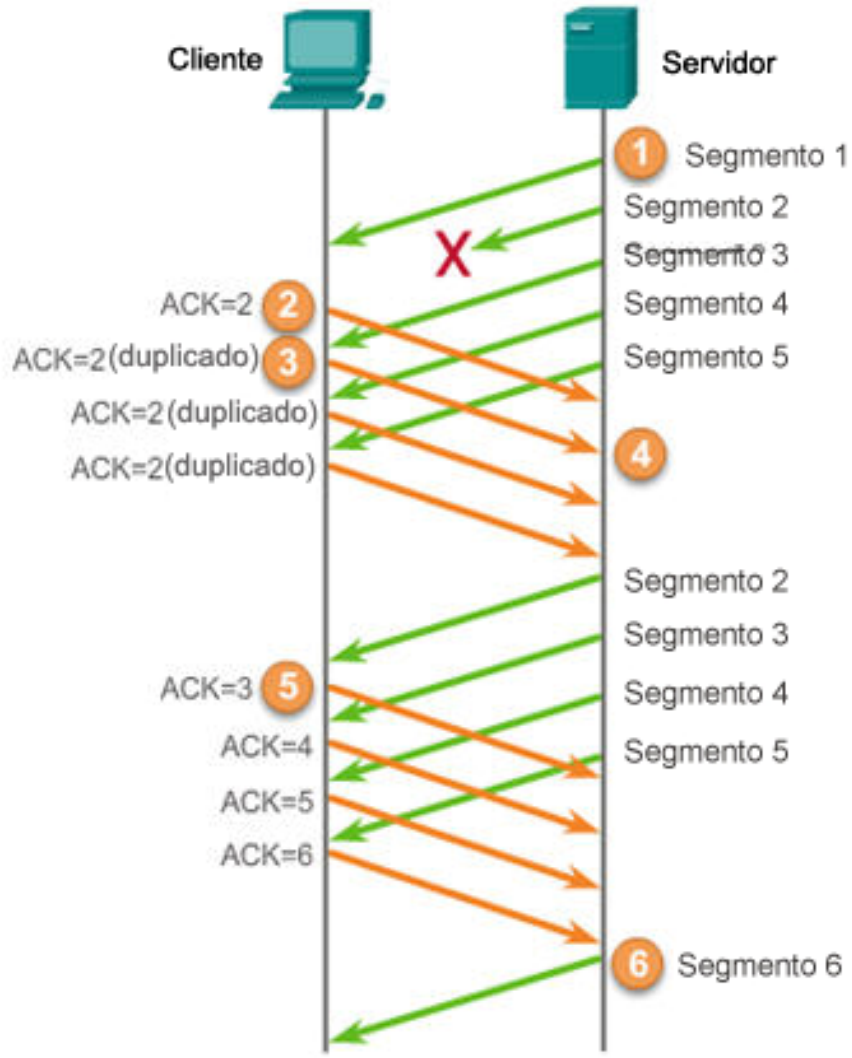
MTU = MSS + cabecera TCP o UDP + cabecera IP

Depende de la red y habitualmente en Ethernet es de **1500** bytes

Confiabilidad de TCP: pérdida de datos



Protocolo TCP: pérdida de datos



Protocolo TCP: prevención de congestiones

- La congestión de red suele dar como resultado el descarte de paquetes
- Los segmentos TCP que no se entregan activan la retransmisión. Pero la retransmisión de segmentos TCP puede empeorar la congestión
- El destino suele desconocer la congestión de red y considera que no es necesario sugerir un nuevo tamaño de ventana
- Para evitar y controlar la congestión, el origen puede estimar un nivel determinado de congestión de red al observar la velocidad a la que se reciben los segmentos TCP con acuse recibo
- **Nota**: La explicación de los mecanismos, temporizadores y algoritmos reales de manejo de la congestión se encuentra fuera del alcance de este curso

Núm. secuencia, reconocimiento y tamaño de ventana

Datos (25 bytes)

