# Tema 4
# Nivel de Transporte y Aplicación

**Redes de Computadores**

**Curso 2017/2018**
**Segundo Semestre**

# Índice

# 4.1 Nivel de transporte

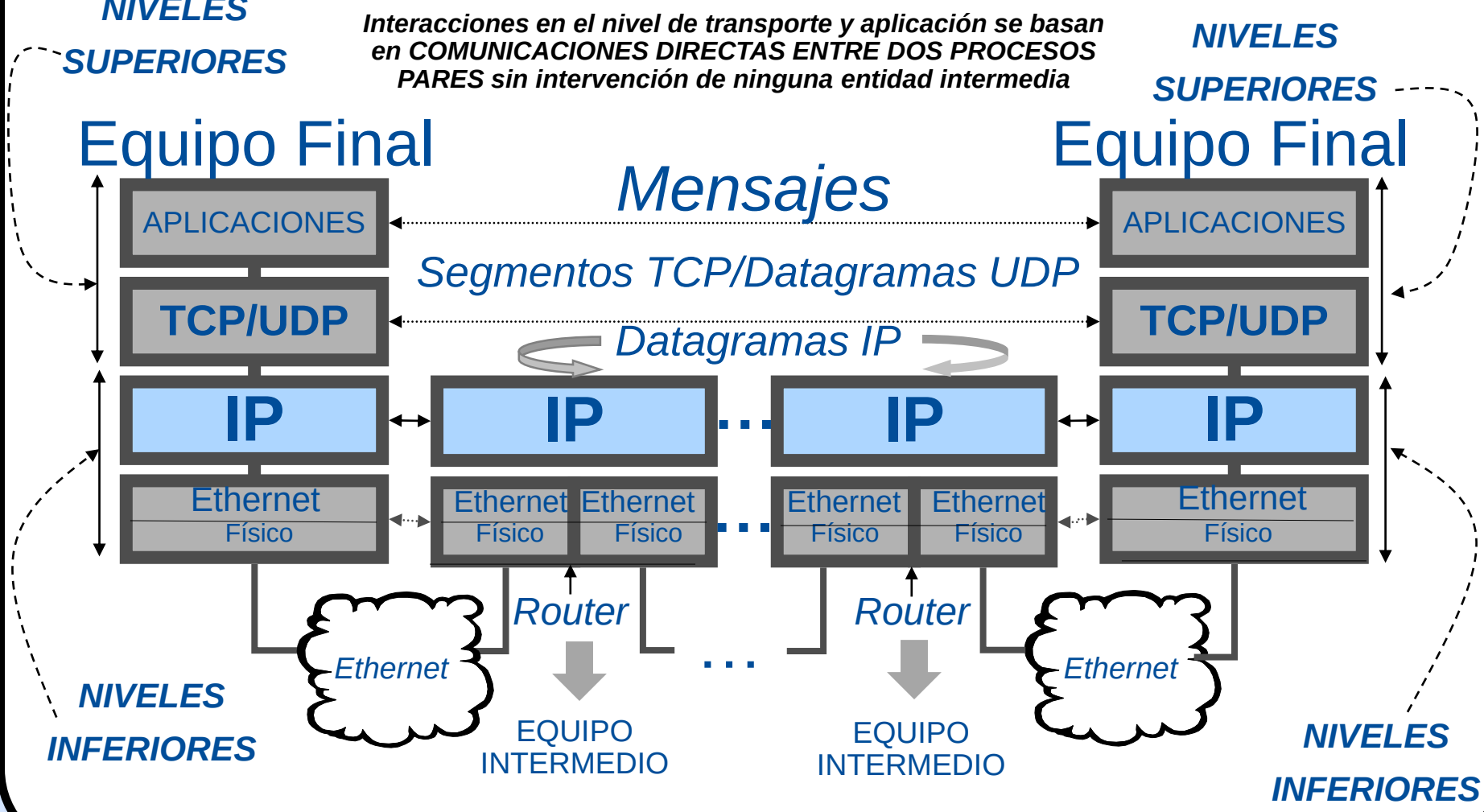
# Niveles de Transporte y Aplicación

## NIVELES SUPERIORES TCP/IP

## NIVELES EXTREMO a EXTREMO

***NO HAY NINGUNA ENTIDAD INTERMEDIA DE TRANSPORTE o APLICACIÓN EN NINGÚN ROUTER por el trayecto en INTERNET***

***Interacciones en el nivel de transporte y aplicación se basan en COMUNICACIONES DIRECTAS ENTRE DOS PROCESOS PARES sin intervención de ninguna entidad intermedia***

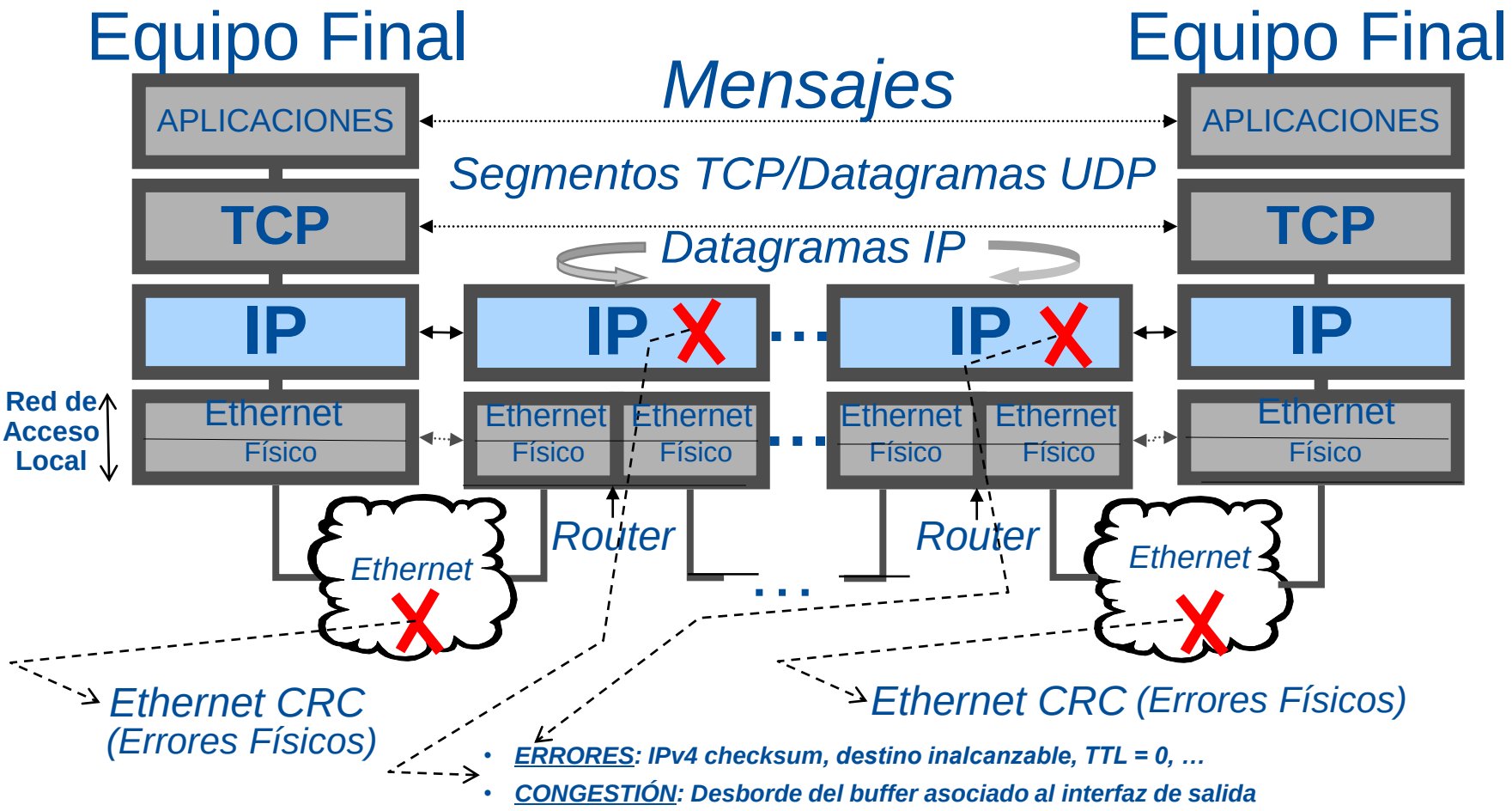


# PROTOCOLO TCP (Transmission Control Protocol) RFC-793, STD 0007

**Las unidades de datos del protocolo TCP se denominan segmentos TCP**

***Transporte fiable de los mensajes de aplicación encapsulados en segmentos TCP***

***Responsable de la recuperación de todos los segmentos perdidos en el nivel de enlace y red***



# FIABILIDAD TCP
# 2 Controles

- **CONTROL DE ERRORES**
  - Lógicos
    - *Bytes (octetos) del campo DATOS de segmentos TCP perdidos, desordenados o duplicados*
  - Físicos
    - ✓ *Producidos localmente, en el nivel de red, en el campo DATOS del paquete IP y* no detectados por el protocolo IP *(sólo los detecta en la cabecera IP)*
- **CONTROL DE FLUJO**
  - *Evita que una entidad o proceso TCP transmita más rápidamente de lo que otra es capaz de almacenar y procesar*

# Control de Errores TCP

## *3 Mecanismos*

### *Números de Secuencia, Confirmaciones y Temporizadores*

- *TODOS LOS OCTETOS DE DATOS contenidos en el CAMPO DE DATOS de cada segmento TCP disponen de*
  - ***UN NÚMERO DE SECUENCIA**: Cada octeto tiene su propio nº de secuencia*
  - ***Una CONFIRMACIÓN***
    - ✓ *Cuando se realiza una confirmación, se está confirmando la numeración de todos los octetos de datos contenidos en un determinado segmento TCP*
    - ✓ *El contenido de cada segmento de información tiene su propia CONFIRMACIÓN*
    - ✓ *Sólo se confirma el contenido de un segmento TCP y no al propio segmento que no va numerado*
  - ***Un TEMPORIZADOR o PLAZO DE ESPERA de la CONFIRMACIÓN***
    - *Cada vez que se envía un segmento TCP se activa el temporizador de dicho segmento*
      - ✓ *Si no llega la confirmación de los octetos de datos de dicho segmento TCP durante un tiempo de espera previsto; se produce un vencimiento del temporizador y se genera una retransmisión de todos los octetos de datos del segmento*

# Diseño Operacional TCP

- *Todo proceso o protocolo de aplicación montado sobre TCP se despreocupa de delimitar sus mensajes*
  - *El proceso de aplicación va pasando sus mensajes de aplicación ilimitados a TCP en forma de "chorro" o flujo de octetos (byte-stream) de una determinada longitud en función del tamaño máximo del buffer de transmisión indicado previamente por su entidad TCP*
  - *A medida que va recibiendo bytes del proceso de aplicación, la entidad TCP los va, ALMACENANDO (buffer de transmisión), NUMERANDO y, posteriormente, AGRUPANDO en segmentos TCP de datos para su envío a IP*
  - Por esta razón, *TCP no numera los segmentos de datos sino los bytes u octetos de datos transmitidos*

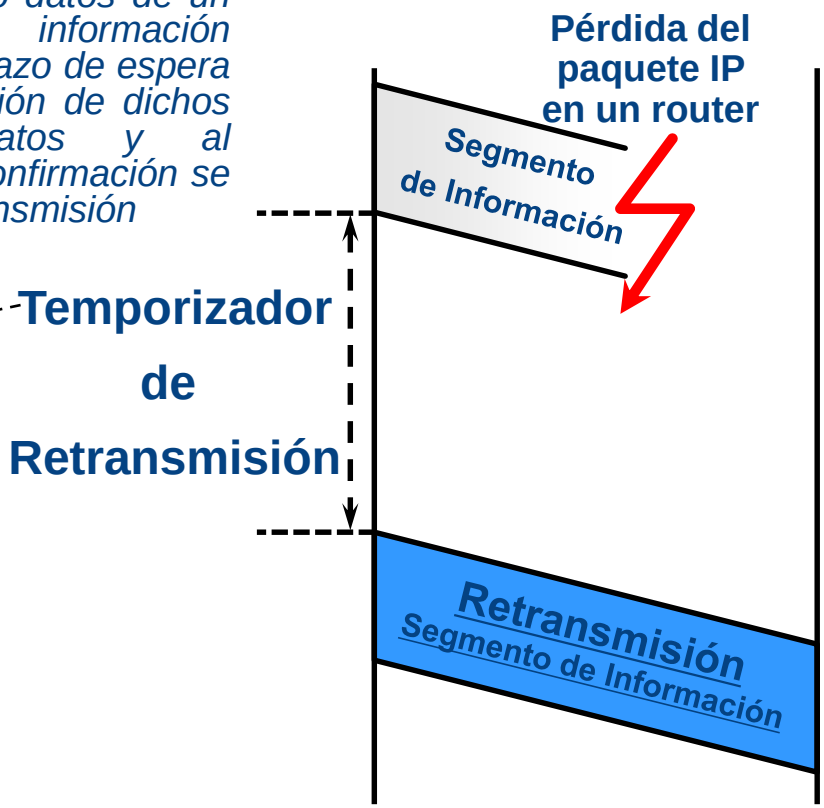
# 3 MECANISMOS DE CONTROL DE ERRORES *(Resumen)*

- ***Números de secuencia***: *Todos los octetos del campo datos de un segmento de información disponen de un número de secuencia*
  - *Los números de secuencia permiten pasar al nivel de aplicación los octetos de datos (cabecera de aplicación + campo datos de usuario o carga útil) ordenadamente y detectar octetos duplicados*
- ***Confirmaciones***: *Todos los octetos del campo datos de un segmento de información tienen asociados una confirmación*
- ***Temporizadores de espera de confirmación***: *Todos los octetos del campo datos de un segmento de información disponen de un* ***PLAZO DE ESPERA*** *para la confirmación de dichos octetos de datos y al vencimiento sin confirmación se produce una retransmisión*

# MECANISMOS DE CONTROL DE ERRORES

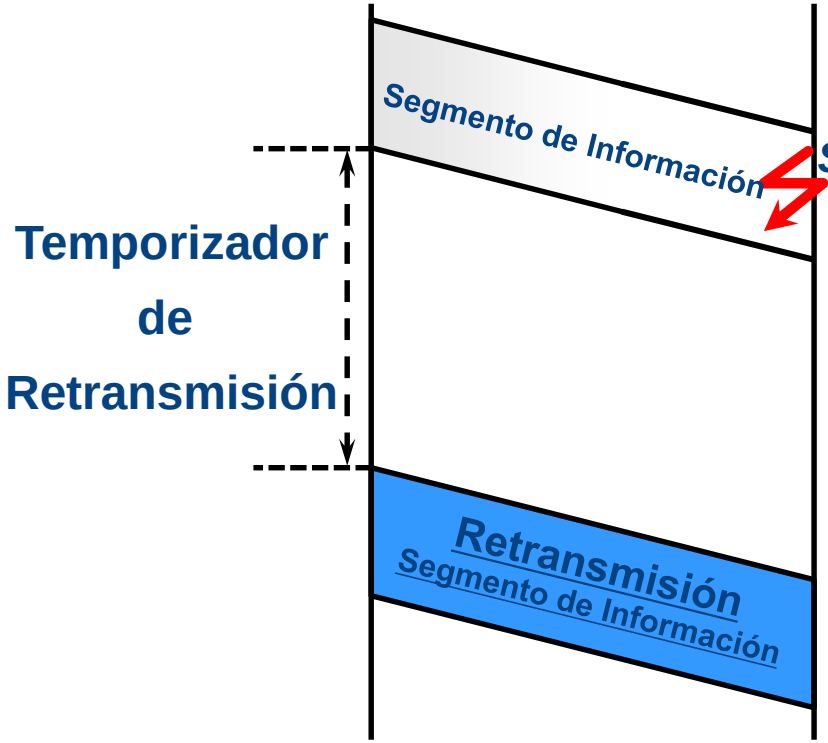
## *Ejemplo de cómo los TEMPORIZADORES DE ESPERA DE CONFIRMACIÓN permiten controlar las pérdidas de segmentos de información en un router*

***Temporizadores***: *Todos los octetos del campo datos de un segmento de información disponen de un plazo de espera para la confirmación de dichos octetos de datos y al vencimiento sin confirmación se produce una retransmisión*



# MECANISMOS DE CONTROL DE ERRORES

***Ejemplo de cómo los TEMPORIZADORES DE ESPERA DE CONFIRMACIÓN, también, permiten controlar las pérdidas de segmentos TCP de datos por errores de transmisión en el destinatario***




**Checksum al segmento completo**

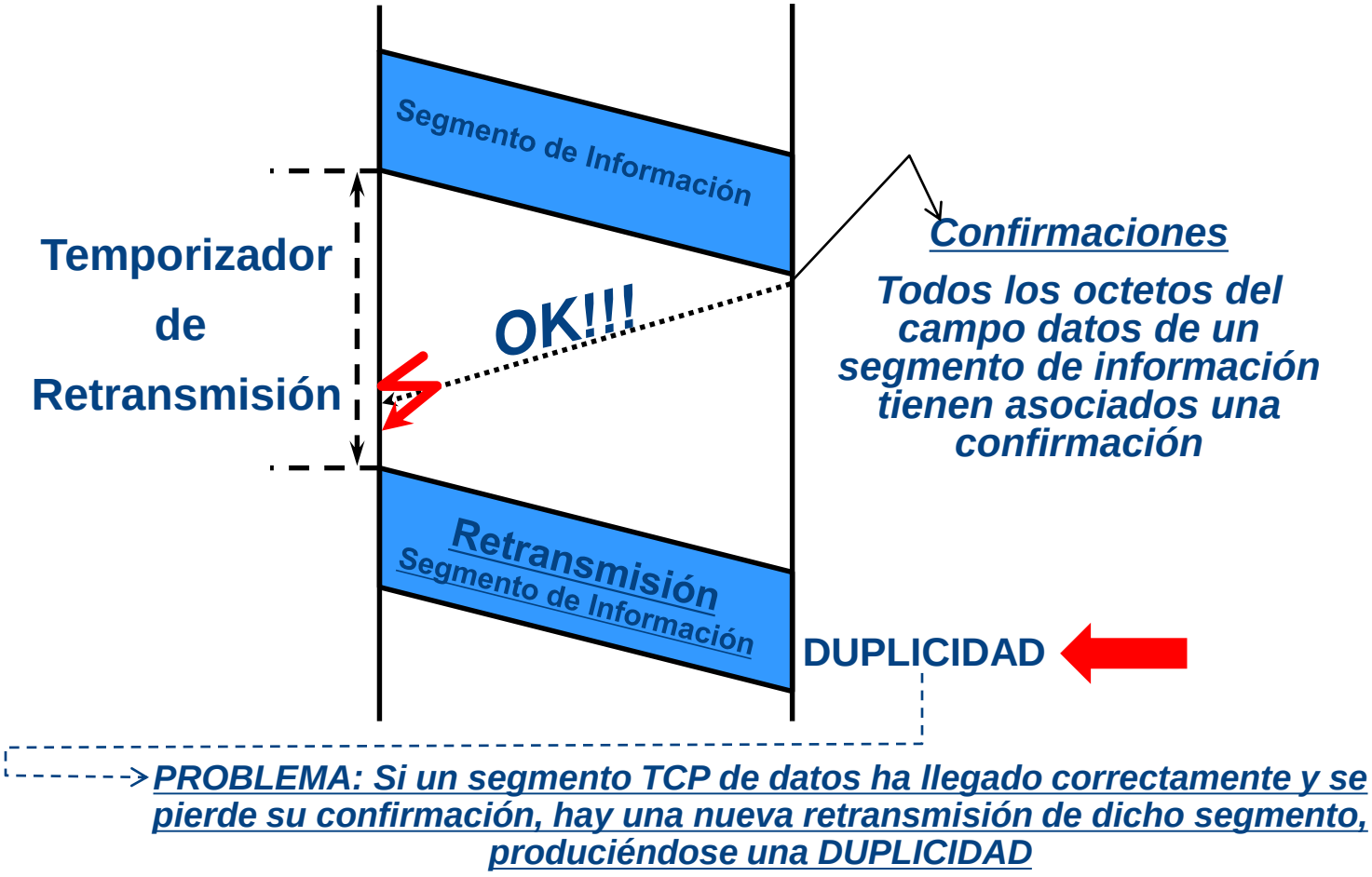
***Segmento con errores físicos***

***OJO!!! Se elimina cualquier segmento de información con errores físicos***

*Producidos localmente, en el nivel de red, en el campo DATOS del paquete IP y* no detectados por el protocolo IP *(sólo los detecta en la cabecera IP)*

# MECANISMOS DE CONTROL DE ERRORES

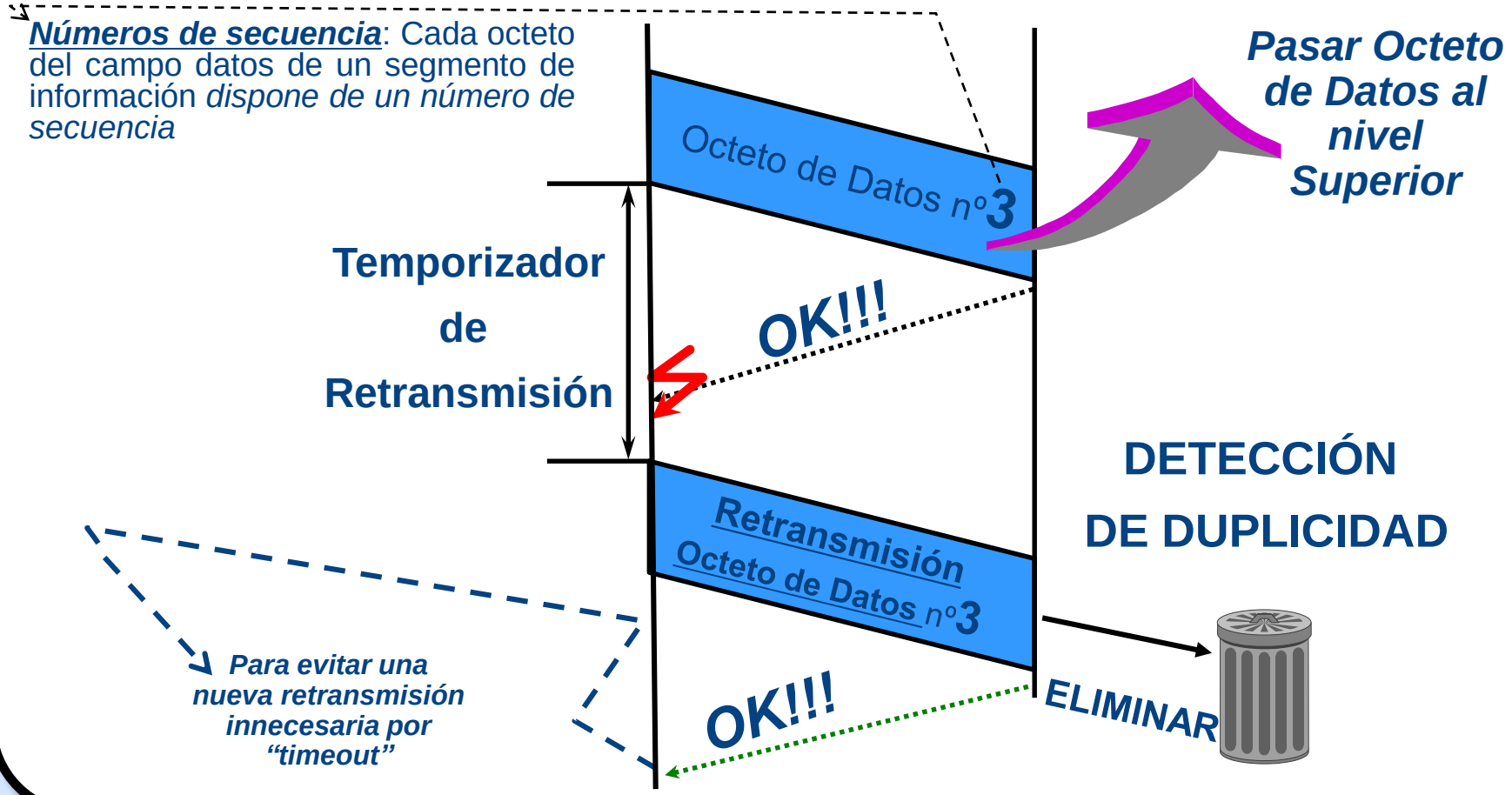
***Asimismo, los TEMPORIZADORES DE ESPERA DE CONFIRMACIÓN, también, permiten controlar las PÉRDIDAS DE CONFIRMACIONES por congestión en un router o por errores de transmisión de dichas confirmaciones, detectados en el destinatario***



# MECANISMOS DE CONTROL DE ERRORES

***LOS NÚMEROS DE SECUENCIA, ADEMÁS, DE CONTROLAR OCTETOS DE DATOS PERDIDOS, DESORDENADOS Y DUPLICADOS; TAMBIÉN CONTROLAN OCTETOS DUPLICADOS cuando llegan las confirmaciones previas con errores físicos o, simplemente, no llegan***

***Números de secuencia***: Cada octeto del campo datos de un segmento de información *dispone de un número de secuencia*



# MECANISMO DE CONTROL DE FLUJO

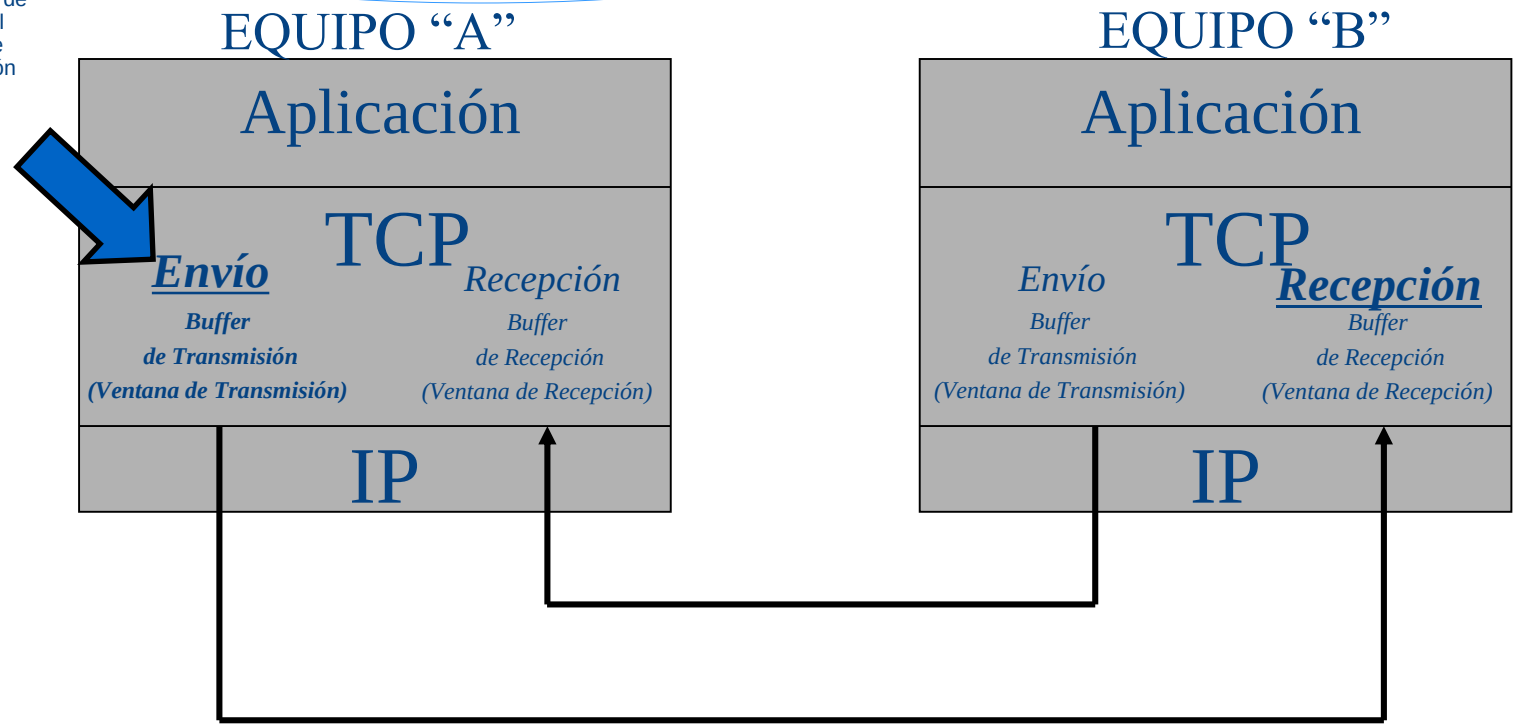
# MECANISMO DE CONTROL DE FLUJO

## CADA PROCESO o ENTIDAD TCP DISPONE DE 2 BUFFERS y 2 VENTANAS DESLIZANTES

Mecanismo de control de numeración de los octetos de datos del buffer de transmisión

$W_T$ (buffer de transmisión) y $W_R$ (buffer de recepción)

Mecanismo de control de numeración de los octetos de datos del buffer de recepción



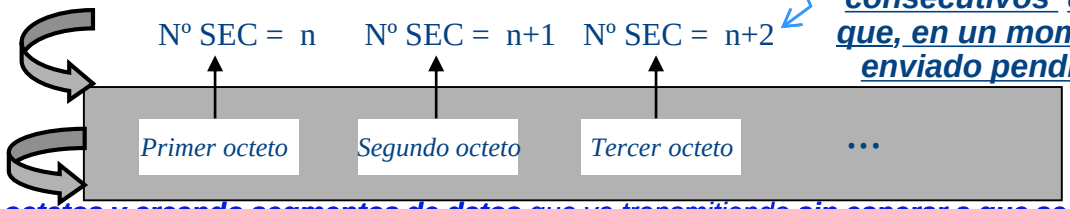

Control de los números del buffer

**VENTANA DESLIZANTE DE TRANSMISIÓN**

**Lista de números de secuencia consecutivos de los octetos de datos que, en un momento dado, el emisor ha enviado pendientes de confirmación**

**BUFFER DE TRANSMISIÓN**

**Los octetos de datos procedentes del proceso de aplicación se almacenan y numeran en el buffer de transmisión**



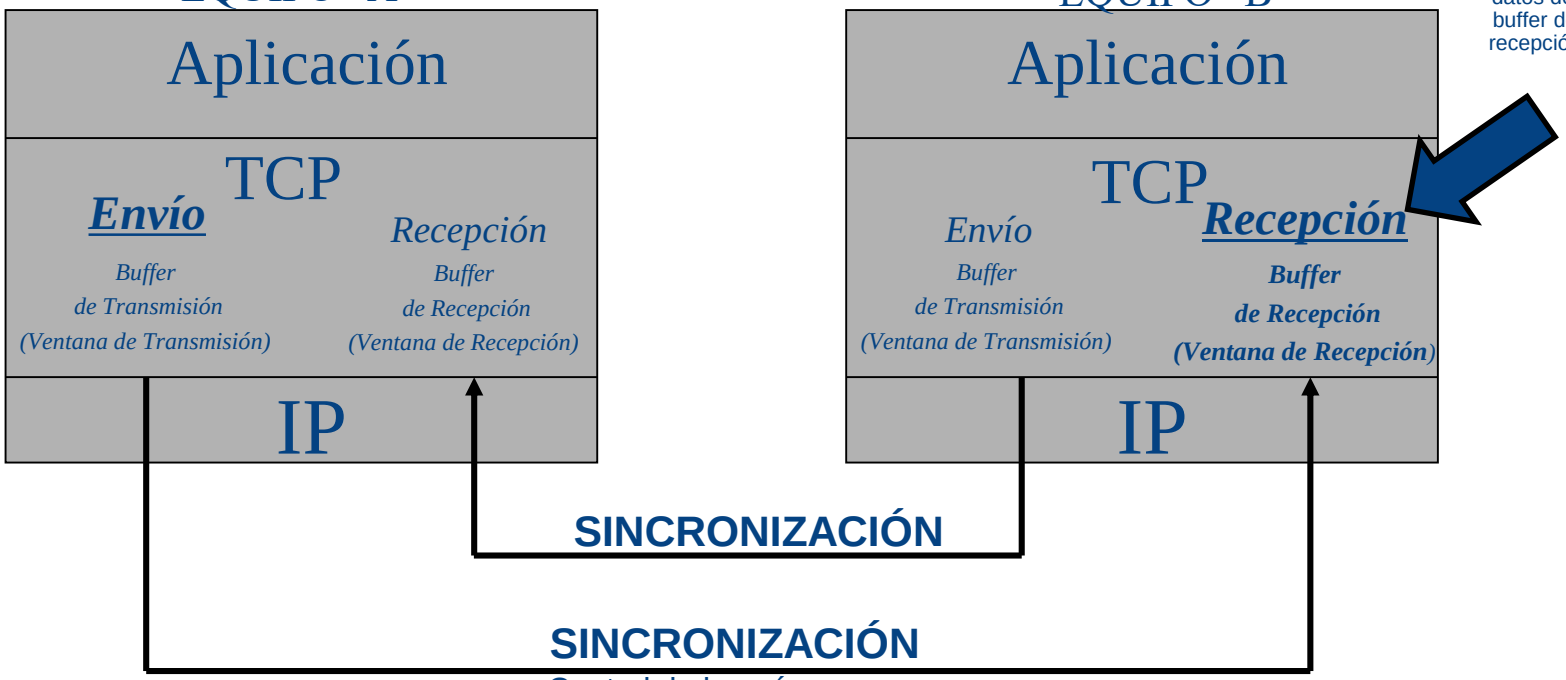

**TCP va AGRUPANDO octetos y creando segmentos de datos** que va transmitiendo **sin esperar a que se llene el buffer de transmisión para acelerar el proceso de transmisión**

# MECANISMO DE CONTROL DE FLUJO

***CADA PROCESO o ENTIDAD TCP DISPONE DE 2 BUFFERS y 2 VENTANAS DESLIZANTES***

***$W_T$ (buffer de transmisión) y $W_R$ (buffer de recepción)***

Mecanismo de control de numeración de los octetos de datos del buffer de recepción



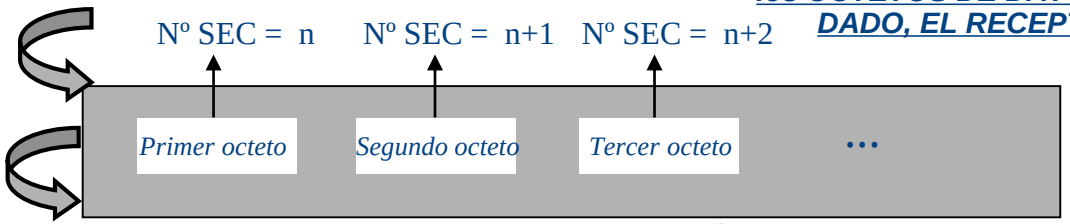

Control de los números del buffer

***VENTANA DESLIZANTE DE RECEPCIÓN***

***Lista de números de secuencia consecutivos de los OCTETOS DE DATOS QUE, EN UN MOMENTO DADO, EL RECEPTOR PUEDE ACEPTAR***

***BUFFER DE RECEPCIÓN***

***Los octetos de datos procedentes de la entidad TCP emisora, y que espera recibir la entidad TCP receptora, SE ALMACENAN POR SU Nº DE SECUENCIA, EN EL BUFFER DE RECEPCIÓN***




***TCP va pasando octetos de datos al proceso de aplicación, si llegan correctamente y consecutivamente a partir del límite inferior, sin esperar a que se llene el buffer de recepción para acelerar el proceso de recepción***

# El Control de Flujo TCP
# Características

- **El Control de Flujo lo ejerce el proceso TCP receptor, a través de su $W_R$, sobre el proceso TCP emisor** para evitar que éste desborde el buffer del receptor
- **$W_T$ en el lado emisor es ESCLAVA de $W_R$ en el lado receptor**
- $W_T$ va variando puntualmente, en fase de transferencia de datos, en función de la $W_R$ del otro extremo