10 bytes	10 bytes	5 byt

sec	ack	vent.	señalizad.
10			SYN

sec	ack	vent.	señalizad.
55	11	20	ACK SYN

sec	ack	vent.	señalizad.
11	56	20	ACK

sec	ack	vent.	datos
11	56	20	10 bytes

sec	ack	vent.	datos
21	56	20	10 bytes

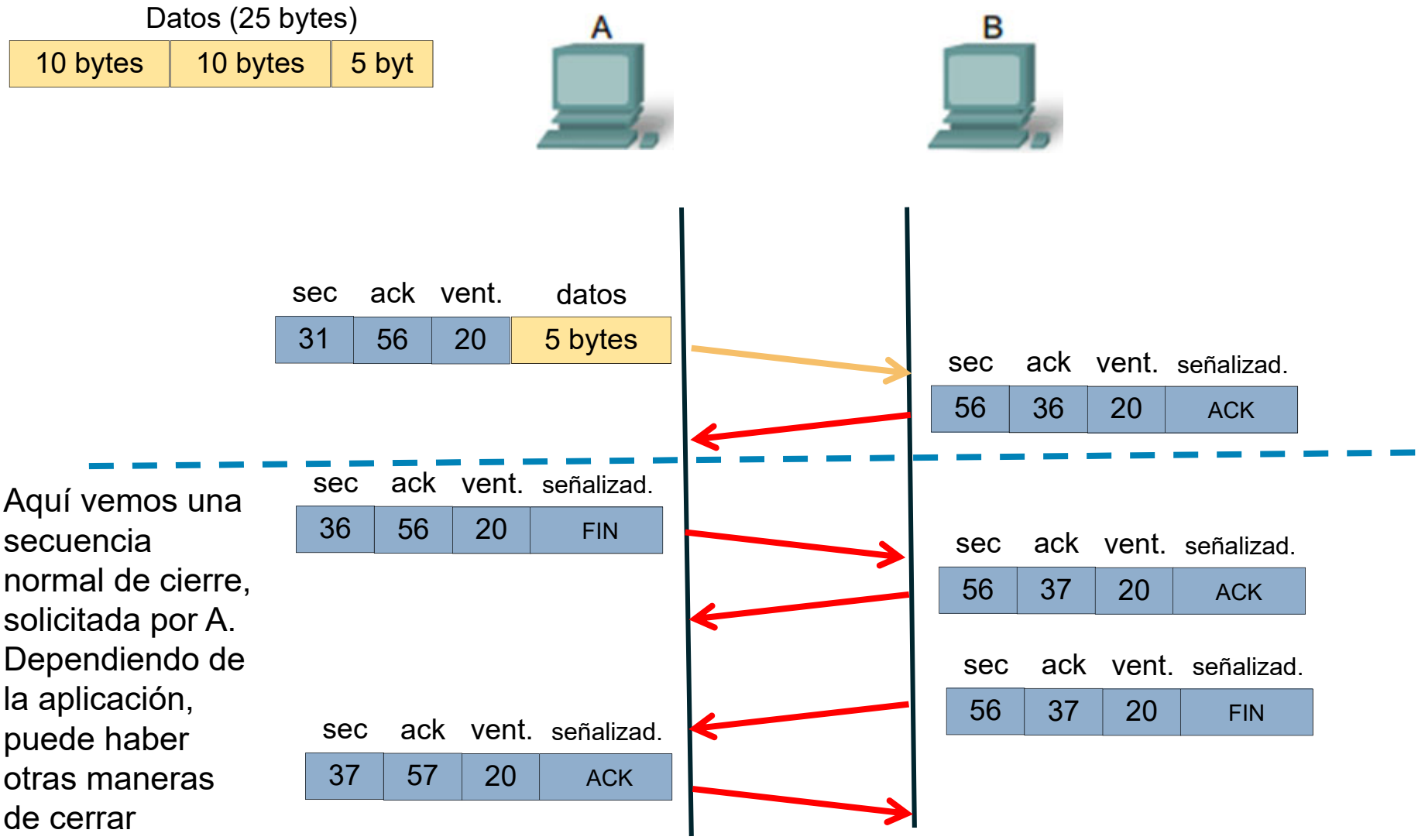
sec	ack	vent.	señalizad.
56	31	20	ACK

sec	ack	vent.	datos
31	56	20	5 bytes

sec	ack	vent.	señalizad.
56	36	20	ACK

Como sólo transmite A, en la fase de establecimiento no aumenta el ACK

Núm. secuencia, reconocimiento y tamaño de ventana

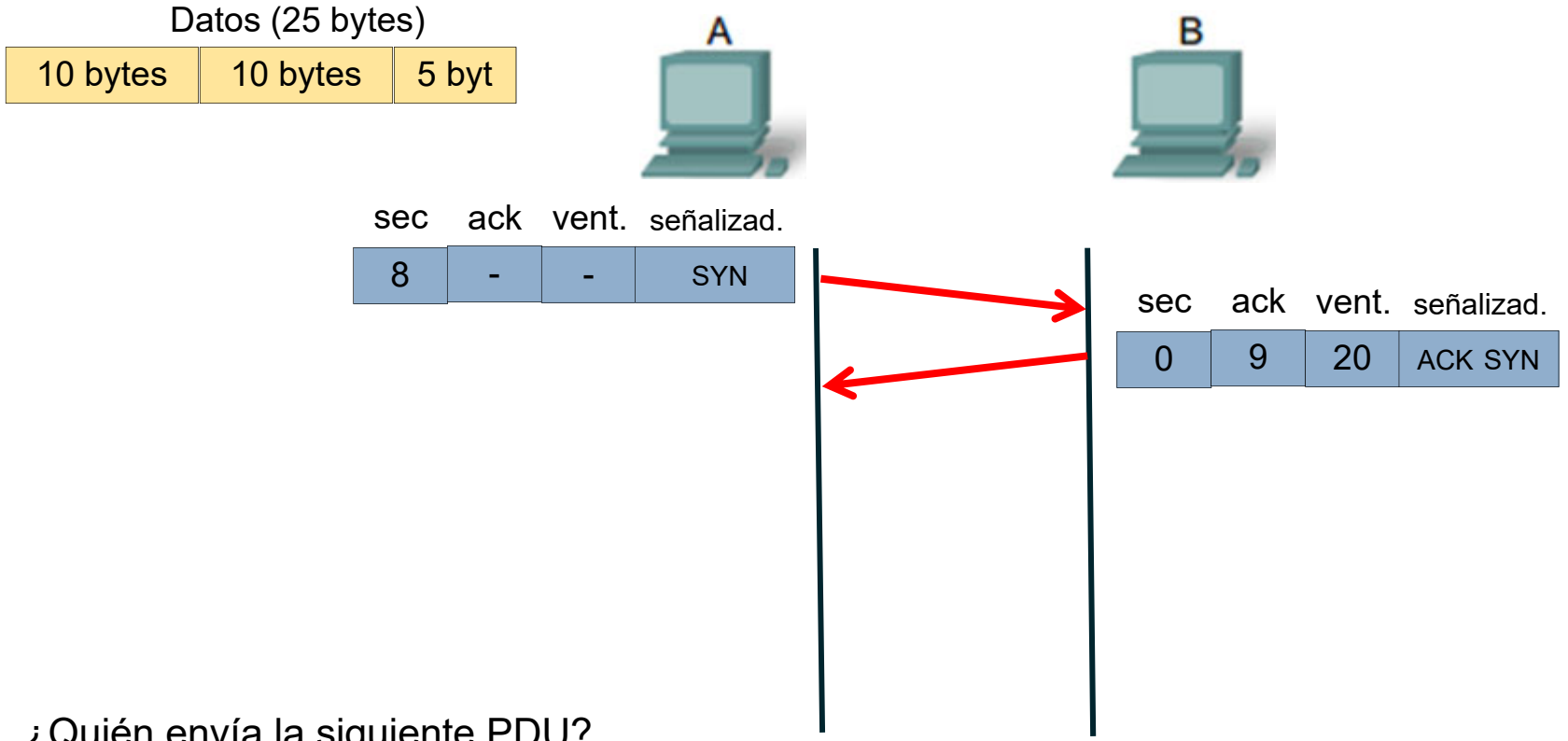


Aquí vemos una secuencia normal de cierre, solicitada por A. Dependiendo de la aplicación, puede haber otras maneras de cerrar

Núm. secuencia, reconocimiento y tamaño de ventana

- Reconocimiento (ACK)
  - Se va indicando en el ACK el siguiente número de secuencia que se espera después del que se acepta
- Número de secuencia:
  - En general, el número de secuencia no se incrementa si el anterior segmento simplemente es un ACK. Por tanto, vemos que:
    - En la fase de conexión se incrementa en uno por parte del que inicia la conexión
    - En la fase de establecimiento, sólo lo incrementa quién envíe datos, en tanta cantidad como bytes vaya enviando
    - En la fase de desconexión se incrementa en uno por parte de quien inicia la desconexión