# Sincronización de $W_R$ y $W_T$

- *Las implementaciones TCP pueden ser diferentes en cuanto a los algoritmos auxiliares empleados (p.ej., algoritmos de gestión de ventanas y temporizadores), dependiendo del sistema operativo y su distribución o versión*

- *$W_R$ inicial = Tamaño máximo del buffer de recepción*

  - Posteriormente, en fase de transferencia de datos, $W_R$ va variando, puntualmente, en función de los octetos libres de su buffer de recepción

  - *Límite Inferior* de $W_R$ = *Primer nº de secuencia del primer octeto de datos esperado*

  - *Límite Superior* de $W_R$ = *Último nº de secuencia del último octeto de datos esperado*

  - $W_R$ = *Límite superior – Límite Inferior + 1*

  - *Cuando llegan octetos cuyos números de secuencia de octetos de datos que se esperan recibir, se ALMACENAN previamente en el buffer de recepción y si el primer nº de secuencia recibido coincide con el límite inferior de WR*

    - ✓ ***SE CONFIRMAN los octetos de datos recibidos***
    - ✓ ***SE PASAN los octetos de datos recibidos al proceso de aplicación***
    - ✓ ***SE DESLIZA $W_R$ (límites inferior y superior) en función del tamaño máximo (inicial) del buffer de recepción ($W_R$ inicial)***

# Sincronización de $W_R$ y $W_T$

- *$W_T$ inicial = Tamaño máximo del buffer de transmisión*
- *$W_T$ inicial = $W_R$ inicial* (tamaño máximo del buffer de recepción del otro extremo)
  - *Límite Inferior* de *$W_T$ = Primer nº de secuencia del primer octeto de datos enviado pendiente de confirmación*
  - *Límite Superior* de *$W_T$ = Último nº de secuencia del último octeto de datos enviado pendiente de confirmación*
  - *$W_T$ = Límite Superior – Límite Inferior + 1*
  - *$W_T$ va variando en fase de transferencia de datos en función de la $W_R$ del otro extremo*
  - *CONFIRMACIÓN (ACK) de la entidad TCP receptora: Primer nº de secuencia del primer octeto del campo de datos del siguiente segmento de información que se espera recibir, con lo cual los números de secuencia anteriores están todos confirmados*
    - ✓ SIEMPRE INDICA EL LÍMITE INFERIOR DE *$W_T$*
      - *El ACTUAL si ha habido SEGMENTOS PERDIDOS*
        - *La entidad TCP emisora NO REALIZA NINGUNA ACCIÓN*
      - *Uno NUEVO si NO ha habido segmentos perdidos*
        - *La entidad TCP emisora:*
          - ***DESACTIVA el temporizador asociado a los octetos confirmados***
          - ***ELIMINA copia de los octetos confirmados en el buffer de transmisión***
          - ***DESLIZA $W_T$ (límites inferior y superior) en función del tamaño máximo (inicial) del buffer de recepción ($W_R$ inicial)***

# Ejemplo de Sincronización de $W_R$ y $W_T$ para un Correcto Control de Flujo

## ESCENARIO INICIAL

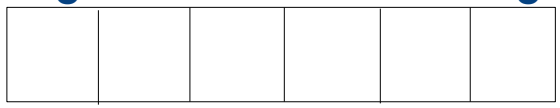
**Se desea transmitir la palabra ALARMA y no se van a producir ERRORES**

**ENTIDAD TCP RECEPTORA**

*Conoce por configuración previa el tamaño de su buffer de recepción y cómo llenarlo (MSS o DATOS)*

$W_R = 6$ *(MSS o DATOS = 2)*

LI=¿? BUFFER DE RECEPCIÓN LS¿?



$W_R$ *inicial = 6 = Tamaño máximo*

**ENTIDAD TCP EMISORA**

¿BUFFER DE TRANSMISIÓN?

LI=¿? LS=¿?

$W_T$ = ¿?


1 *Intercambio inicial de información de control*

Nº SEC=0 ---------- *Primer nº de secuencia elegido por el emisor para el primer octeto del campo de datos del primer segmento de información que SE VA A ENVIAR*

CONF=1 ---------- *Nº SEC elegido por el emisor +1*

*Si Nº SEC = n entonces CONF = n+1*

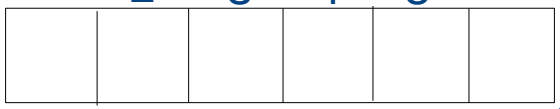
$W_R = 6$

MSS o DATOS = 2

**ENTIDAD TCP EMISORA**

LI=1 2 3 4 5 LS=6

$W_T = 6$



*LA ENTIDAD TCP EMISORA SABE QUE PUEDE ENVIAR COMO MÁXIMO 6 bytes ($W_R$ = 6) en 3 SEGMENTOS DE DATOS (MSS recibido = 2) y el primer byte = 1 (LI=1)*

*Límite Inferior de $W_T$ = Primer nº de secuencia del primer octeto de datos enviado pendiente de confirmación*

*Límite Superior de $W_T$ = Último nº de secuencia del último octeto de datos enviado pendiente de confirmación*

2

**ENTIDAD TCP RECEPTORA**

$W_R = 6$ *(MSS o DATOS =2)*

LI=1 2 3 4 5 LS=6

*Límite Inferior de $W_R$ = Primer nº de secuencia del primer octeto de datos esperado*

*Límite Superior de $W_R$ = Último nº de secuencia del último octeto de datos esperado*

# Ejemplo de Sincronización de $W_R$ y $W_T$ para un Correcto Control de Flujo

2 *$W_T$ en el lado emisor SE HA AJUSTADO a la $W_R$ en el lado receptor*

**ENTIDAD TCP RECEPTORA**

***$W_R$ = 6 (MSS o DATOS = 2)***

**ENTIDAD TCP EMISORA**

***$W_T$ = 6***


***LA ENTIDAD TCP EMISORA SABE QUE PUEDE ENVIAR COMO MÁXIMO 6 bytes ($W_R$ = 6) en 3 SEGMENTOS DE DATOS (MSS recibido = 2) y el primer byte = 1 (LI=1)***

**PROCESO DE APLICACIÓN**

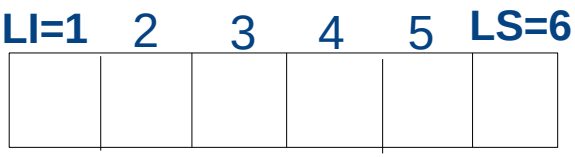
*(Una vez ha terminado la fase de establecimiento de la conexión comienza la fase de transferencia fiable de datos)*

*Envíame un byte-stream no superior a 6 octetos*

**ENTIDAD TCP EMISORA**

3

***$W_T$ = 6***



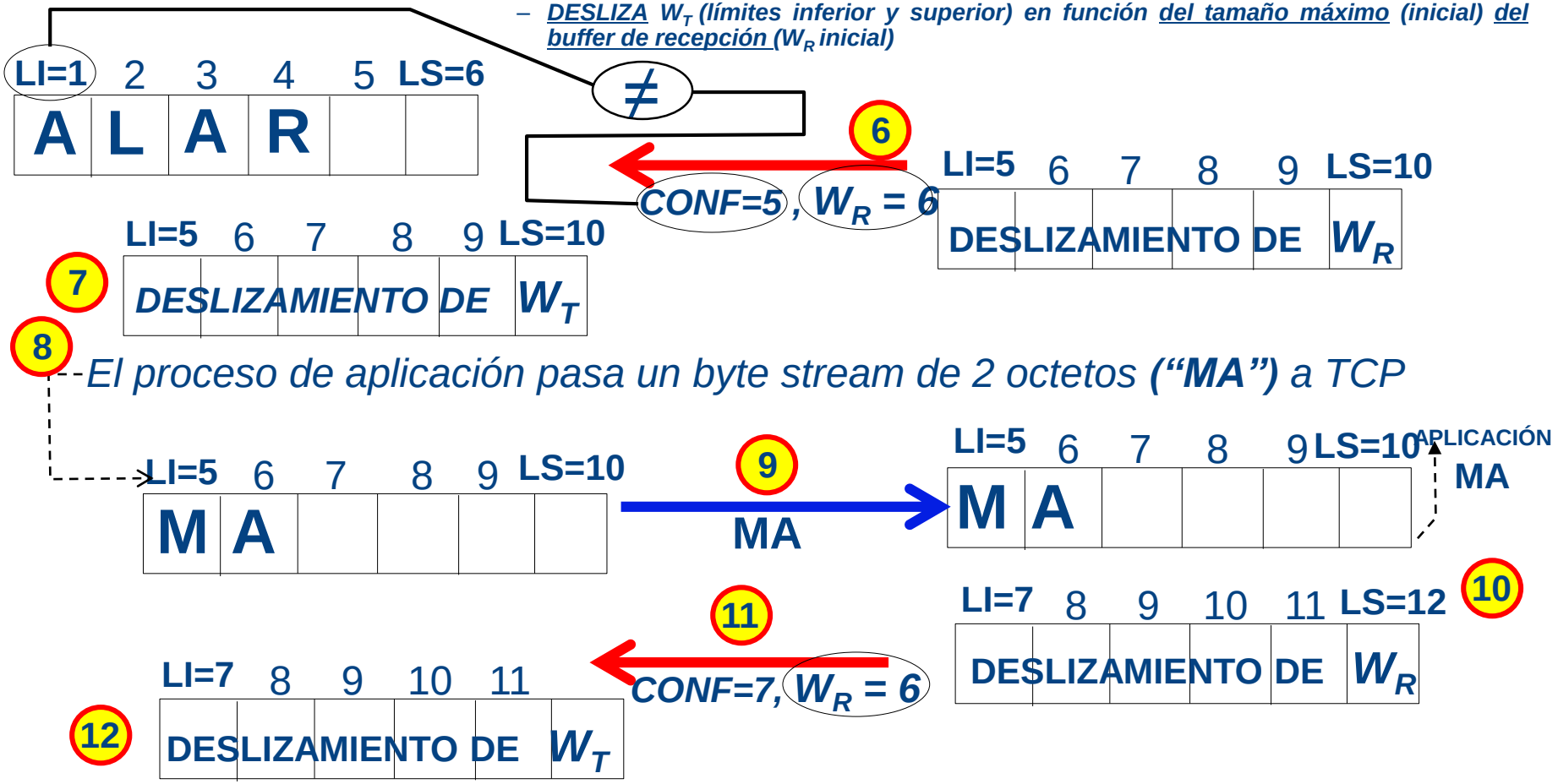
# TRANSMISIÓN DE SEGMENTOS TCP SIN ERRORES

*El proceso de Aplicación pasa un byte-stream de 4 octetos*


- ***Cuando llegan octetos cuyos números de secuencia se esperan recibir, SE ALMACENAN previamente en el buffer de recepción y SI EL PRIMER NÚMERO DE SECUENCIA RECIBIDO COINCIDE CON EL LÍMITE INFERIOR DE WR***
  - ***SE CONFIRMAN los octetos de datos recibidos***
  - ***SE PASAN los octetos de datos recibidos al proceso de aplicación*** *(TCP va pasando octetos de datos, si son los esperados, y llegan correctamente y consecutivamente, a partir del límite inferior sin esperar a que se llene el buffer de recepción)*
  - ***SE DESLIZA $W_R$ (límites inferior y superior) en función del tamaño máximo (inicial) del buffer de recepción ($W_R$ inicial)***

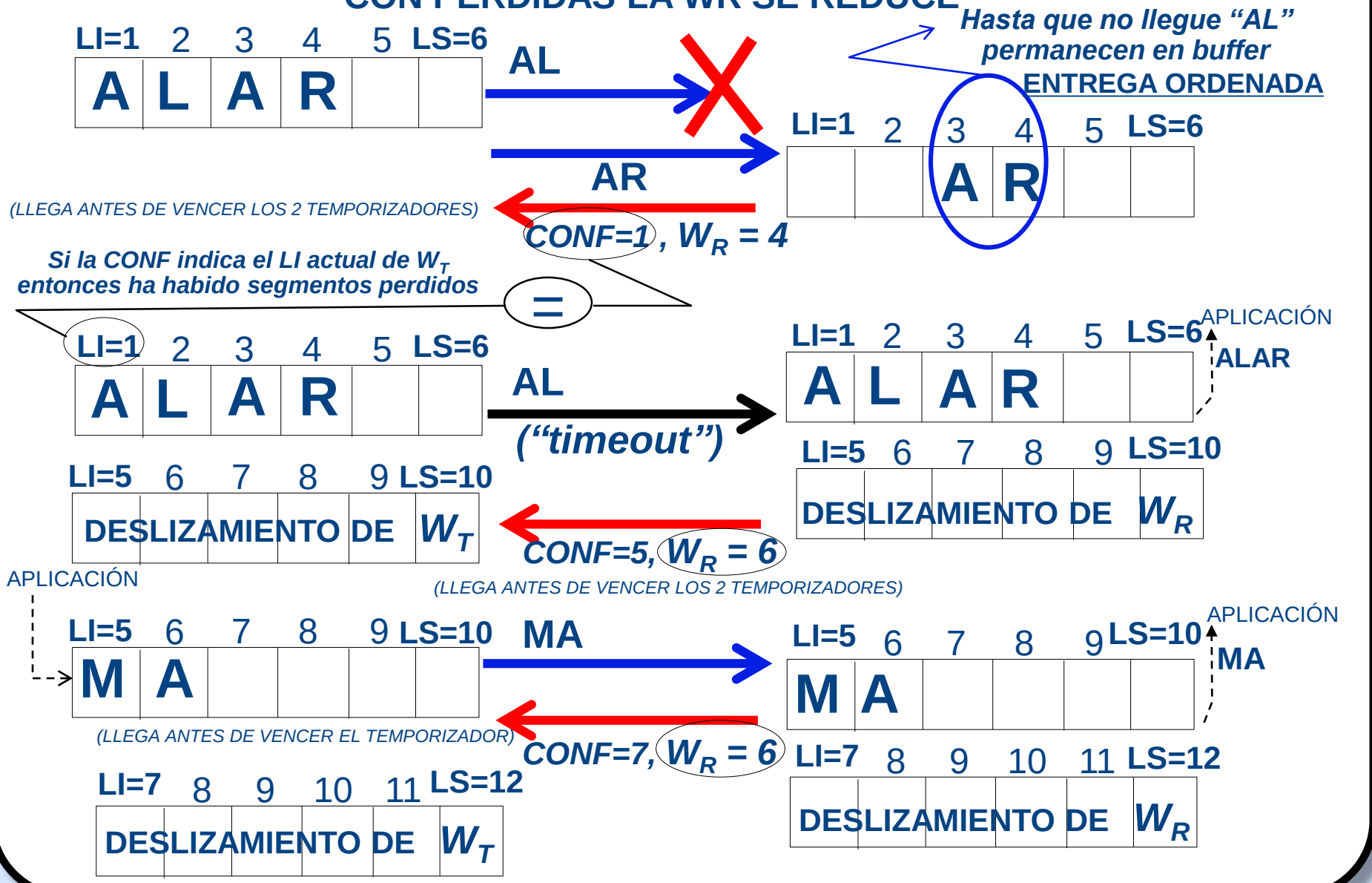

- ***CONFIRMACIÓN (CONF o ACK) de la entidad TCP receptora: Primer nº de secuencia del primer octeto del campo de datos del siguiente segmento de información que se espera recibir, con lo cual los números de secuencia anteriores están todos confirmados***
  - **SIEMPRE INDICA EL LÍMITE INFERIOR QUE TIENE QUE TENER $W_T$**

- **LA CONFIRMACIÓN SIEMPRE INDICA EL LÍMITE INFERIOR DE $W_T$**
  - ***Si la CONFIRMACIÓN indica un LÍMITE INFERIOR DE $W_T$ diferente al que tiene actualmente $W_T$ es que no ha habido segmentos perdidos***
    - *La entidad TCP emisora:*
      - ***DESACTIVA el temporizador asociado a los octetos confirmados***
      - ***ELIMINA copia de los octetos confirmados en el buffer de transmisión***
      - ***DESLIZA $W_T$ (límites inferior y superior) en función del tamaño máximo (inicial) del buffer de recepción ($W_R$ inicial)***



# TRANSMISIÓN DE SEGMENTOS TCP CON ERRORES

## CON PÉRDIDAS LA WR SE REDUCE

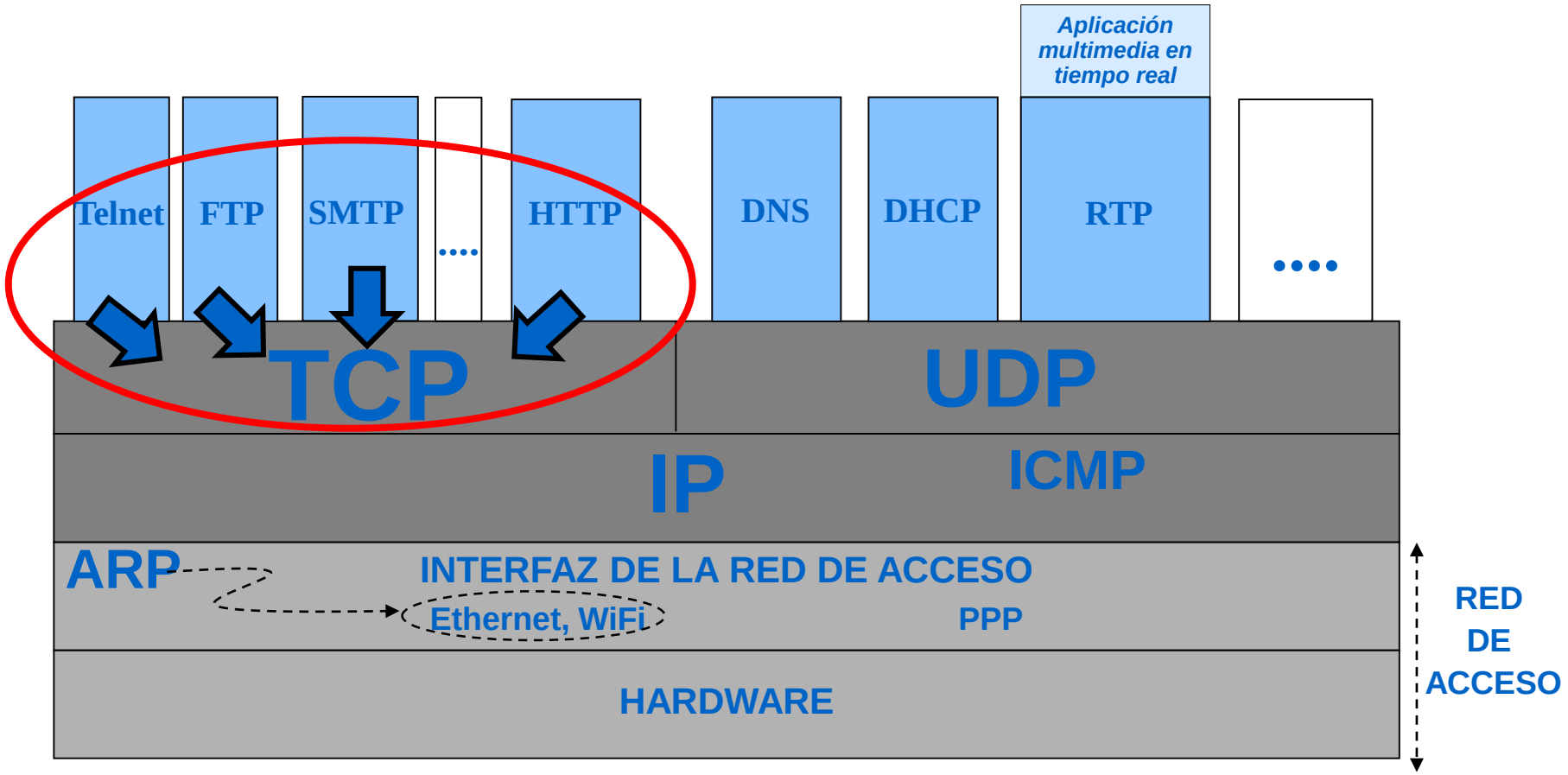


# Protocolo TCP: 4 Servicios

1. **FLUJO DE OCTETOS (BYTE-STREAM):**
   - ***Entidad TCP emisora ALMACENA y NUMERA en el buffer de transmisión los octetos de datos (byte-stream) pasados por el proceso de aplicación***
   - ***Posteriormente, AGRUPA dichos octetos en SEGMENTOS TCP***
2. **FIABLE**
   - ✓ **CONTROL DE ERRORES:**
     - ✓ **Lógicos de octetos de datos perdidos, desordenados y duplicados (detección: *número de secuencia* y corrección: *temporizadores y retransmisión*)**
     - ✓ **Físicos de bits cambiados (detección: *suma de comprobación* y corrección: *temporizadores y retransmisión*)**
   - ✓ **CONTROL DE FLUJO: *Mecanismo de ventana deslizante ejercido por el proceso TCP receptor sobre el proceso TCP emisor para evitar que éste desborde el buffer del receptor***
3. **MULTIPLEXADO o simultáneo y diferenciado a través de los números de puerto**
   - ***Mecanismos y recursos de fiabilidad por separado para cada proceso de aplicación***
4. **DÚPLEX: TRANSFERENCIAS SIMÚLTÁNEAS EN LOS DOS SENTIDOS**

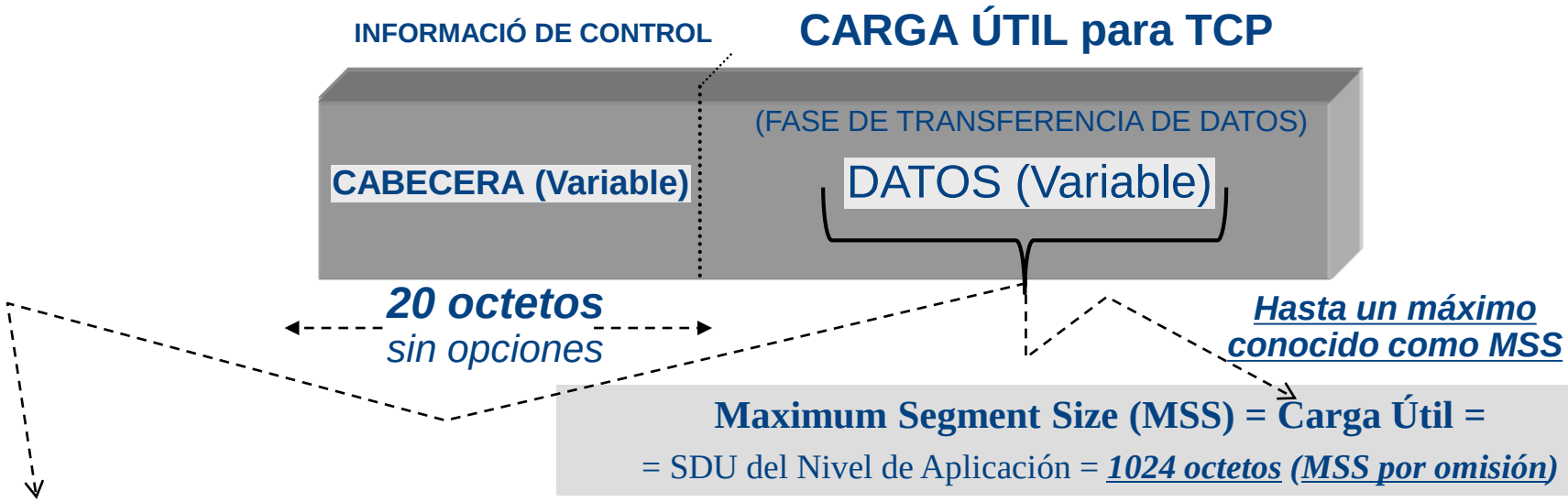
# Servicio Multiplexado TCP

***SIMULTÁNEO y DIFERENCIADO a través de los números de puerto, aplicando mecanismos y recursos de fiabilidad por separado***



# PROTOCOLO TCP (RFC-793)
# Formato de un Segmento TCP

*Engloba dos tipos de información:* ***Cabecera*** + ***Datos***




**Maximum Segment Size (MSS) = Carga Útil =**

= SDU del Nivel de Aplicación = ***1024 octetos (MSS por omisión)***

- ***CREA UN un campo DATOS (MSS) en función:***
  - ***Del MSS solicitado por el otro extremo (por omisión, 1024 octetos)***
  - ***Del tamaño de la cabecera TCP (por omisión, 20 octetos)***
  - ***Del tamaño de la cabecera IP (por omisión, 20 octetos)***
  - ***De la MTU de salida (1500 octetos)***

# Cabecera TCP + DATOS

*Primer nº de secuencia del primer octeto del campo de datos del segmento de información que se va a enviar = Nº de secuencia= 32 bits= Módulo $2^{32}$= 0, 1, 2, …$2^{32}$-1, 0, 1, …, $2^{32}$-1, 0,1…*

*Primer nº de secuencia del primer octeto del campo de datos del siguiente segmento de información que se espera recibir (LOS OCTETOS ANTERIORES ESTÁN CONFIRMADOS)*


*CABECERA TCP Mínima por Omisión =20 octetos*

*Máxima=60 octetos*

*DESP = Nº de bloques de 4 octetos que ocupa la cabecera*

*Por omisión 5 bloques (0101), como máximo 15 bloques (1111)*

*SYN=1 ACK=0, Solicitud conexión*

*SYN=1 ACK=1, Respuesta OK!*

*Ventana deslizante de recepción:*

*Nº de octetos de datos que el emisor de esta información puede manejar en función del tamaño puntual de su buffer de recepción =*

*= 0 octetos (descanso), 1 octeto, 2 octetos hasta $2^{16}$-1 octetos (65.535 octetos )*

*Cantidad de octetos libres que se pueden almacenar en el buffer de recepción*

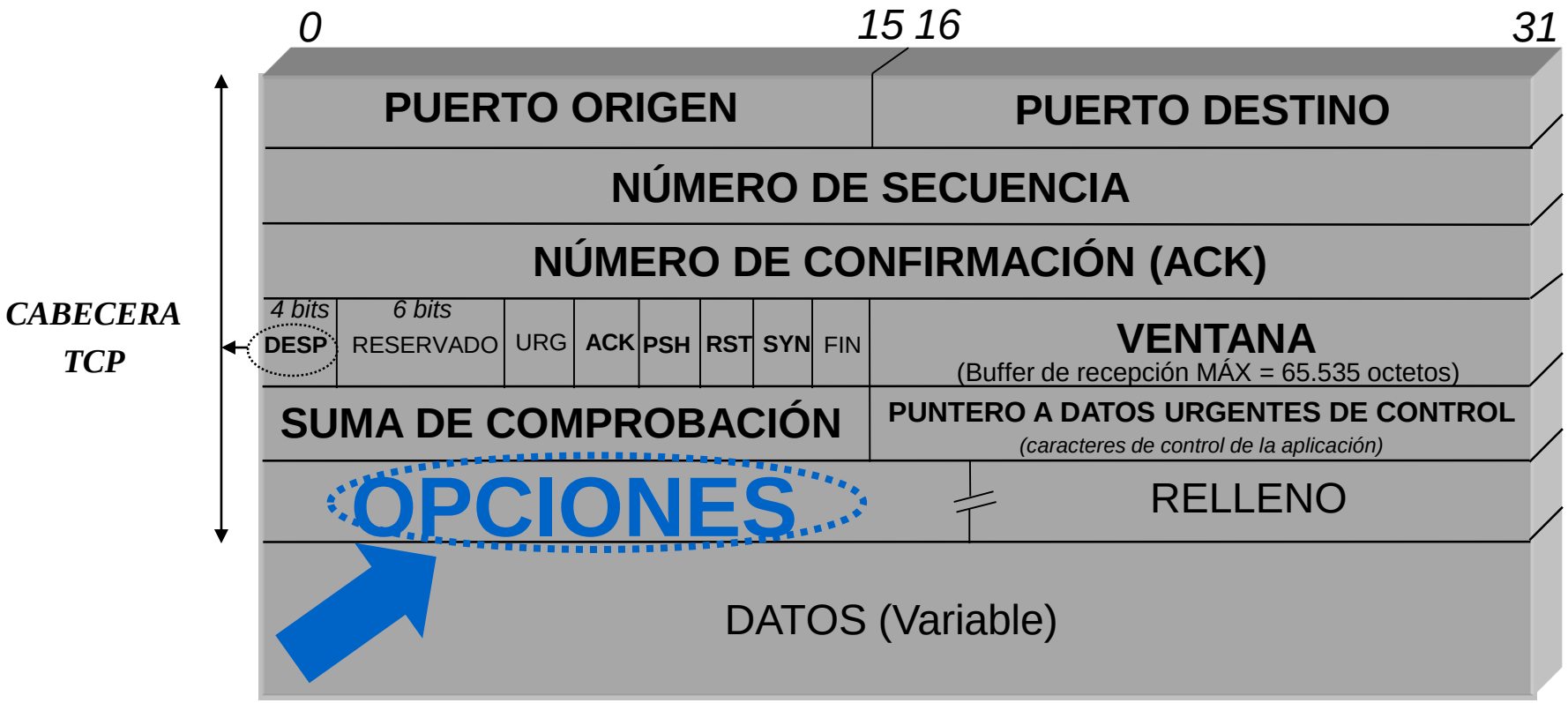
# Bit Push

- *El bit PSH se utiliza en TCP para forzar el envío de segmentos de datos cortos e independientes*
  - INTERACCIONES CLIENTE SERVIDOR manejando bloques reducidos de datos Y LO USAN TANTO EL CLIENTE COMO EL SERVIDOR
- *Lo activa la entidad TCP emisora cuando se lo indica en una llamada (byte-stream + push = 1) su proceso de aplicación*
- ***Se activa cuando el proceso de aplicación desea evitar que su entidad TCP EMISORA esté esperando más octetos de datos de la aplicación para construir un segmento mayor en función del MSS esperado en el otro extremo***
- A su vez, la ENTIDAD TCP RECEPTORA debe pasar cuanto antes, y SIEMPRE DE FORMA ORDENADA Y CONSECUTIVA A PARTIR DEL LÍMITE INFERIOR, los octetos de datos de un segmento con el bit PUSH = 1

# Temporizadores de Espera de Confirmación

- **No es lo mismo que el destino esté ubicado en la misma Ethernet o WiFi que disperso geográficamente por Internet**
- ***Para una mayor eficiencia de TCP, los valores de los temporizadores de espera de confirmación se establecen dinámicamente mediante ALGORITMOS AUTOADAPTATIVOS* que *ajustan sus valores a la dispersión geográfica y al estado de la red*, según es percibido éste por la entidad de transporte emisora**
- **La mayoría de las implementaciones se basan en el Algoritmo autoadaptativo de Karn para el cálculo del *RTT (Round Trip Time)***
  - ***Se toma una muestra RTT de cada segmento de información transmitido y su correspondiente confirmación***
  - ***Se obtiene un promedio de dichas muestras***
  - ***Se le añade un margen de seguridad***

# Opciones TCP

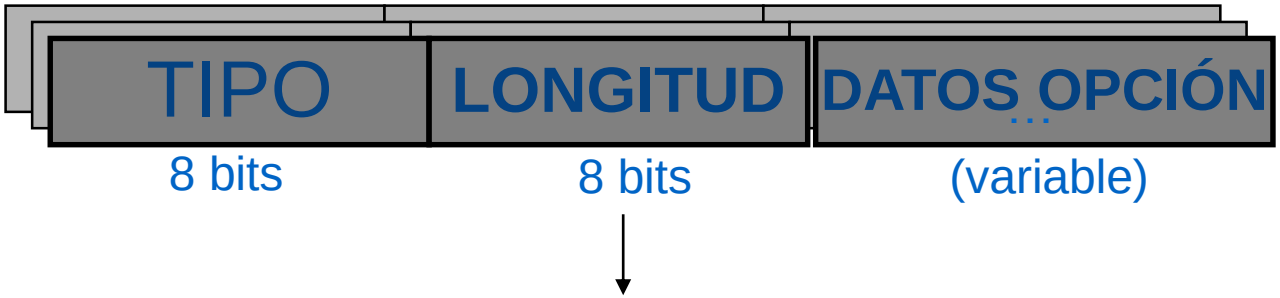


# Formato de las Opciones TCP

## Campo de Información de Control de Longitud Variable para Servicios Adicionales

*RFC-793 y RFC-1323*

## Formato TLV: Tipo-Longitud-Valor o Datos




***Longitud completa***

***de la opción en octetos***

***(TIPO + LONGITUD + DATOS)***

# Opciones TCP más Relevantes Descripción

| TIPO | LONGITUD (octetos) | DATOS | Descripción |
|---|---|---|---|
| **2** | 4 | Valor MSS | ***MSS*** |
| **3** | 3 | Tamaño de la Ventana | ***Escala de la Ventana*** |
| **4** | 2 | **DATOS = 0** | *SACK PERMITTED* *(Fase de establecimiento)* |
| **5** | Variable | $X_1 - Y_1$ ... $X_n - Y_n$ | *SACK* *(Fase de transferencia de datos)* |
| **8** | 10 | Valor actual reloj emisor (4 octetos) + Respuesta Eco (4 octetos) | ***Marca de Tiempo*** |
| ... | ... | ... | ... |

# CAMPO DE OPCIONES DE INFORMACIÓN DE CONTROL

## (LONGITUD VARIABLE DE HASTA 40 OCTETOS)

- ***Según la implementación TCP que disponga el sistema operativo, habrá más o menos opciones***
- ***SE ESPECIFICAN EN LA FASE DE ESTABLECIMIENTO DE LA CONEXIÓN (sólo en aquellos segmentos de control con el bit SYN = 1)***
- **Tamaño Máximo de Segmento (MSS)**: ***Nº máximo de octetos de la carga útil (campo de datos) que el emisor de esta información desea recibir para un procesado más óptimo de dicha carga***
  - ***Por omisión: 1024 octetos*** *(otros: 1460, 1360, … en función del nº máx de opciones TCP/IP )*
  - *No confundir MSS con el tamaño del buffer de recepción*
- **Factor de Escala de la Ventana (RFC-1323)**:
  - ***Permite ampliar el campo VENTANA de 16 bits ($2^{16}$ = 65.536 octetos) hasta un máximo de 30 bits ($2^{30}$ = 1 Gbyte)***
    - **Por omisión, el tamaño máximo de la ventana de recepción es de *$2^{16}$-1 octetos (65.535 octetos)***
- **Marca o sello de tiempo (timestamp)**:

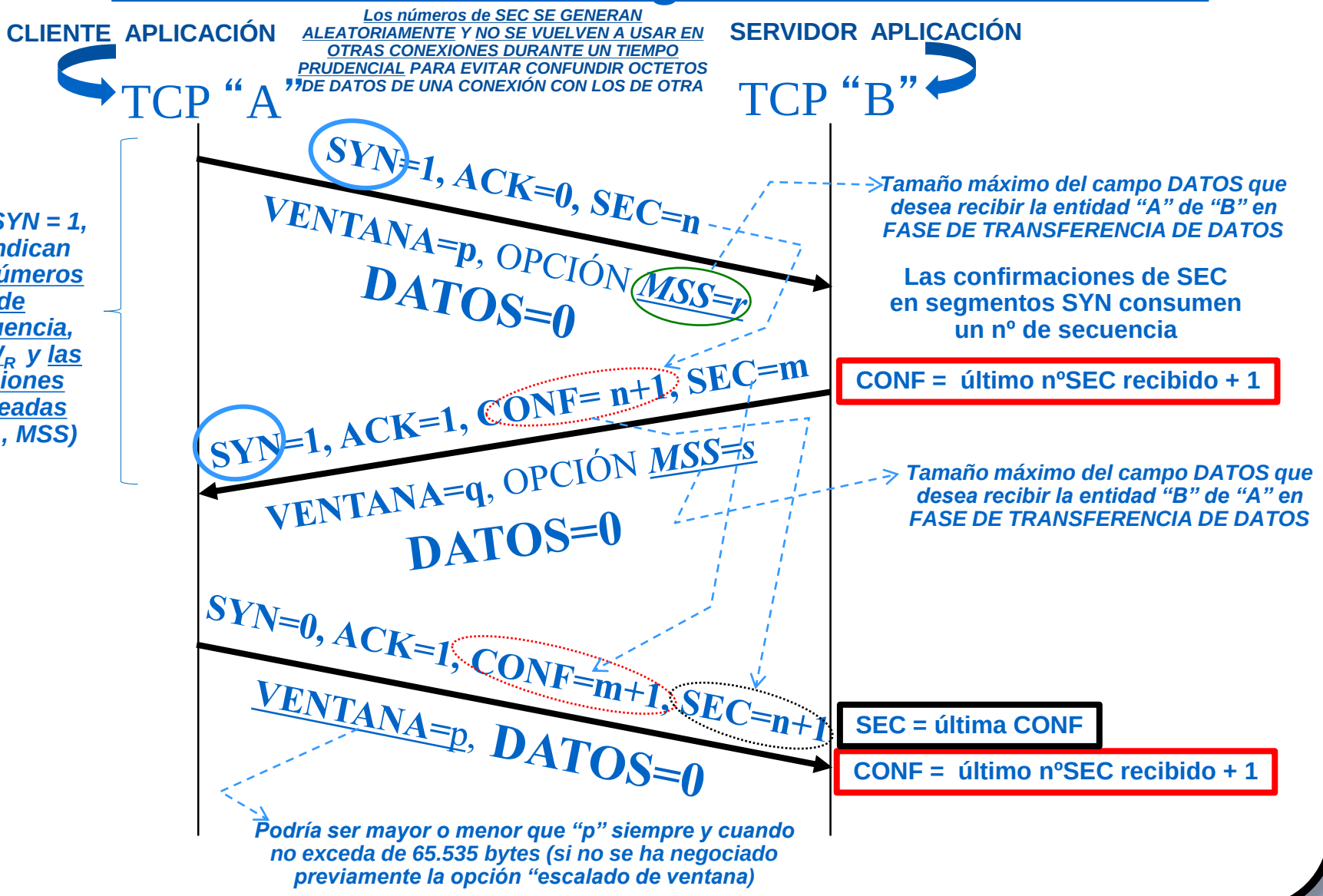
| Tipo =8 | Longitud = 10 | Valor actual del reloj emisor | Eco del valor actual del reloj emisor |
|---|---|---|---|

  - ***Permite al emisor configurar, CON MÁS EXACTITUD, sus TEMPORIZADORES DE ESPERA DE CONFIRMACIÓN***
  - **El emisor pone el valor de su reloj**
  - **El receptor pone el valor de su reloj y devuelve el mismo valor (eco) en el ACK del segmento**
    - **Valor actual del reloj del emisor = *valor monótamente creciente (formato UT = nº de mseg después de medianoche)***
  - ***El eco es válido si el bit ACK = 1 en la cabecera del segmento***

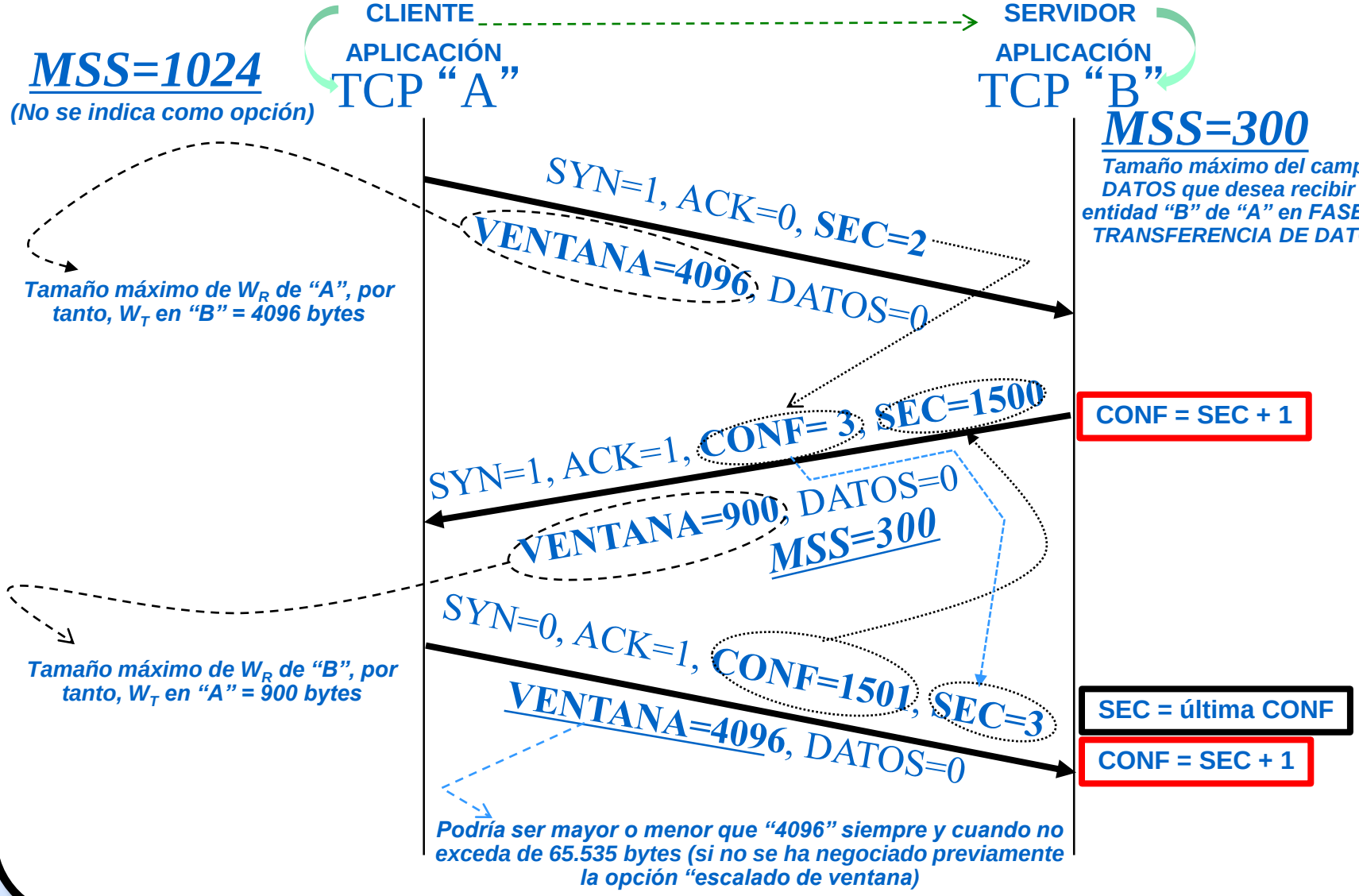
# TCP ofrece un SERVICIO FIABLE a los Procesos del Nivel de Aplicación

- **Un SERVICIO FIABLE significa un SERVICIO ORIENTADO A CONEXIÓN, el cual dispone de TRES FASES**
  1. **ESTABLECIMIENTO DE LA CONEXIÓN**
  2. **TRASFERENCIA DE DATOS**
  3. **LIBERACIÓN DE LA CONEXIÓN**

# Fase de Establecimiento de una Conexión TCP
## Intercambio de 3 segmentos SIN DATOS



# EJEMPLO de Fase de Establecimiento de una Conexión TCP
## Intercambio de 3 segmentos con la Opción MSS en un extremo ("B")

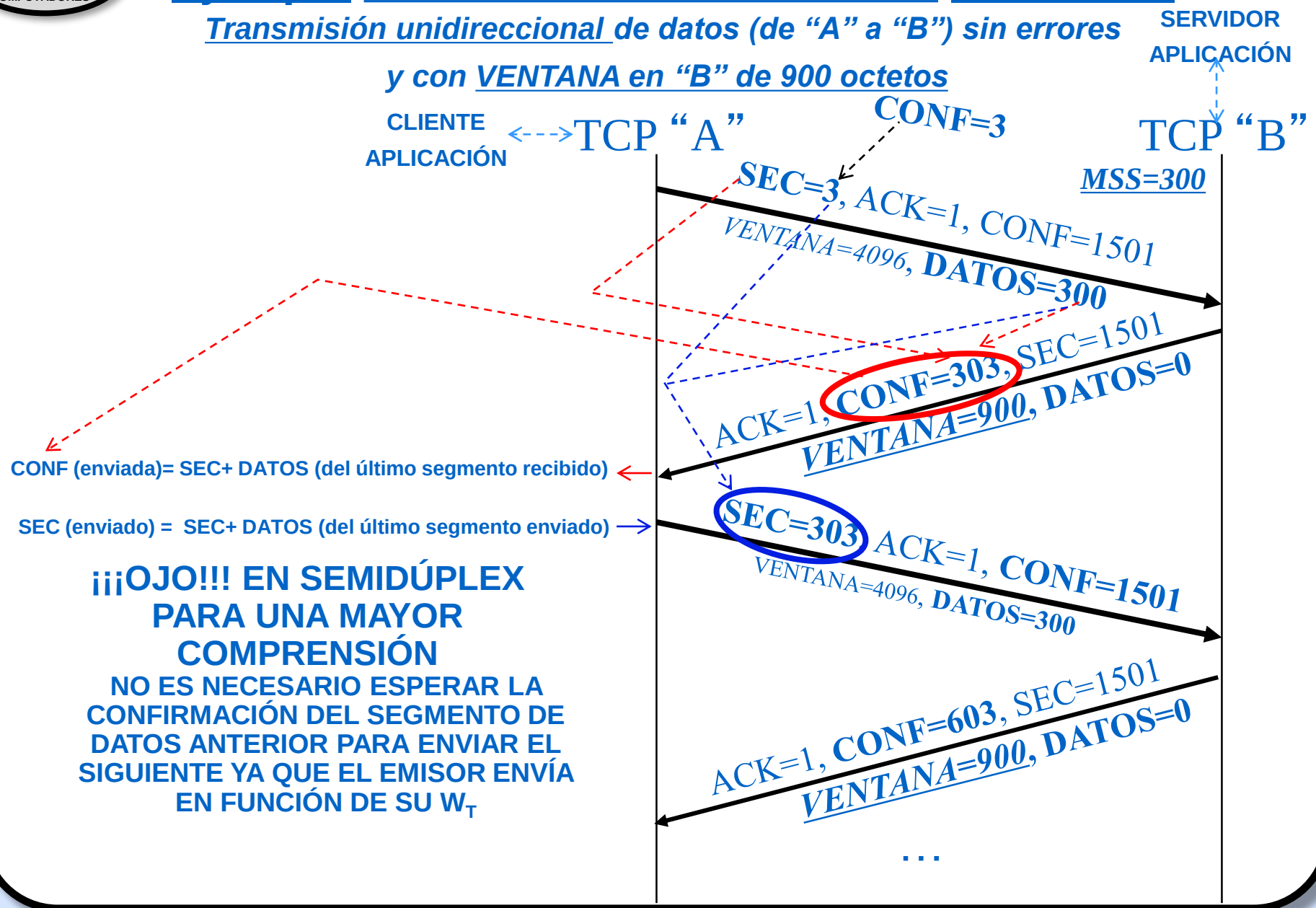


# Características de WR y MSS

- WR (buffer de recepción) local se va llenando en función del MSS local indicado previamente al otro extremo
- La WR MÁXIMA INICIAL, al principio, se indica mediante el bit SYN = 1
- La WR MÁXIMA INICIAL de hasta 65.535 bytes (y a partir de 65.536 hasta 1 Gbyte con escalado de ventana) en función de los recursos RAM disponibles, puede variar en cualquier momento disminuyendo o aumentando (sin sobrepasar 65.535 bytes si no se ha negociado previamente la opción de escalado de ventana)
  - Dependen de los recursos RAM de cada entidad TCP y puede ser diferente para cada sentido
- El valor por omisión de MSS es de 1024 bytes
  - El MSS elegido, para recibir en fase de transferencia de datos, puede ser mayor o menor que el valor por omisión y se indica mediante una opción de la cabecera TCP con el bit SYN = 1
    - Cada entidad TCP indica a su "entidad par" qué MSS está dispuesto a aceptar, y la otra debe obedecer (puede usar ese valor u otro menor, pero no mayor)
- Si el otro extremo no está de acuerdo con el WR o el MSS recibido envía un RST = *1*

# Ejemplo de Transferencia de Datos sin Errores

***Transmisión unidireccional de datos (de "A" a "B") sin errores y con VENTANA en "B" de 900 octetos***



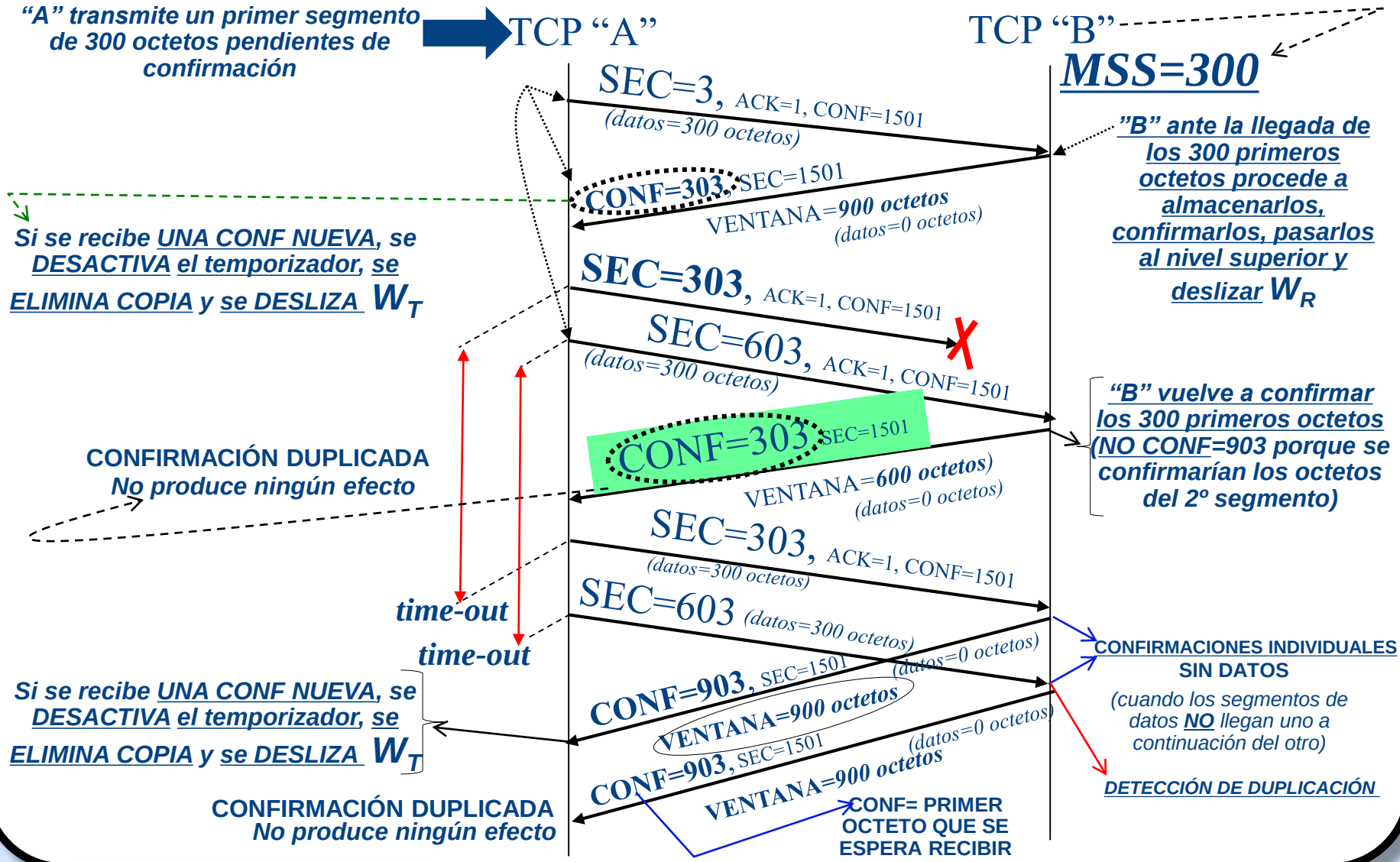

**¡¡¡OJO!!! EN SEMIDÚPLEX PARA UNA MAYOR COMPRENSIÓN**

**NO ES NECESARIO ESPERAR LA CONFIRMACIÓN DEL SEGMENTO DE DATOS ANTERIOR PARA ENVIAR EL SIGUIENTE YA QUE EL EMISOR ENVÍA EN FUNCIÓN DE SU $W_T$**