Ejercicio: fase de conexión



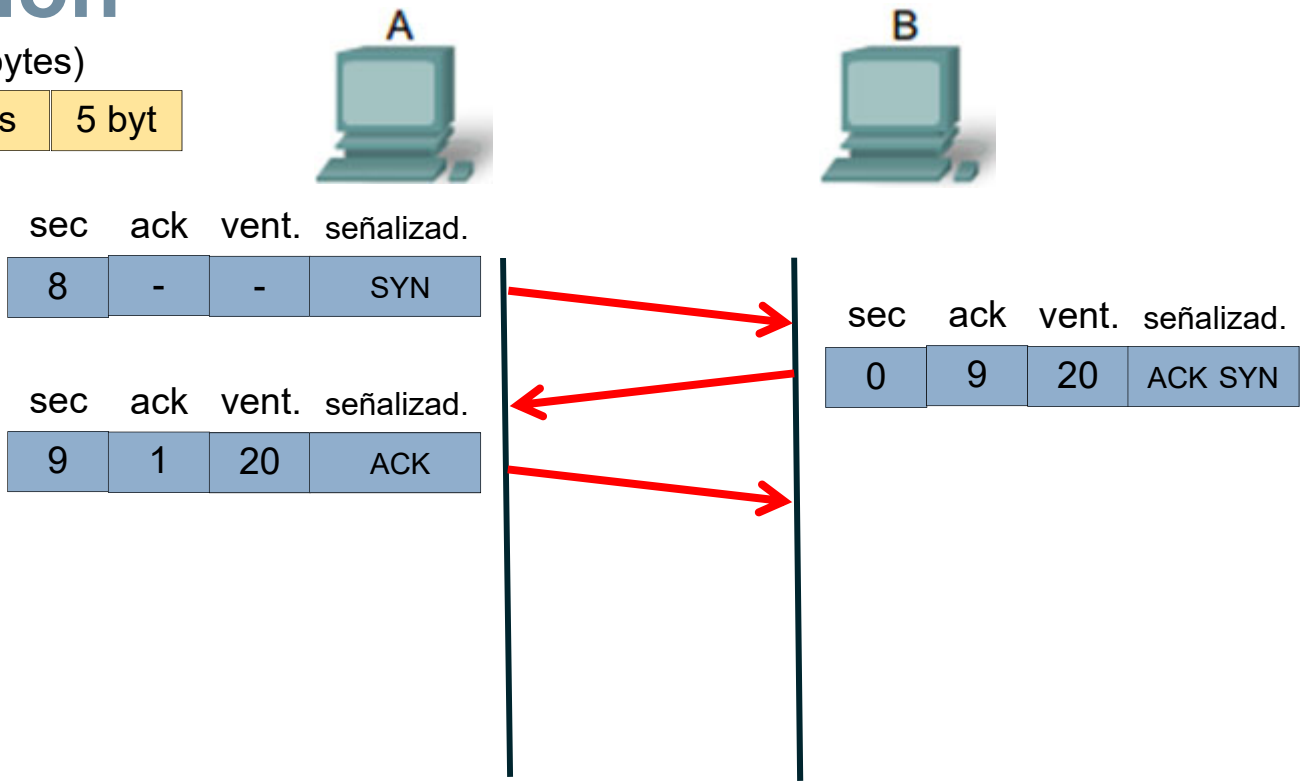
¿Quién envía la siguiente PDU?

¿Qué valores tendrán los campos del encabezado?

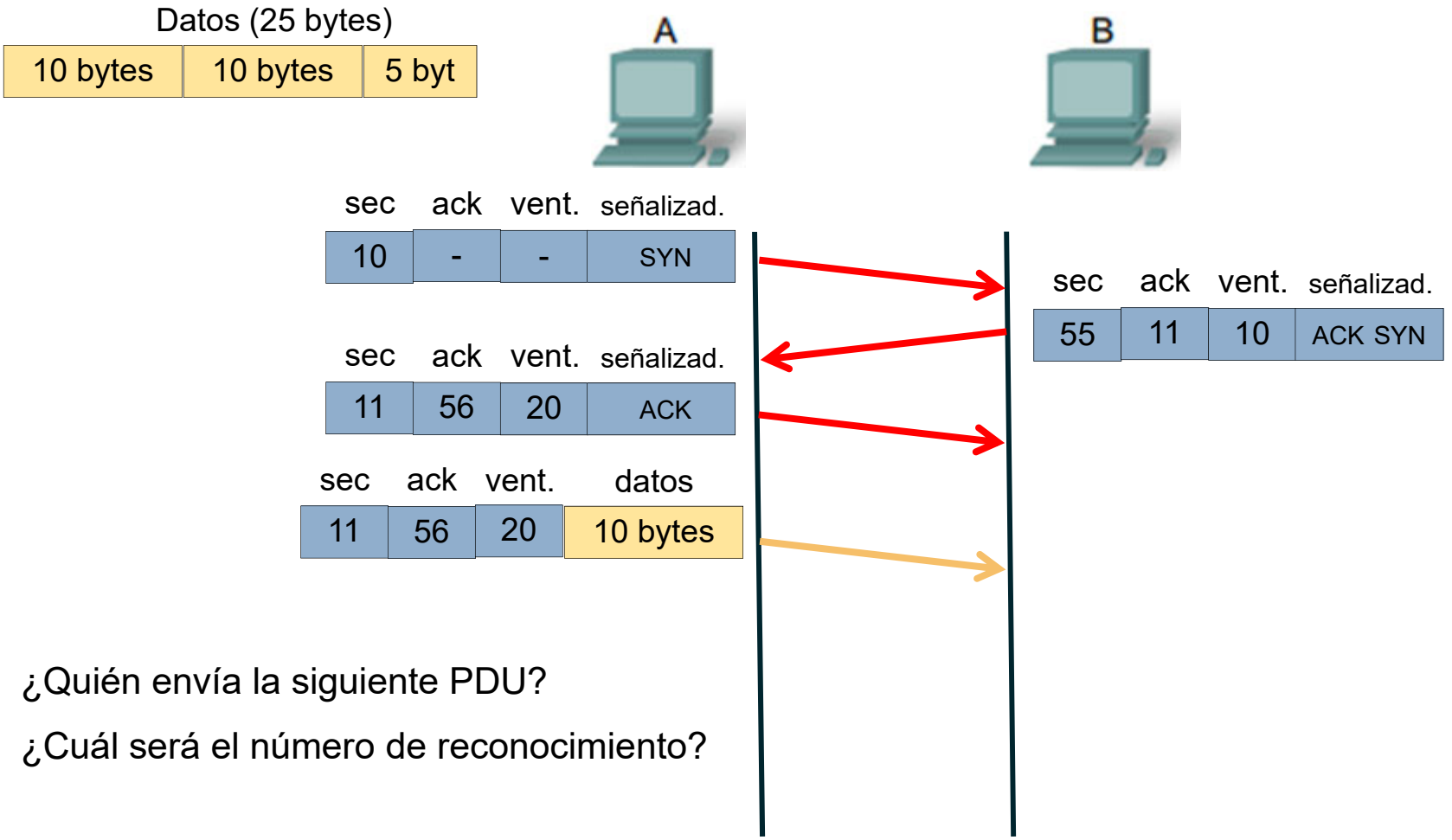
Solución

Datos (25 bytes)

10 bytes	10 bytes	5 byt



Ejercicio: fase de establecimiento



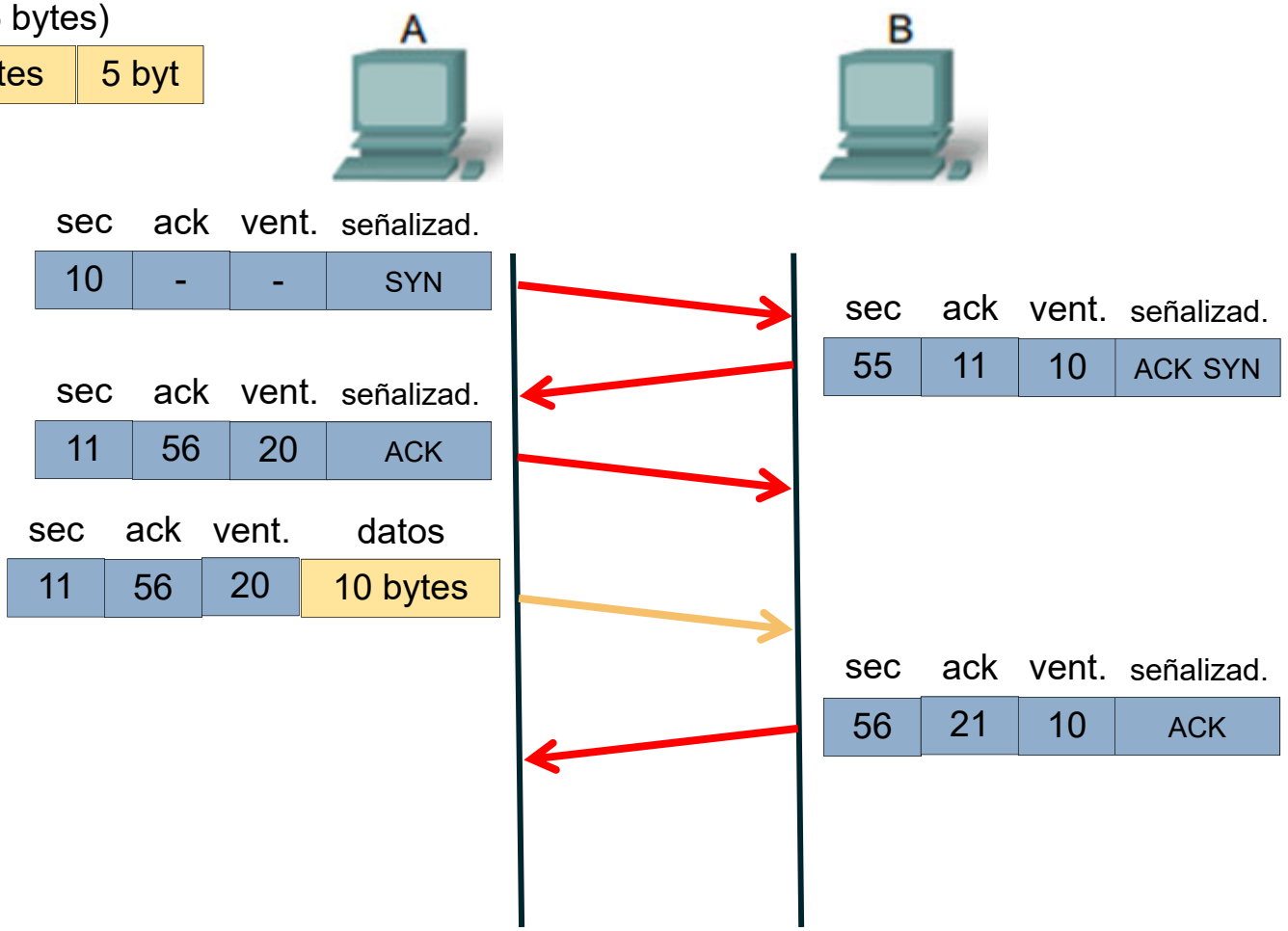
¿Quién envía la siguiente PDU?

¿Cuál será el número de reconocimiento?

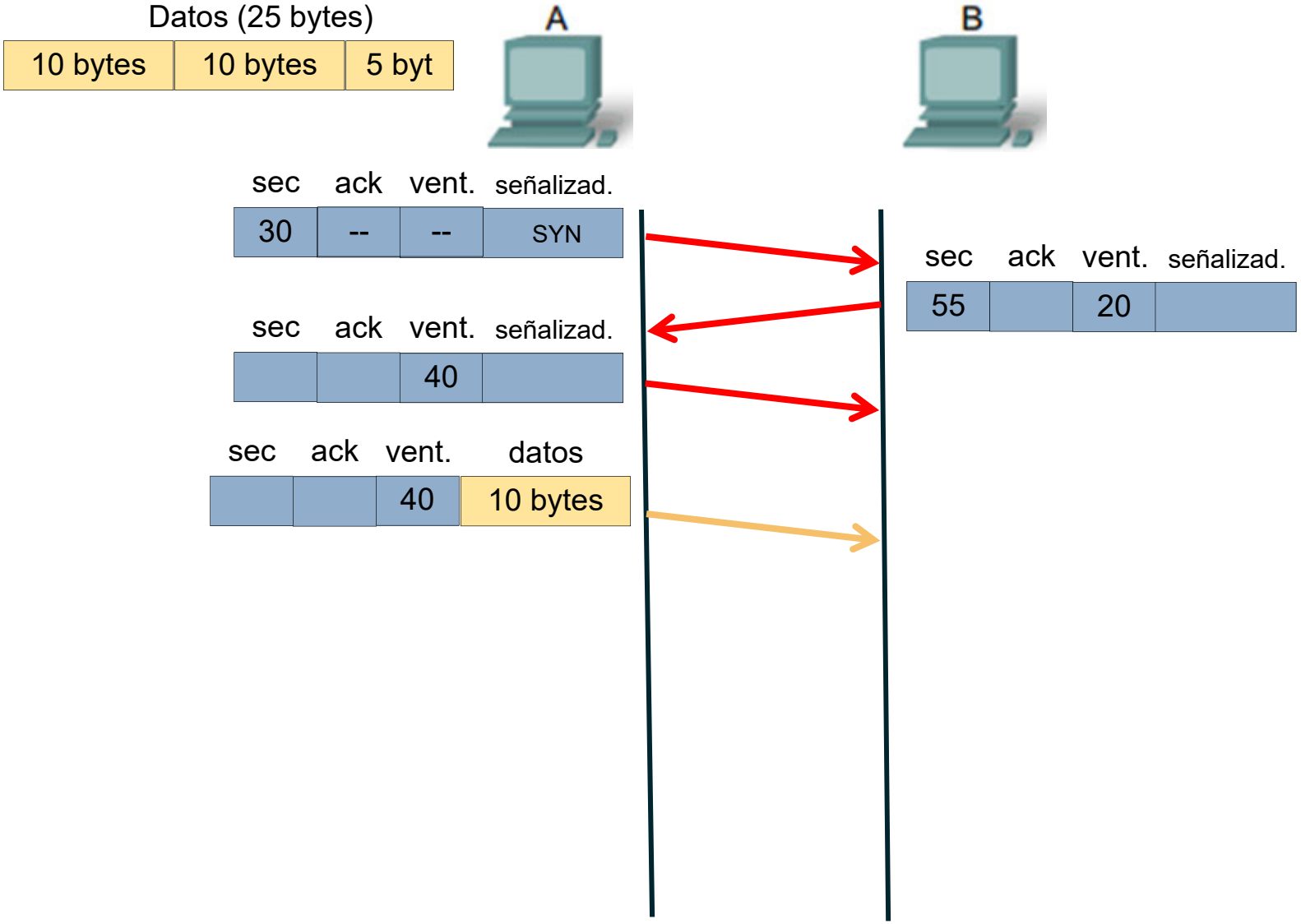
Solución

Datos (25 bytes)