# Ejemplo de Transferencia de Datos con Errores

***Transmisión unidireccional de datos (de "A" a "B") con errores***

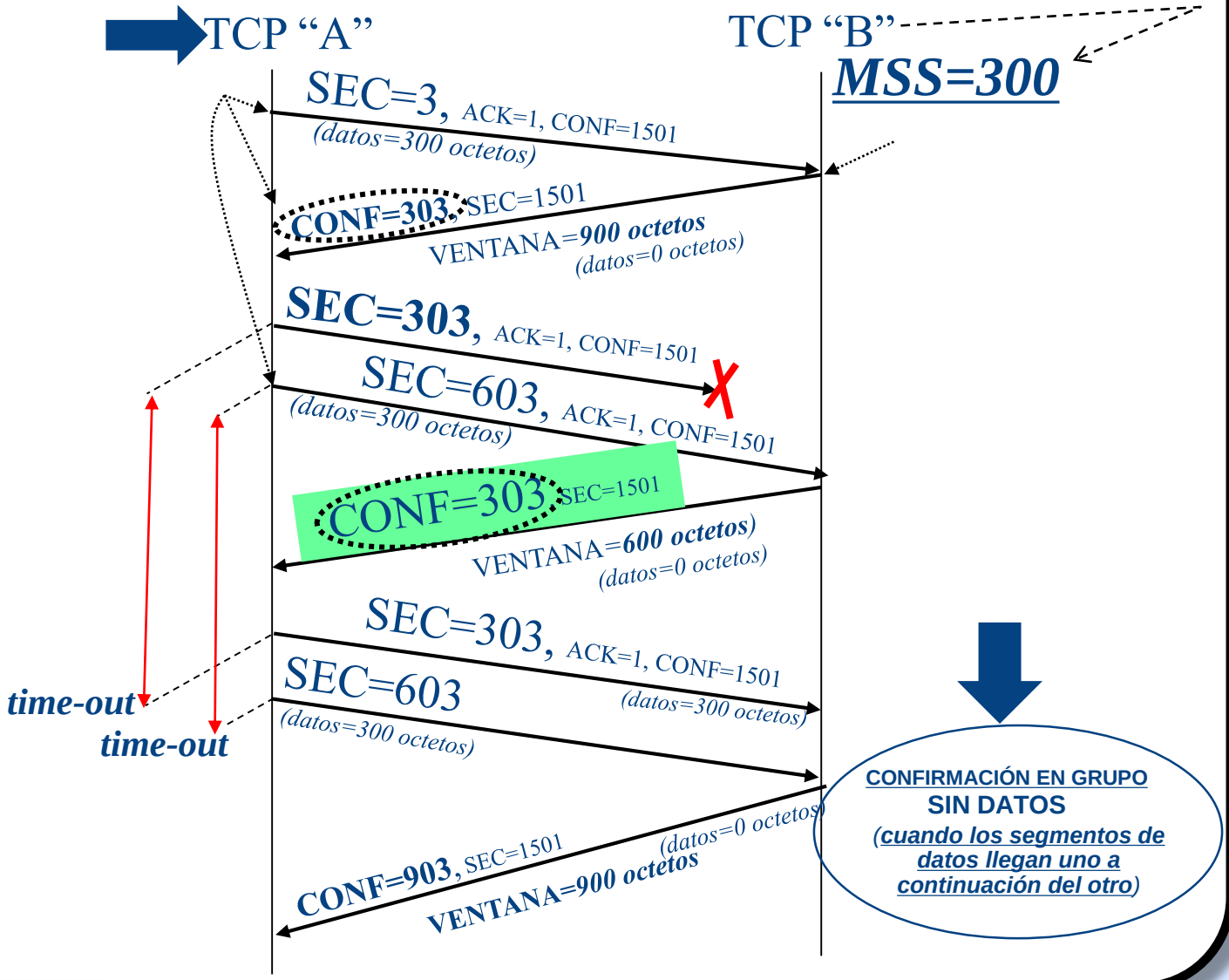
***(pérdidas de segmentos TCP) y con VENTANA en "B" de 900 octetos***



# Ejemplo de Transferencia de Datos con Errores

***Transmisión unidireccional* de datos (de "A" a "B") con errores**

**(pérdidas de segmentos TCP) y con *VENTANA* en "B" de 900 octetos**



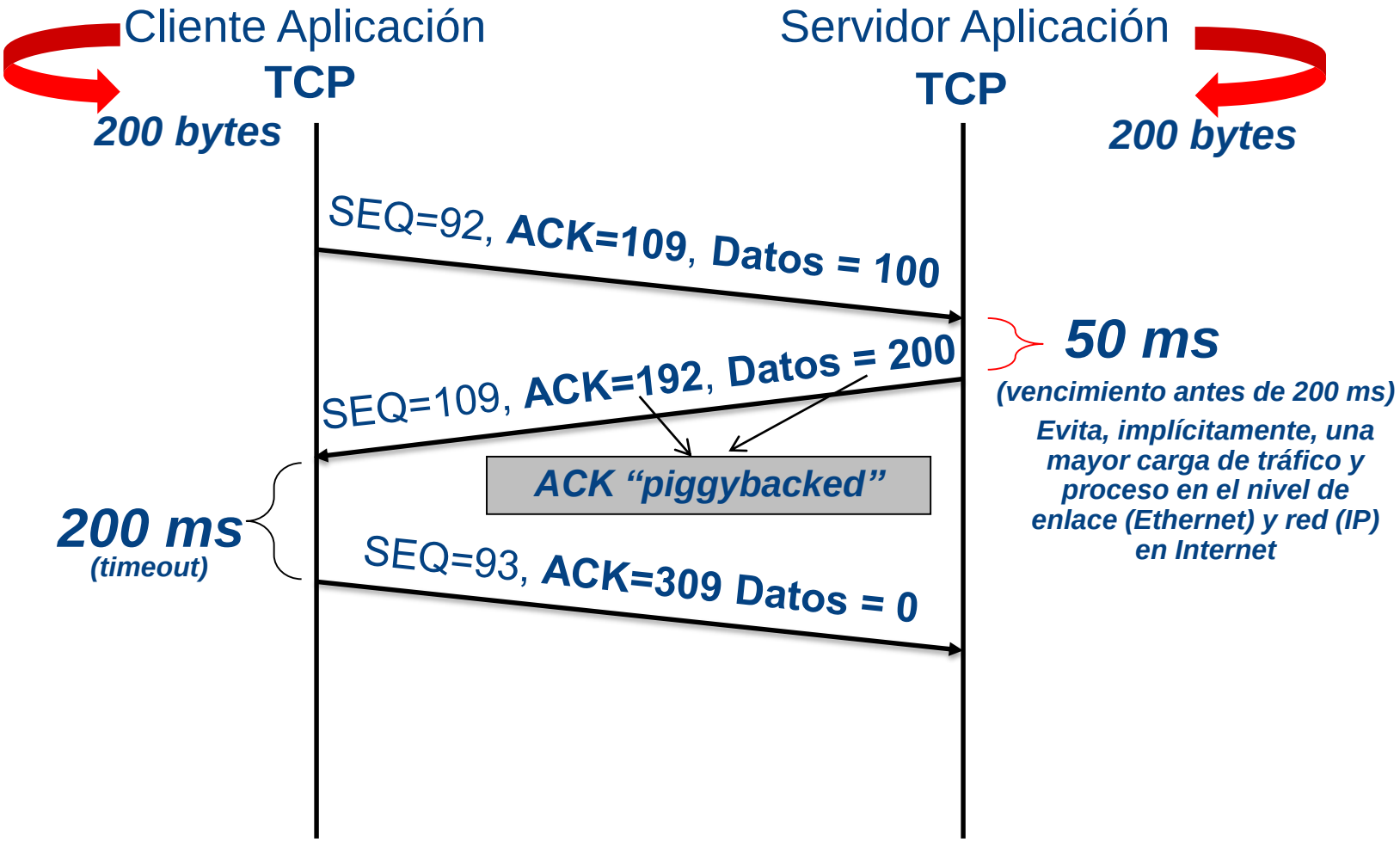
# Temporizador de Espera de Datos

## *(ACK "piggybacked")*

- **Confirmaciones individuales o en grupo (en función del retardo entre segmentos de datos recibidos) CON DATOS**
- **Cuando una entidad TCP recibe uno o varios segmentos de datos y, a su vez, no tiene datos que enviar de vuelta, no envía el ACK inmediatamente**
  - *Espera "un poco" por si el proceso de aplicación genera datos y de ese modo aprovecha para enviar el ACK (individual o en grupo) en el segmento de datos (ACK "piggybacked")*
    - *En muchos sistemas ese temporizador (espera de datos de la aplicación) suele ser de unos 200 ms*
  - Si se agota el timer sin que la aplicación haya producido datos se envía un segmento sin datos con el ACK

# Ejemplo de Temporizador de Espera de Datos

## *Temporizador de Espera de Datos = 200 ms*



# Opciones TCP más Relevantes
## Descripción

| TIPO | LONGITUD (octetos) | DATOS | Descripción |
|---|---|---|---|
| **2** | 4 | Valor MSS | *MSS* |
| **3** | 3 | Tamaño de la Ventana | *Escala de la Ventana* |
| **4** | 2 | **DATOS = 0** | ***SACK PERMITTED*** ***(Fase de establecimiento)*** |
| **5** | Variable | $X_1 - Y_1$ ... $X_n - Y_n$ | ***SACK*** ***(Fase de transferencia de datos)*** |
| **8** | 10 | Valor actual reloj emisor (4 octetos) + Respuesta Eco (4 octetos) | *Marca de Tiempo* |
| ... | ... | ... | ... |

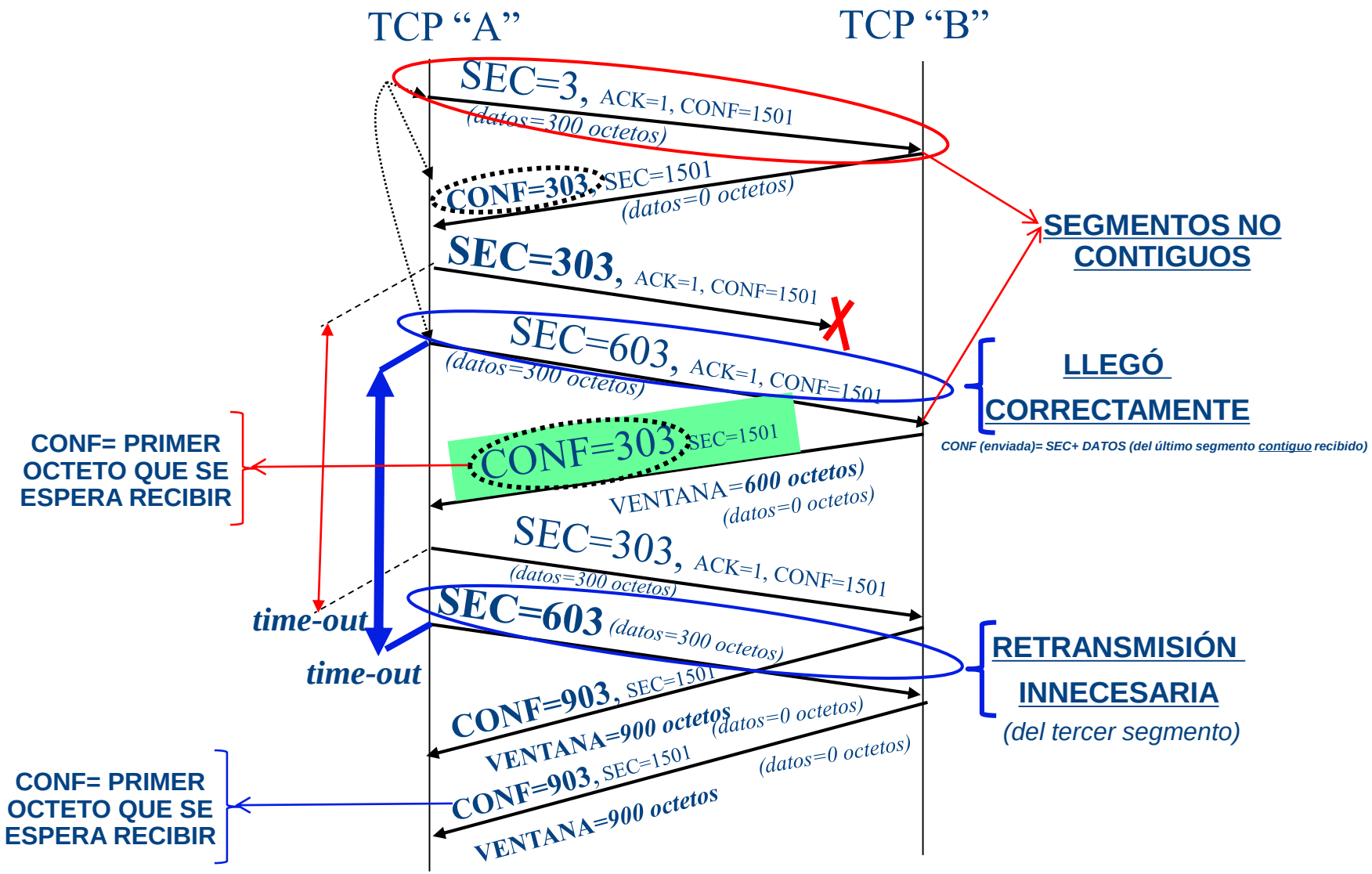
# TCP SACK
# Confirmación Selectiva
# *Selective Acknowledgment (SACK)*

- Definido en el *RFC-2018* y, posteriormente, extendido en el *RFC-2883*
  - Es una opción, por omisión, en toda implementación TCP para, EN CASO DE PERDIDAS DE SEGMENTOS TCP, aumentar la eficiencia del protocolo
    - *Permite informar a la entidad TCP emisora de aquellos octetos de datos de segmentos de información NO CONTIGUOS que han sido recibidos correctamente*
      - *Un segmento no contiguo es aquél cuyo primer octeto de datos no es el primero que se espera recibir*
      - *EVITA VENCIMIENTOS DE TEMPORIZADORES y RETRANSMISIONES INNECESARIAS*
        - *Evita, implícitamente, una mayor carga de tráfico y proceso en el nivel de enlace (Ethernet) y red (IP) en Internet*

# TCP SACK
# Confirmación Selectiva
# *Selective Acknowledgment (SACK)*

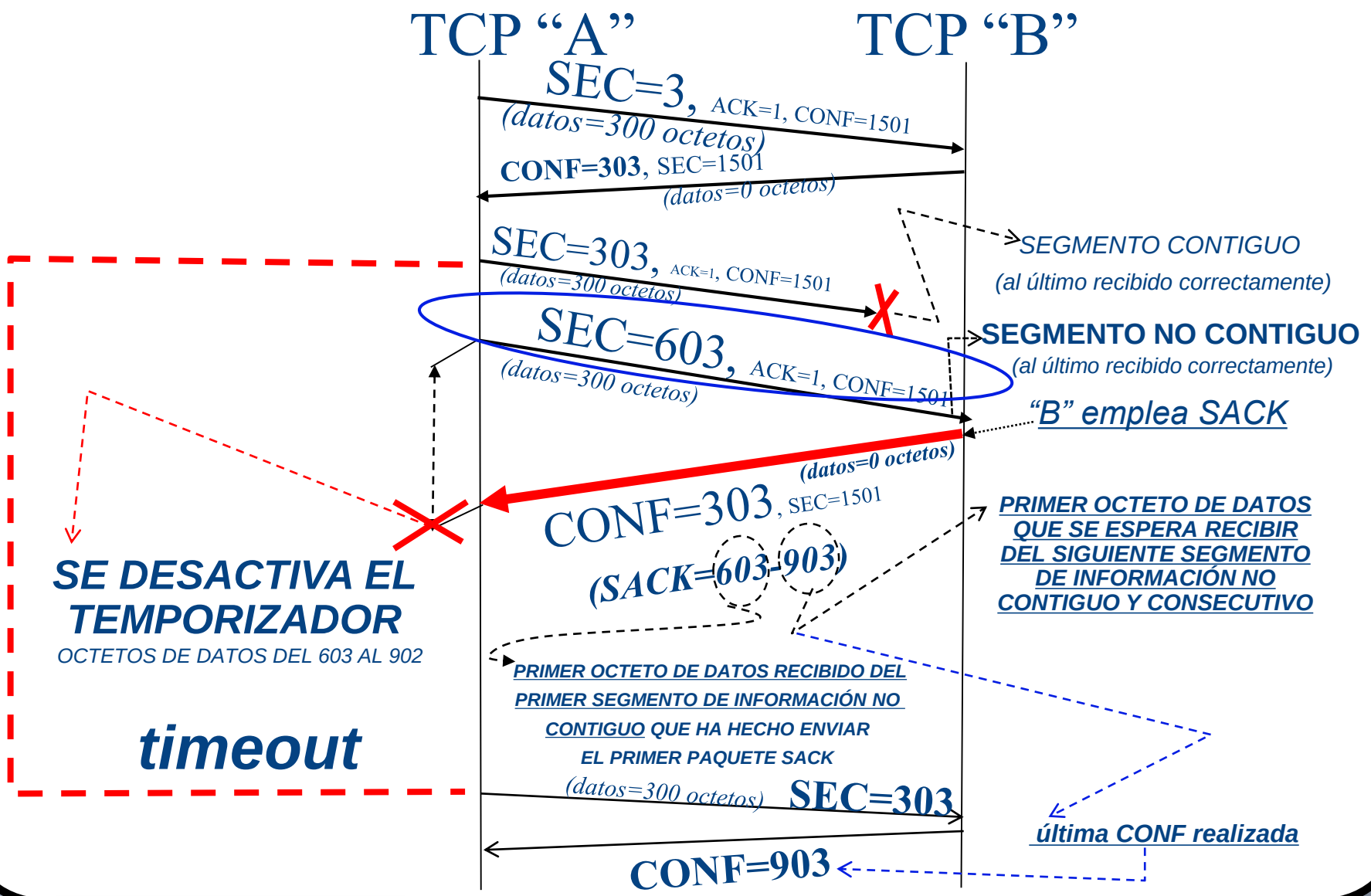
- ***Para poder hacer uso de la opción Tipo 5 SACK en fase de transferencia de datos hay que indicarlo previamente en fase de establecimiento de la conexión TCP***
  - Se indica mediante la **opción *Tipo 4 TCP SACK-PERMITTED*** en la fase de establecimiento de la conexión TCP en un **segmento *SYN = 1***
  - **Si el receptor no ha recibido la *opción Tipo 4 TCP SACK-PERMITTED*** en el establecimiento de la conexión, **no debe usar la opción *Tipo 5 TCP SACK-PERMITTED* en la fase de transferencia de datos**

# EJEMPLO DE RETRANSMISIÓN INNECESARIA DE SEGMENTOS NO CONTIGUOS



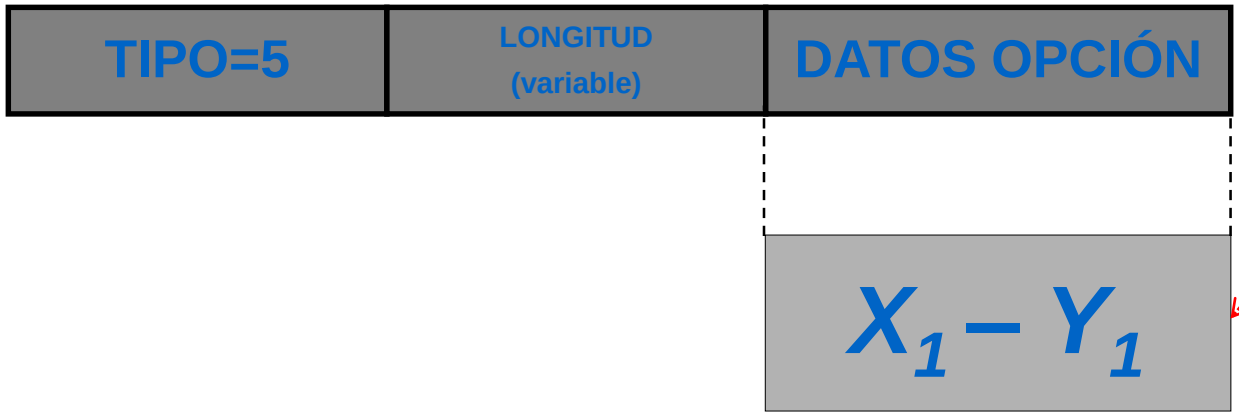
# Ejemplo de un Paquete SACK

*Un segmento no contiguo es aquél cuyo primer octeto de datos no es el primero que se espera recibir*