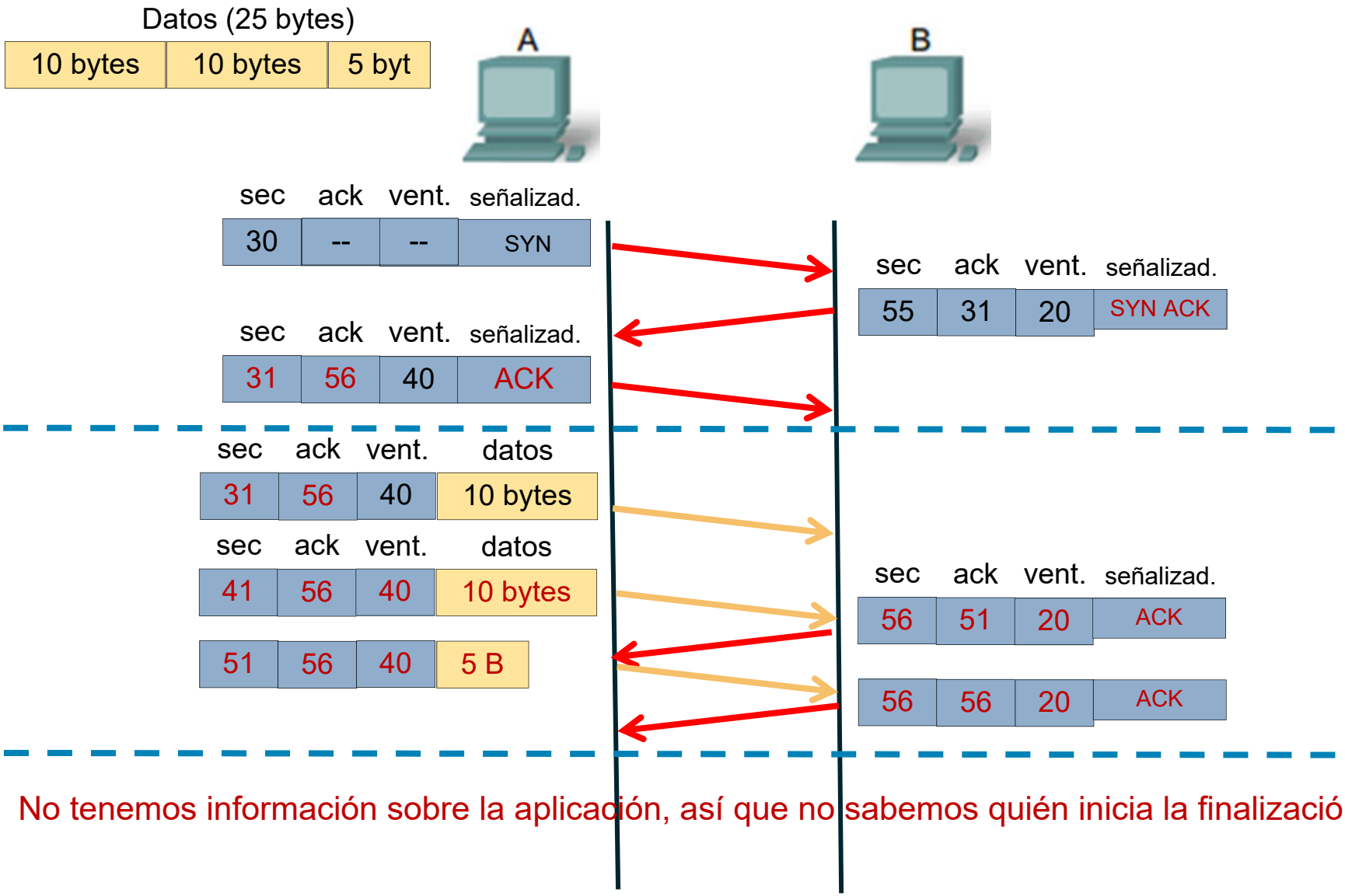
10 bytes	10 bytes	5 byt



Ejercicio: Rellena número de secuencia, reconocimiento y señalización, y continúa el diagrama

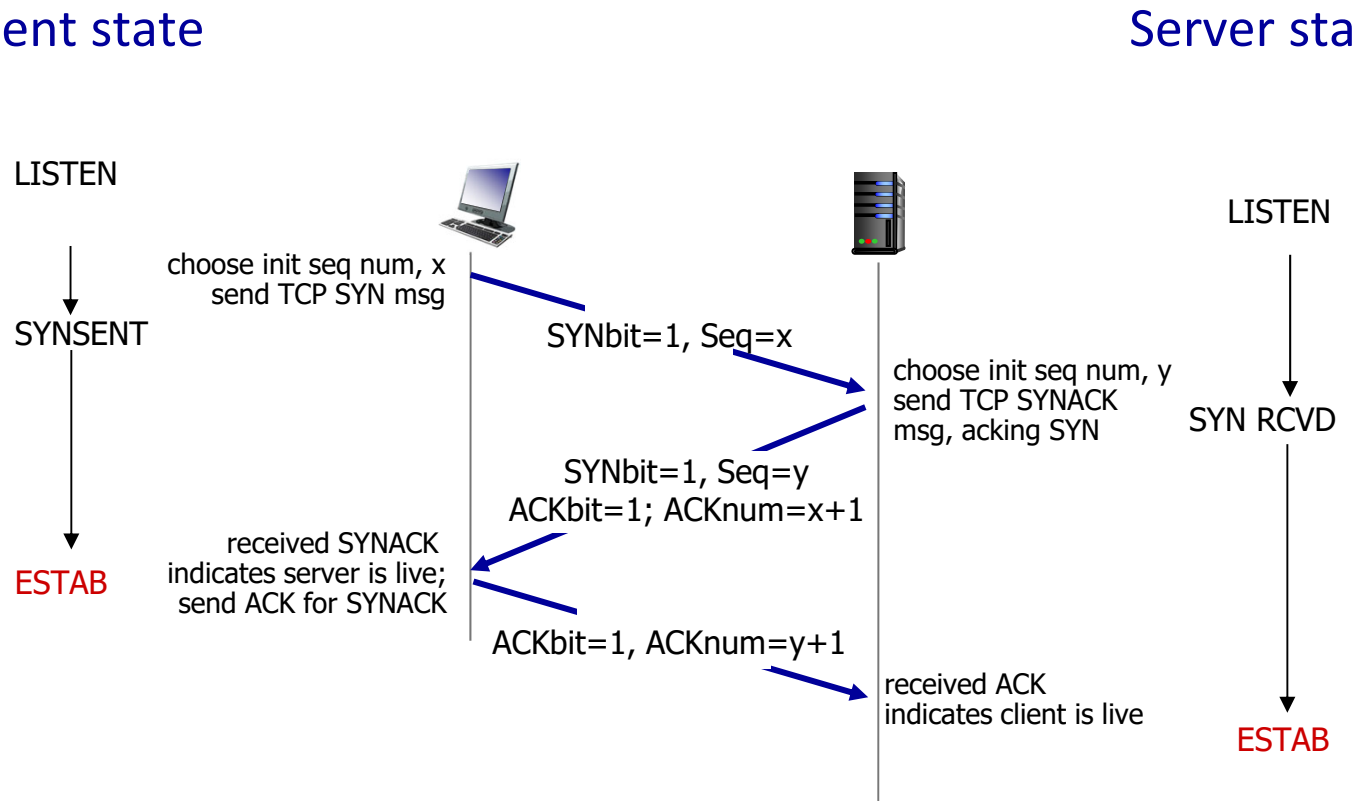


Solución

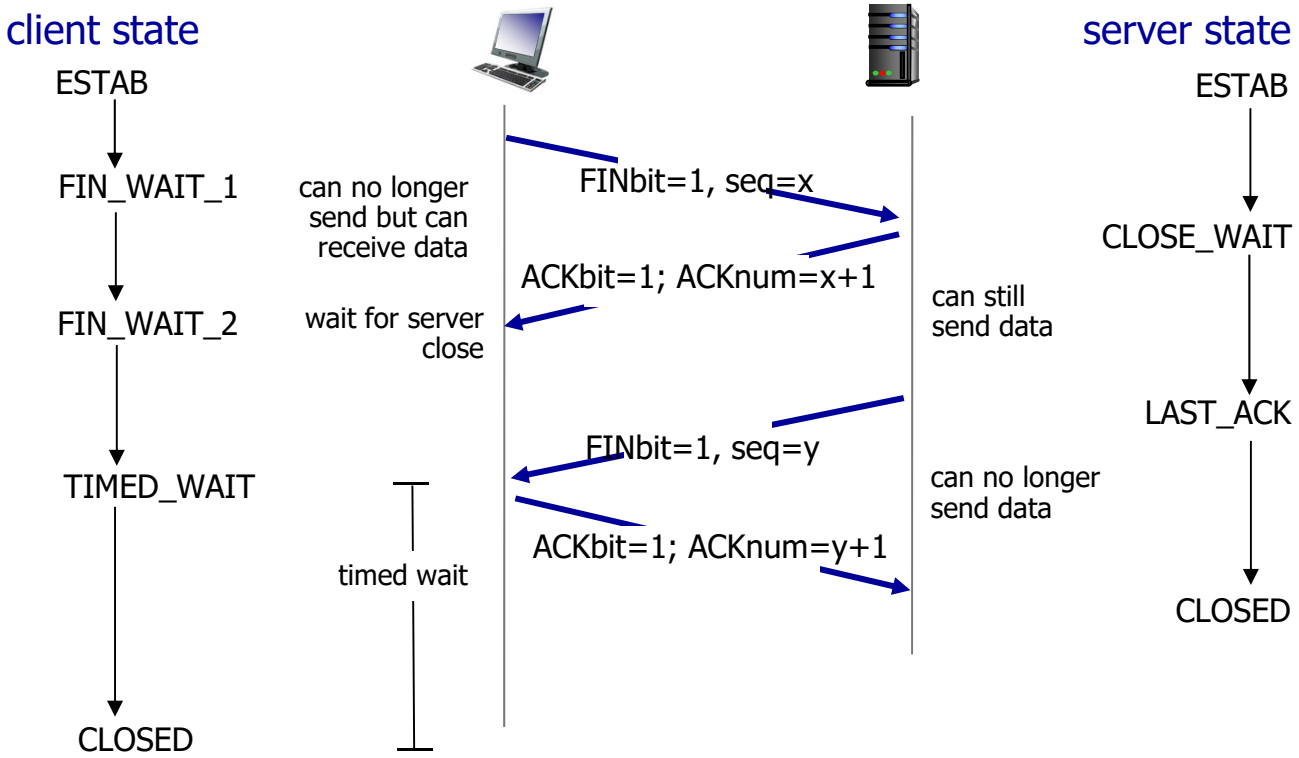


No tenemos información sobre la aplicación, así que no sabemos quién inicia la finalización

TCP 3-way handshake



Closing a TCP connection

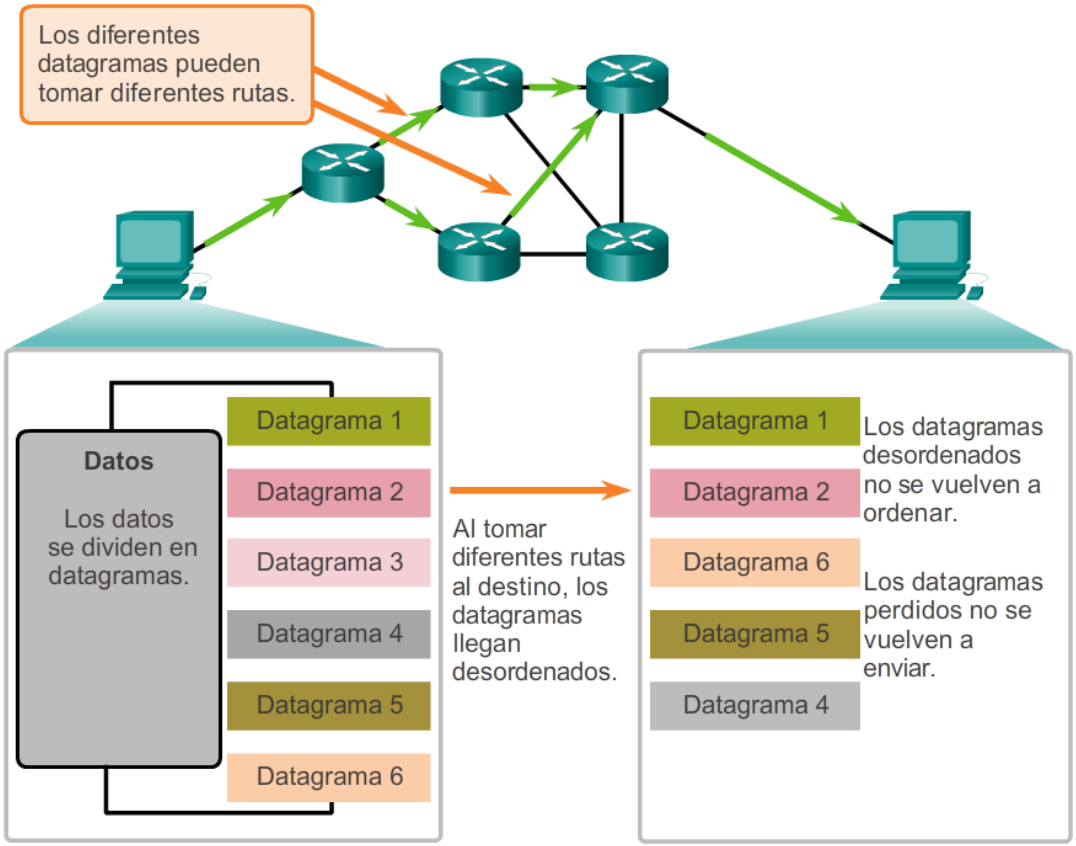


Protocolo UDP

- Protocolo simple que no ofrece las ventajas de confiabilidad de TCP
- Sólo separa conversaciones (puertos)
- Lo utilizan las aplicaciones que no pueden tolerar retrasos (juegos, IPTV, VoIP)
- También aplicaciones que pueden tolerar una pequeña pérdida de datos (juegos)
- O aplicaciones que no quieren demasiada sobrecarga. Ejemplo: una app de envío de posición GPS:
  - TCP: 3 segmentos de conexión, 1 de datos, 1 ACK y 4 de desconexión
  - UDP: 1 datagrama con los datos