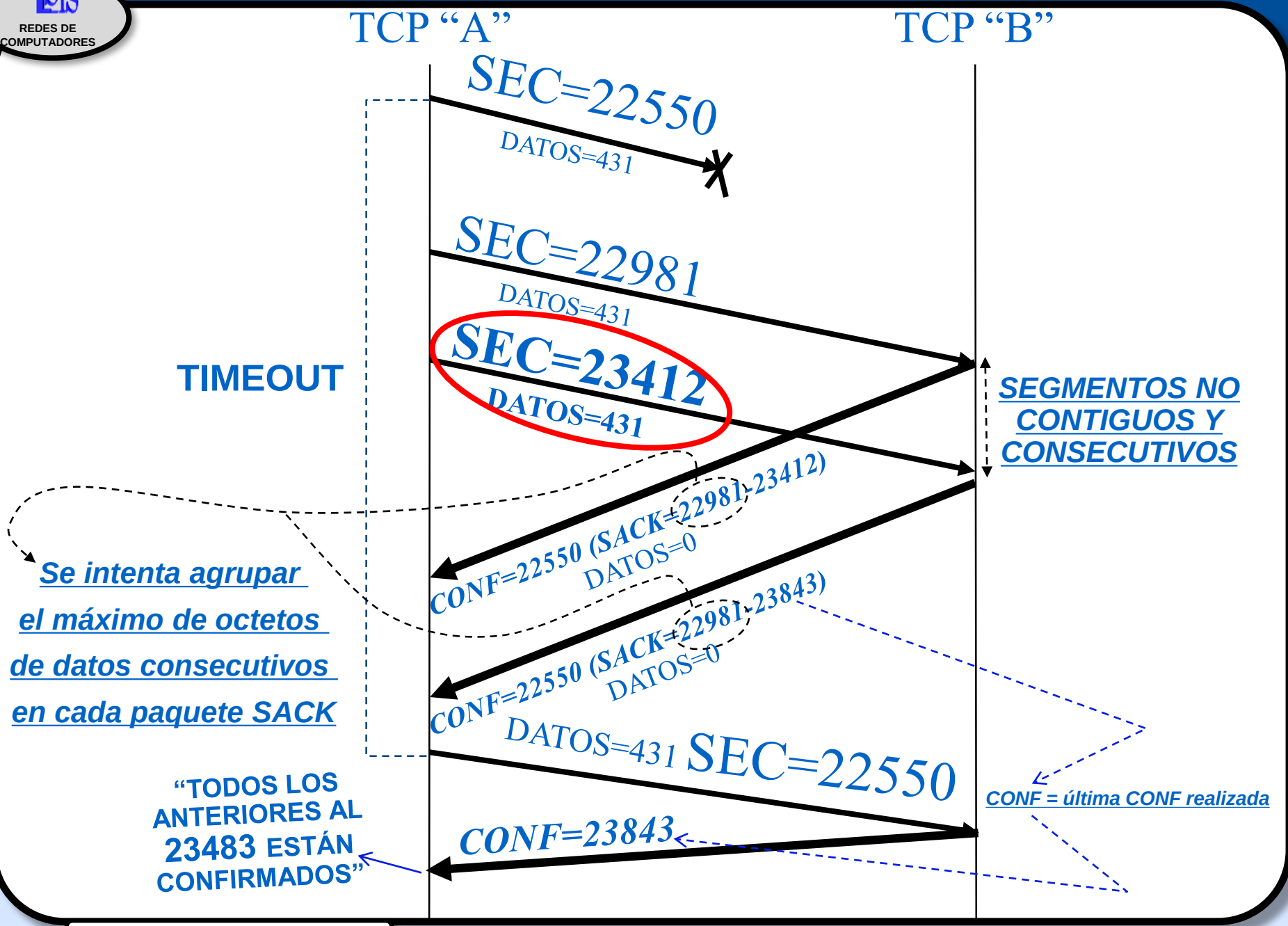
# Paquete SACK con "1 Bloque" para Segmentos No Contiguos y Consecutivos (Sin Pérdidas)

## USO DE LA OPCIÓN SACK CON CAMPO DATOS OPCIÓN DE UN ÚNICO BLOQUE

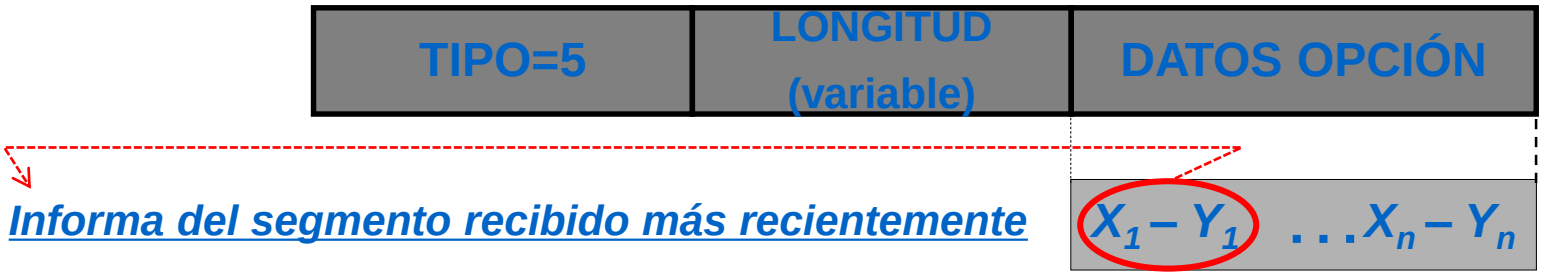



- ✓ *"$X_1$" (32 bits) = BORDE IZQUIERDO DEL BLOQUE "1" = "PRIMER" OCTETO DE DATOS RECIBIDO DEL "PRIMER" SEGMENTO DE INFORMACIÓN NO CONTIGUO*
  - *$X_1$ es el mismo valor para todos los paquetes SACK siempre que los siguientes segmentos de información no contiguos recibidos sean consecutivos*
- ✓ *"$Y_1$" (32 bits) = BORDE DERECHO DEL BLOQUE "1" = PRIMER OCTETO DE DATOS QUE SE ESPERA RECIBIR DEL SIGUIENTE SEGMENTO DE INFORMACIÓN NO CONTIGUO Y CONSECUTIVO*

➡ • ***Si los octetos de datos recibidos, de los siguientes segmentos de información no contiguos, son consecutivos, entonces, se agrupan éstos en el mismo bloque $X_1$-$Y_1$***


TCP "A"
TCP "B"
SEC=22550
DATOS=431
SEC=22981
DATOS=431
TIMEOUT
SEC=23412
DATOS=431
SEGMENTOS NO CONTIGUOS Y CONSECUTIVOS
CONF=22550 (SACK=22981-23412)
DATOS=0
Se intenta agrupar el máximo de octetos de datos consecutivos en cada paquete SACK
CONF=22550 (SACK=22981-23843)
DATOS=0
DATOS=431 SEC=22550
CONF = última CONF realizada
CONF=23843
"TODOS LOS ANTERIORES AL 23483 ESTÁN CONFIRMADOS"

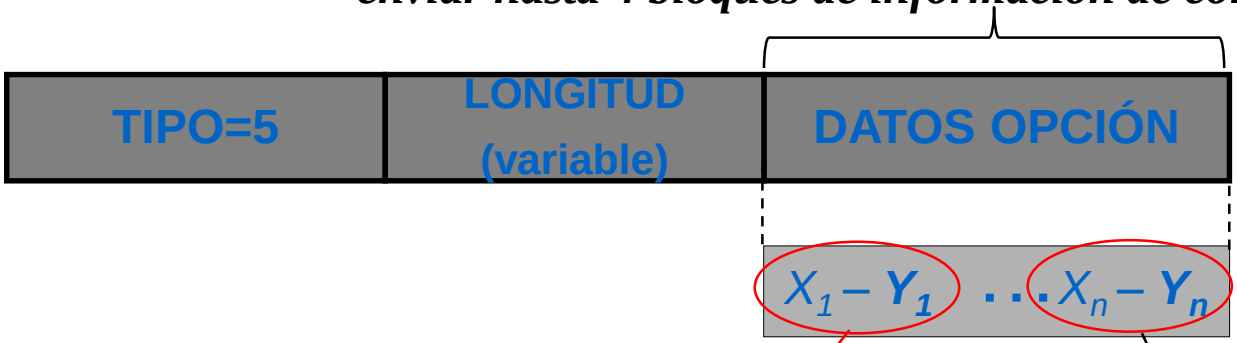
# Extensión con "N" Bloques del Paquete SACK para Segmentos No Contiguos y No Consecutivos (Con Pérdidas)




- ✓ *Se pueden enviar hasta 4 bloques NO CONSECUTIVOS SACK (32 octetos)*
  - *Se recuerda que si los nuevos octetos de datos son consecutivos con los anteriores ya recibidos, entonces, se agrupan en un mismo primer bloque $X_1$-$Y_1$*
- ✓ *El bloque 1 ($X_1$ – $Y_1$) debe informar del segmento recibido más recientemente*
  - ✓ *Asegura que el ACK con la opción SACK refleje el cambio más reciente en el buffer de recepción.* **Así el emisor tiene información actualizada del estado de la red y del estado de la cola de recepción**
- ✓ *Después del primer bloque SACK, los siguientes pueden estar listados en orden arbitrario*

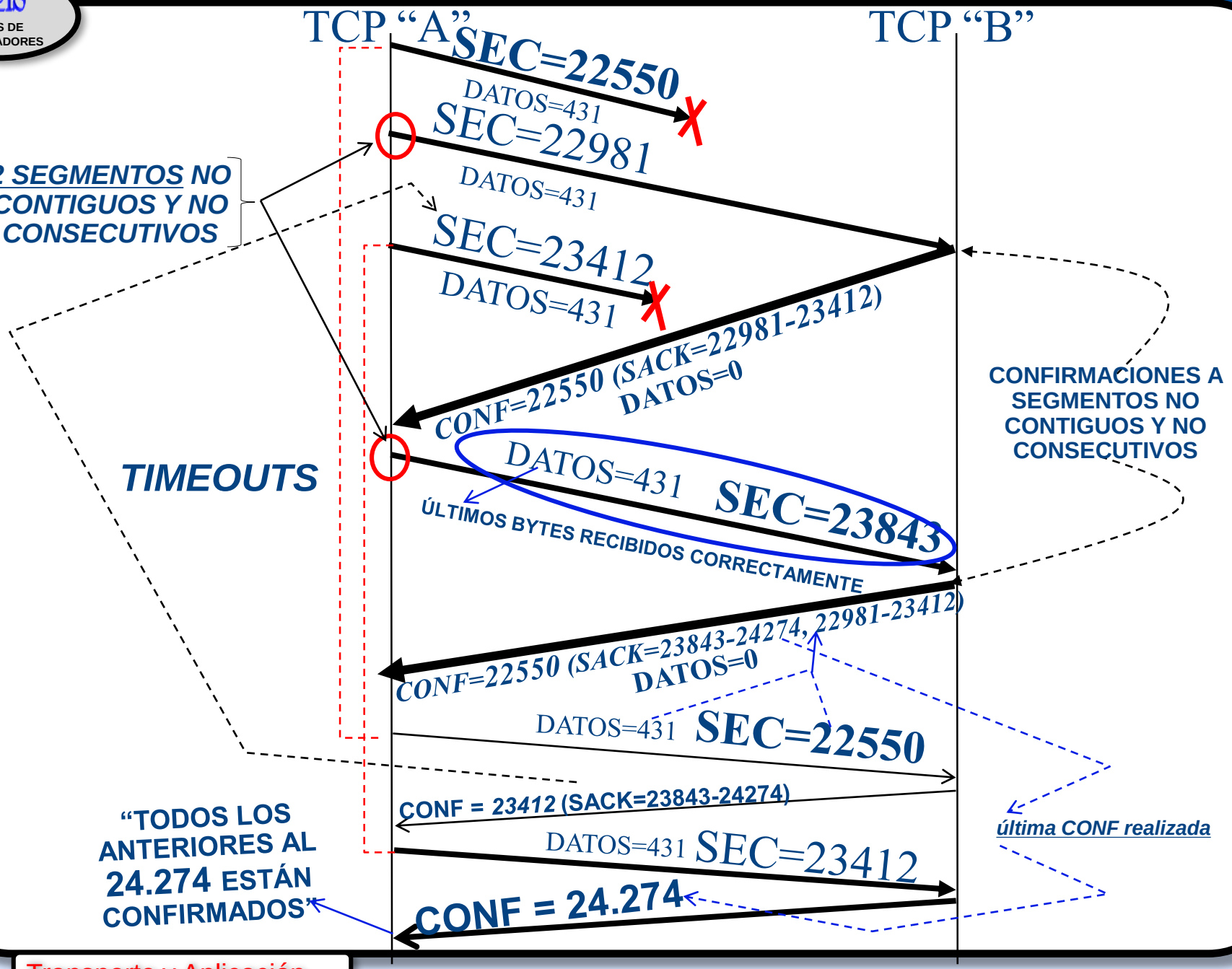
# Extensión con "N" Bloques del Paquete SACK para Segmentos No Contiguos y No Consecutivos (Con Pérdidas) (RESUMEN)

***Si hay espacio en el campo de opciones TCP (máximo 60 octetos), se pueden enviar hasta 4 bloques de información de control***




***RECORDATORIO DE UN SEGMENTO DE INFORMACIÓN NO CONTIGUO YA RECIBIDO***

***El bloque ($X_1 - Y_1$) debe informar del segmento no contiguo recibido más recientemente***


TCP "A"
TCP "B"
SEC=22550
DATOS=431
SEC=22981
DATOS=431
SEC=23412
DATOS=431
2 SEGMENTOS NO CONTIGUOS Y NO CONSECUTIVOS
CONF=22550 (SACK=22981-23412)
DATOS=0
CONFIRMACIONES A SEGMENTOS NO CONTIGUOS Y NO CONSECUTIVOS
TIMEOUTS
DATOS=431 SEC=23843
ÚLTIMOS BYTES RECIBIDOS CORRECTAMENTE
CONF=22550 (SACK=23843-24274, 22981-23412)
DATOS=0
DATOS=431 SEC=22550
CONF = 23412 (SACK=23843-24274)
última CONF realizada
DATOS=431 SEC=23412
CONF = 24.274
"TODOS LOS ANTERIORES AL 24.274 ESTÁN CONFIRMADOS"

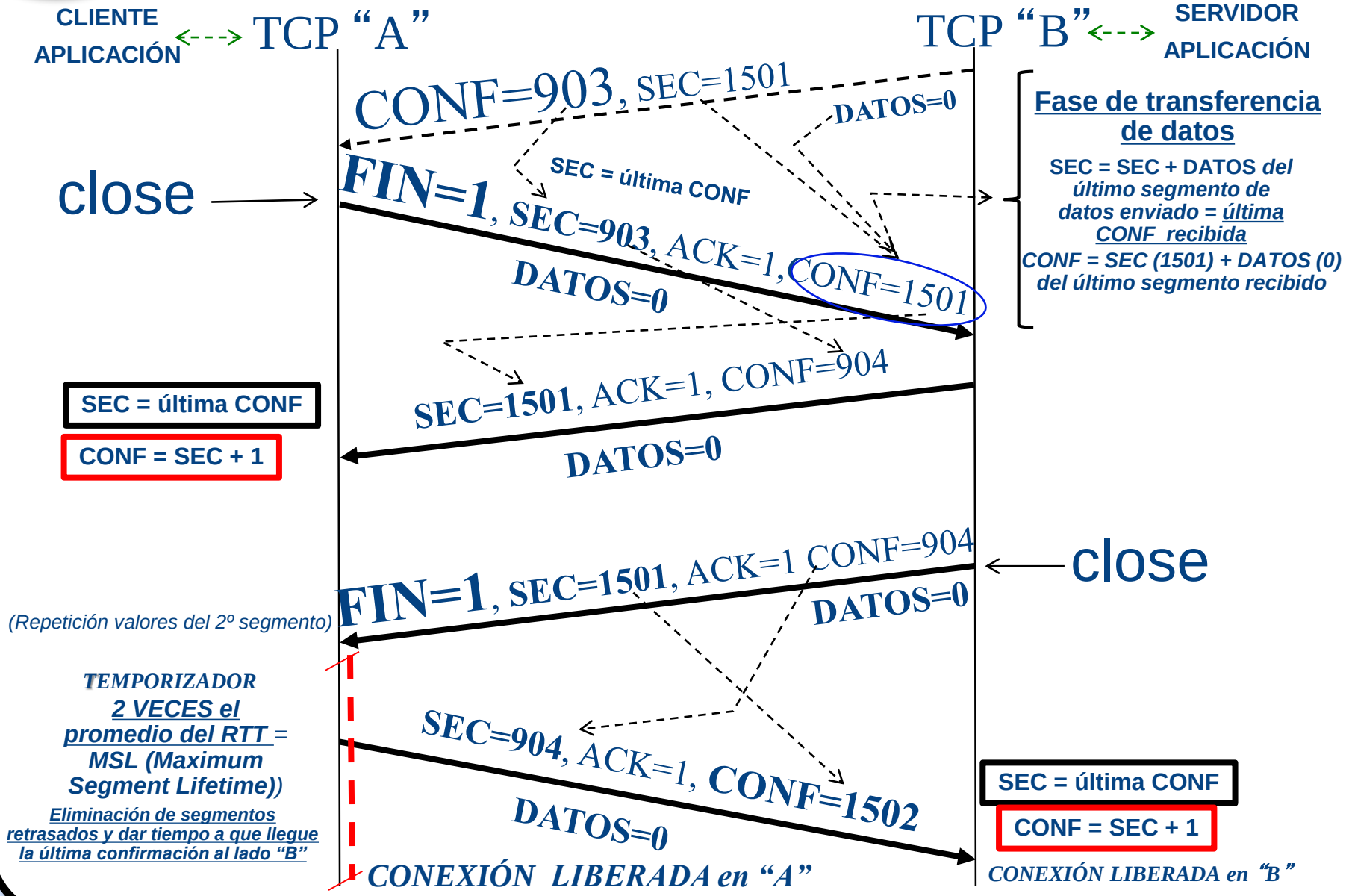
# 2 Tipos de Liberación de la Conexión

- **Bilateral u ordenada**
  - **Bit FIN en los dos sentidos (FIN = invitación al otro extremo a cerrar también)**
  - **La desconexión puede iniciarla cualquiera de los dos equipos (el cliente o el servidor) invitando al otro a cerrar**
  - **El cierre de un sentido por parte de un equipo se interpreta como una 'invitación' a cerrar al otro**
- **Unilateral o abrupta**
  - **Bit RST sin datos en un sentido**
    - **Un equipo termina y cierra sin esperar a recibir confirmación**
    - **El otro equipo se ve obligado a cerrar la conexión y eliminar todos los datos en buffers**