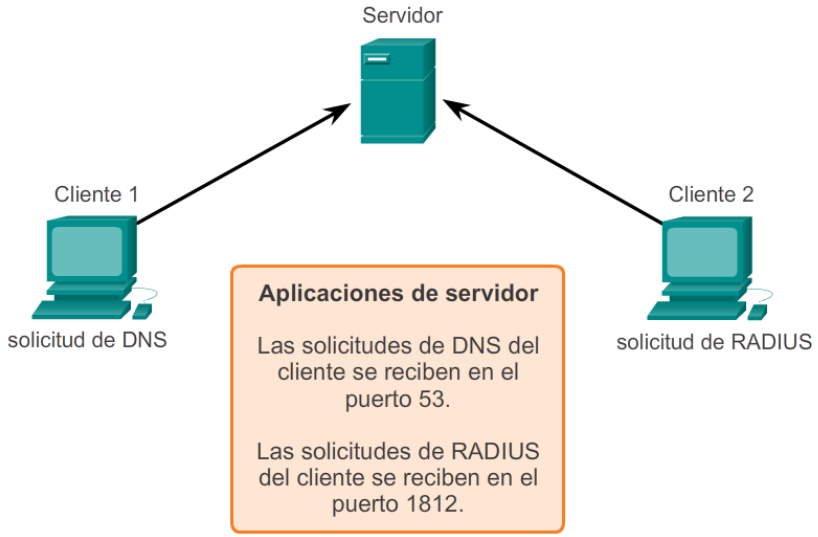
Protocolo UDP: entrega no ordenada

- Datagramas UDP
  - UDP simplemente entrega los datos en el orden en el que se recibieron.
  - Si es necesario, la aplicación debe identificar la secuencia correcta.



Protocolo UDP: separación conversaciones

- Procesos y solicitudes del servidor UDP
  - A las aplicaciones de servidor basadas en UDP se les asignan números de puerto conocidos o registrados.
  - Las solicitudes que se reciben en un puerto específico se reenvían a la aplicación adecuada según los números de puerto.

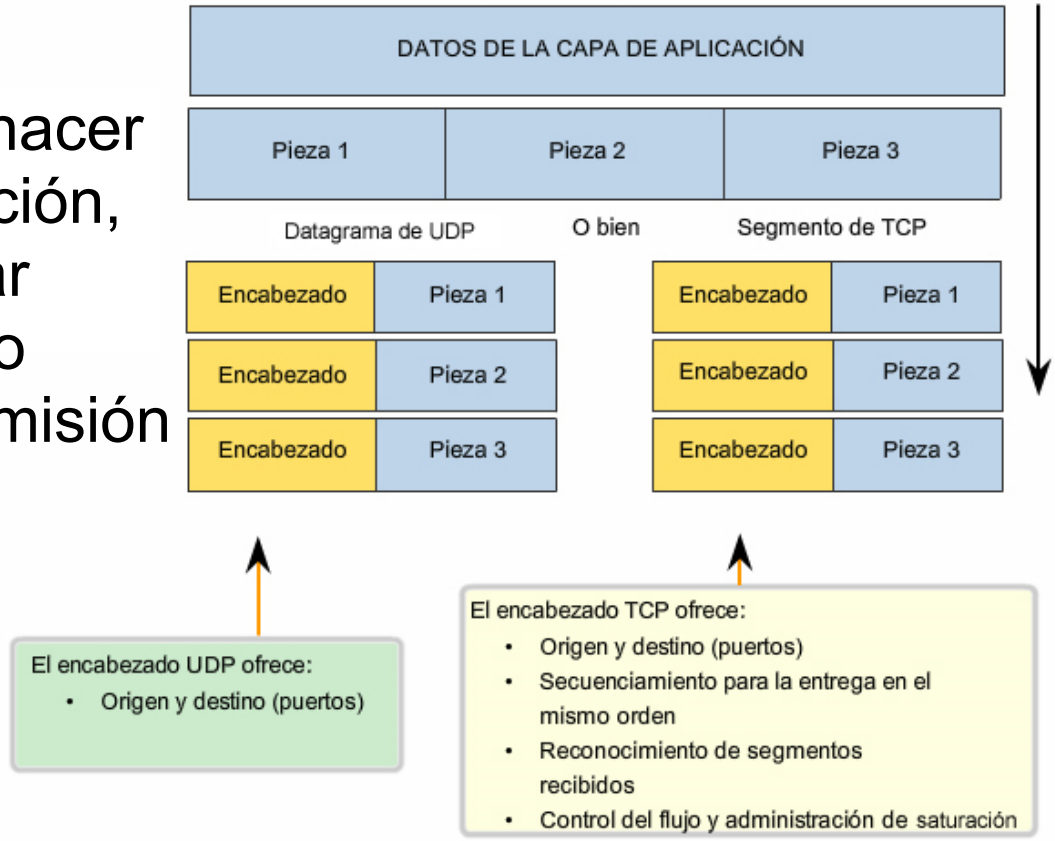


Protocolo UDP: separación conversaciones

- Procesos de cliente UDP
  - La comunicación entre cliente y servidor UDP también se inicia con una aplicación cliente.
  - Al cliente UDP se le asigna de manera dinámica un número de puerto origen.
  - Por lo general, el puerto de destino es el número de puerto conocido o registrado que asignado al proceso de servidor.
  - Se utiliza el mismo par de puertos de origen o destino en el encabezado de todos los datagramas usados en la transacción.
  - En la devolución de datos del servidor al cliente, se invierten los números de puerto de origen y de destino en el encabezado del datagrama.

Función de la capa de transporte

- Segmentación y reensamblaje en UDP y TCP
  - En TCP es la capa de transporte la que segmenta y reensambla
  - En UDP lo debe hacer la capa de aplicación, que debe entregar piezas del tamaño correcto en transmisión

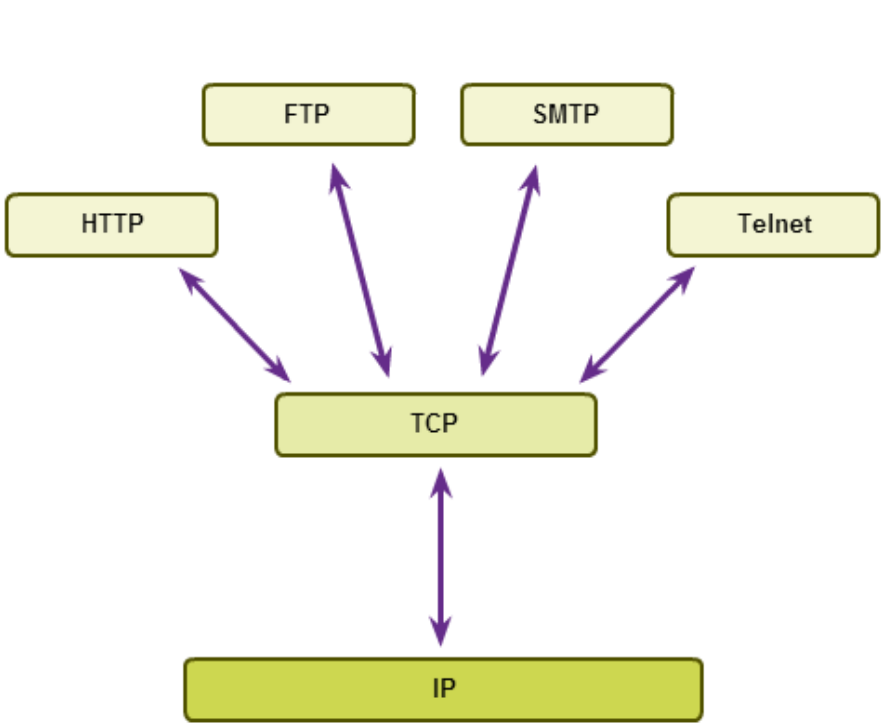


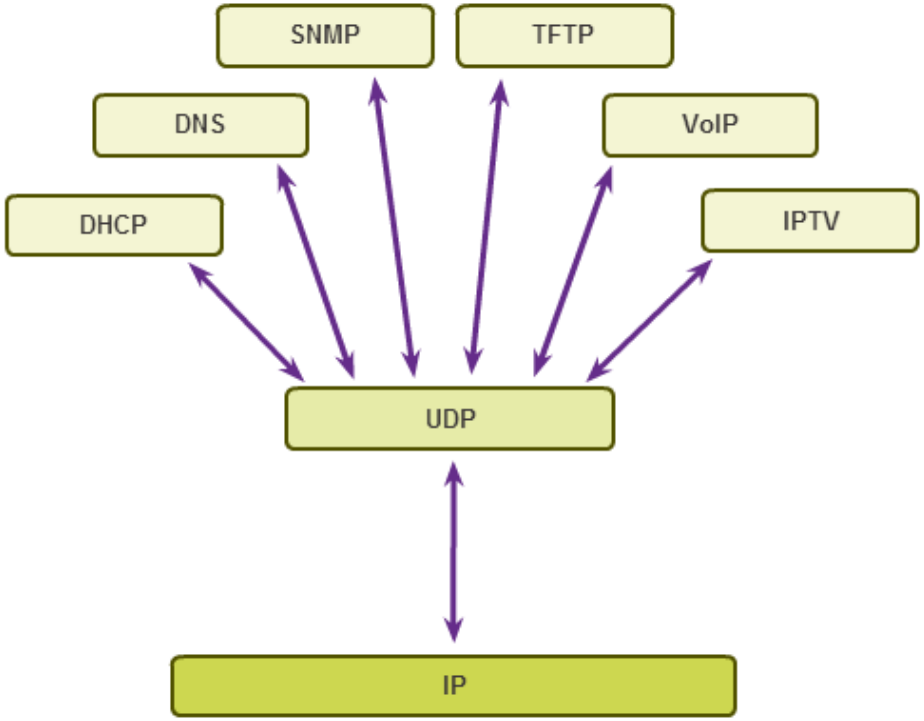
Protocolo UDP: encabezado

- Tamaño: 32 bits (8 bytes)
- El puerto de origen puede ser opcional en aquellos protocolos que no exijan respuesta
- La suma de verificación es siempre opcional (hay medidas adicionales que garantizan la integridad)
- Si no se usan, se ponen a cero

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

Protocolos de aplicación: UDP o TCP





Protocolos de la capa de transporte

TCP o UDP

- Aplicaciones que utilizan TCP
  - TCP maneja todas las tareas relacionadas con la capa de transporte.
  - Esto hace que la aplicación no tenga que administrar ninguna de dichas tareas.
  - Las aplicaciones simplemente pueden enviar el flujo de datos a la capa de transporte y utilizar los servicios de TCP.
- Aplicaciones que utilizan UDP
  - Aplicaciones multimedia en vivo: pueden tolerar cierta pérdida de datos, pero requieren demoras breves o que no haya demoras. Los ejemplos incluyen VoIP y la transmisión de vídeo en vivo.
  - Aplicaciones con solicitudes y respuestas simples: aplicaciones con transacciones simples en las que un host envía una solicitud y existe la posibilidad de que reciba una respuesta o no. Los ejemplos incluyen DNS y DHCP. DNS también puede utilizar TCP si la respuesta tiene más de 512 bytes
  - Aplicaciones que manejan la confiabilidad por sí mismas: comunicaciones unidireccionales en las que no se requiere control de flujo, detección de errores, acuses de recibo ni recuperación de errores, o en las que la aplicación pueda ocuparse de estas tareas. Los ejemplos incluyen SNMP y TFTP.

Ejercicio

- Queremos implementar un protocolo a nivel de aplicación entre un móvil y un servidor:
  - ¿Usamos TCP o UDP?
    - Transferir una foto
    - Apuesta online
    - Mandar una posición GPS cada 5 segundos
    - Walkie talkie
    - Mensajería por voz (ej.: audio de WhatsApp)
    - Videojuego online

Cálculo de sobrecarga según protocolo

- Queremos implementar un protocolo a nivel de aplicación que envíe la posición GPS de un móvil cada minuto. Queremos realizar un estudio sobre qué protocolo de transporte hemos de utilizar. Se han de enviar 4 datos: la longitud y latitud son codificadas mediante el tipo *double* de Java; el tiempo en que se envía la posición y el identificador del móvil se codifican mediante mediante dos *long*. ¿Cuántos bytes son enviados por minuto si utilizamos TCP y cuántos si utilizamos UDP?
  - Cabecera TCP: 20 bytes (28 en los SYN)
  - Cabecera UDP: 8 bytes
  - Cabecera IP: 20 bytes
  - Cabecera 4G: 34 bytes
  - Tamaño de *double* y *long*: 8 bytes
  - Recordatorio: En TCP se ha de abrir y cerrar una conexión cada vez que se envía una posición

Capa Transporte: Resumen

- Funciones principales:
  - Separación de comunicaciones entre aplicaciones (puertos)
  - Segmentación de datos y reensamblado
  - Cubrir diferentes requisitos de aplicación (TCP/UDP)
  - Conversaciones orientadas a la conexión
  - Entrega confiable
  - Control del flujo
- TCP implementa todas estas funciones
  - Número de secuencia, acuse de recibo, tamaño de ventana, reenvío,…
- Mientras que UDP
  - Sólo separación de comunicaciones (puertos)
  - Menos sobrecarga (DNS, DHCP, ejemplo GPS)
  - No introduce retrasos (VoIP, IPTV, juegos en línea)

Rellena los estados del protocolo TCP según netstat