# Liberación Bilateral u Ordenada de la Conexión

- Estándar de Liberación: 4 envíos SIN DATOS
  - *LA CONEXIÓN TCP SE LIBERA COMPLETAMENTE cuando cada entidad TCP, después de haber transmitido todos sus datos, envía en cada sentido un SEGMENTO SIN DATOS con el bit FIN activado*
  - *La confirmación al FIN=1 recibido, SIEMPRE CON UN SEGMENTO SIN DATOS*

# Ejemplo de la Fase de Liberación de la Conexión



# Problemática de la Fiabilidad TCP

- ***Problema con TCP: Si se desea rapidez en el transporte, TCP retarda dicho transporte porque ofrece un servicio orientado a conexión FIABLE (tres fases y mecanismos de FIABILIDAD)***
- ***Hay aplicaciones que no toleran el retardo extremo a extremo producido por los ACKs, los temporizadores, las retransmisiones y controles de TCP***
  - **Las *retransmisiones TCP son inaceptables para aplicaciones en tiempo real, al incrementar el retardo extremo a extremo***
- ***Si la aplicación requiere un transporte rápido se monta sobre UDP***
- ***Cada vez hay más aplicaciones sobre UDP***
  - ***Tráfico interactivo en tiempo real***
    - ✓ ***Vídeoconferencias o vídeollamadas, VoIP, juegos en red, …***
  - ***Tráfico no interactivo en tiempo real***
    - ✓ ***Streaming de audio y vídeo***

# Protocolo UDP (User Datagram Protocol)
## *RFC-768, STD 0006*

- **Se utiliza en los siguientes escenarios:**
  - ✓ ***En aplicaciones en tiempo real***
    - ✓ ***interactivas (vídeoconferencias o vídeollamadas, VoIP, etc.) y no interactivas (streaming de audio y vídeo, etc.)***
  - ✓ **Cuando el *intercambio de mensajes es muy escaso* y los *mensajes son cortos*, por ejemplo, consultas al *DNS***
  - ✓ ***Los mensajes se envían en una RAL del tipo Ethernet* (*sin errores físicos*): *DHCP*, *SNMP*, …**
  - ✓ ***Cuando los mensajes se producen regularmente y no importa si se pierde alguno*: *SNMP*, *NTP*, …**
  - ✓ ***Cuando se envía tráfico de broadcast/multicast***

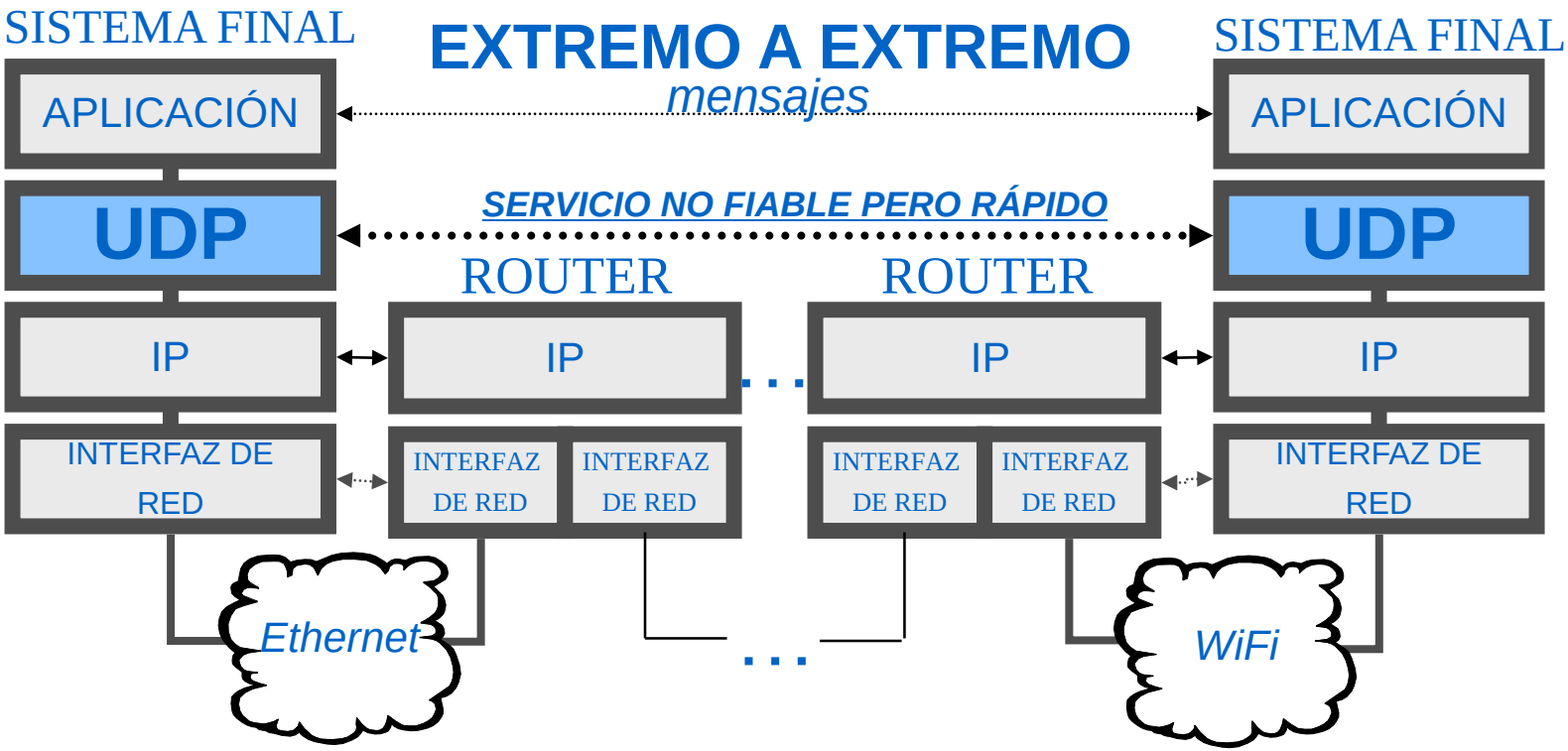
# Protocolo UDP (User Datagram Protocol)

*RFC-768, STD 0006*

***Transporte NO FIABLE PERO RÁPIDO***

***de los mensajes de aplicación encapsulados en datagramas UDP***

## SIN CONTROLES




Las unidades de datos del protocolo UDP se denominan
***datagramas UDP* o *mensajes UDP***

# Protocolo UDP (User Datagram Protocol)

## *RFC-768, STD 0006*

- ***Protocolo muy sencillo que añade un mínimo de sobrecarga***
  - **Añade muy poco al servicio de IP como es proporcionar comunicación proceso a proceso en lugar de máquina a máquina**
- **3 Servicios**
  - ***NO FIABLE***
    - ***Con detección (opcional) y no recuperación de errores físicos***
    - ***Sin control (detección y recuperación) de errores lógicos (datagramas UDP perdidos y desordenados)***
    - ***Sin control de flujo***
  - **MULTIPLEXADO o simultáneo y diferenciado a través de los números de puerto**
  - **DÚPLEX: TRANSFERENCIAS SIMÚLTÁNEAS EN LOS DOS SENTIDOS**

# UDP ofrece un SERVICIO NO FIABLE a los Procesos del Nivel de Aplicación

- Un SERVICIO NO FIABLE significa un SERVICIO NO ORIENTADO A CONEXIÓN, el cual dispone de UNA FASE
  1. TRASFERENCIA DE DATOS

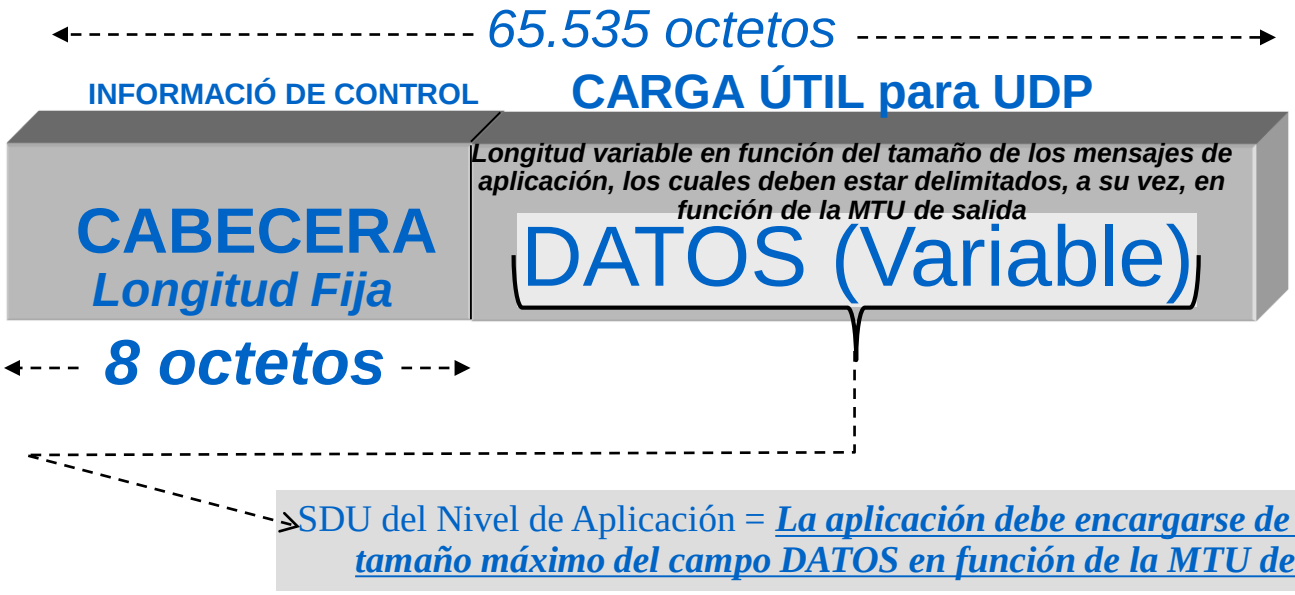
# PROTOCOLO UDP

## Formato de un Datagrama UDP

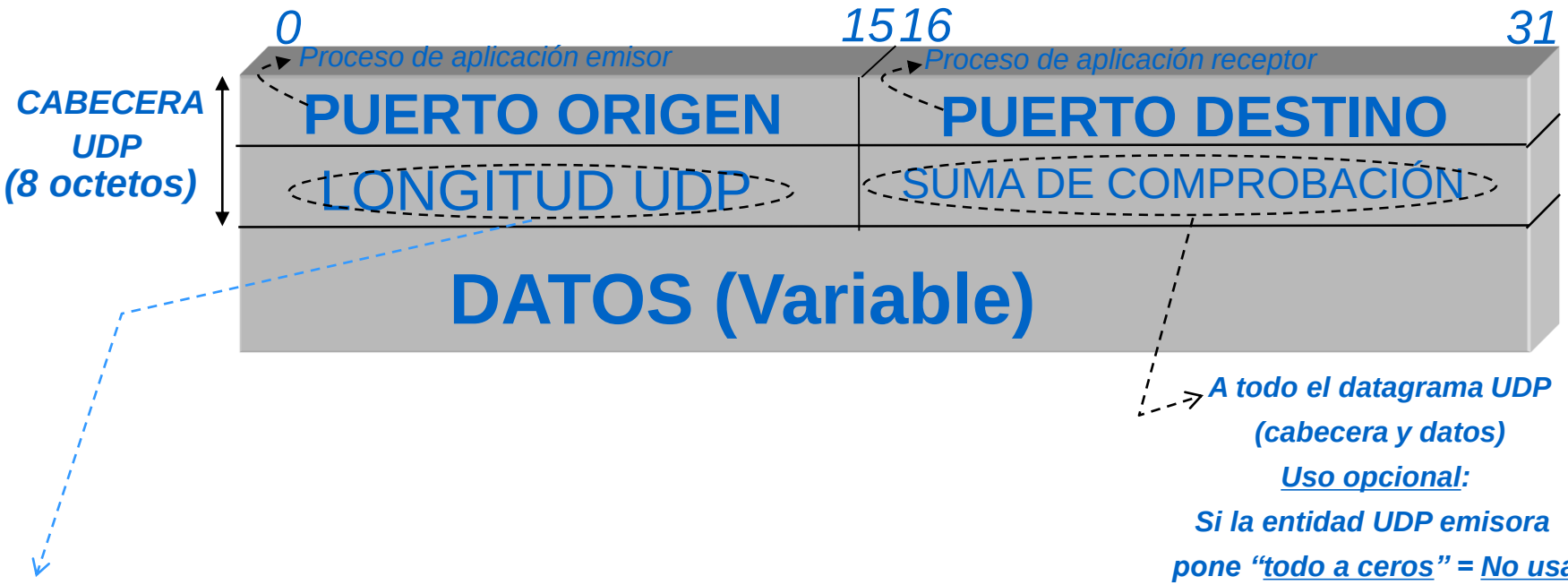
*RFC-768, STD 0006*

*Engloba dos tipos de información: **Cabecera** + **Datos***

## Longitud Máxima Teórica



# Formato de un Datagrama UDP



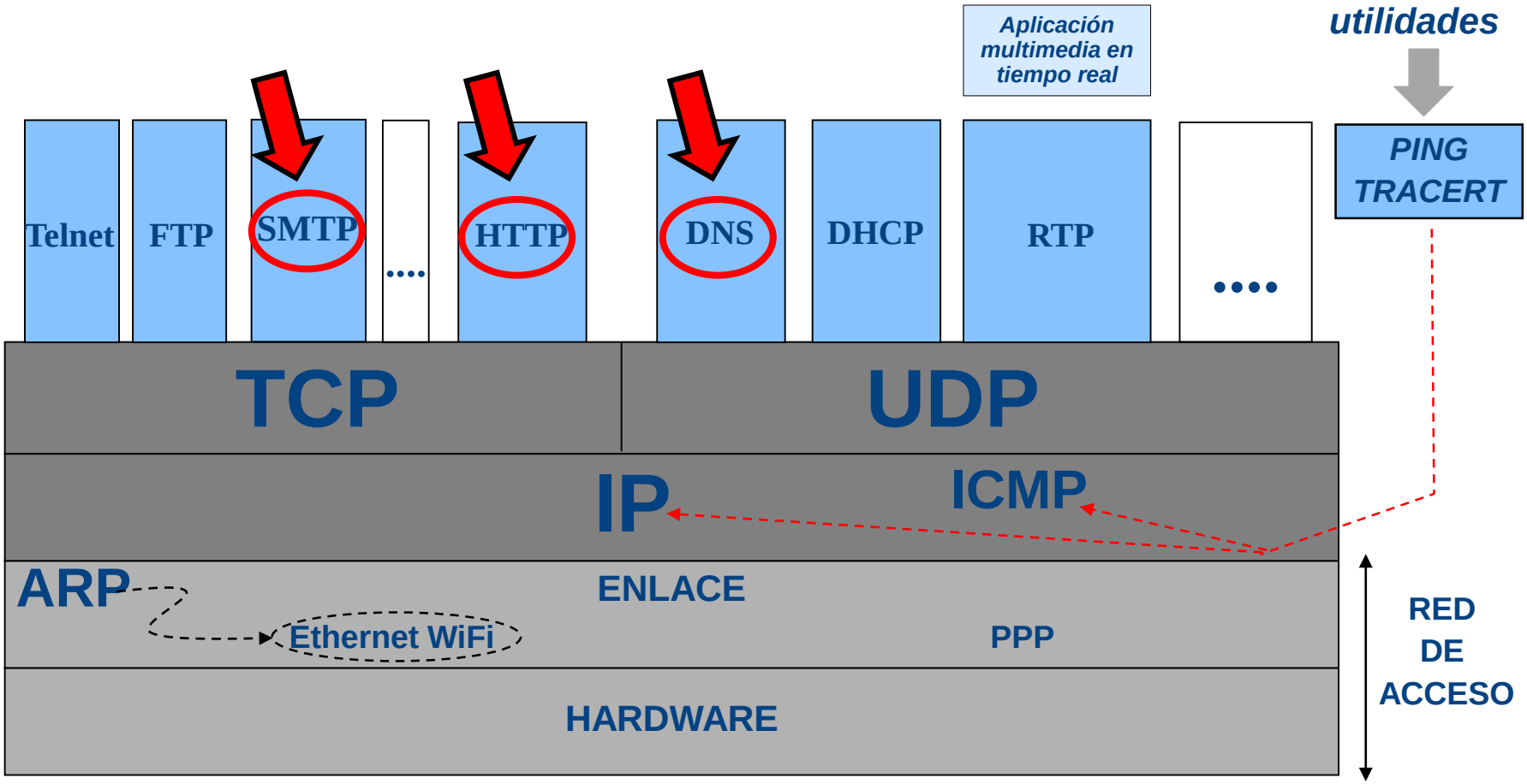

***Longitud en octetos del datagrama UDP (cabecera + datos) =***

= ***Longitud mínima de 8 octetos (cabecera sin datos)***

= ***Longitud máxima TEÓRICA de 65.535 octetos*** *pero la longitud total debe ser menor debido a que la* ***MTU = 1500 octetos***

# 4.2 Nivel de aplicación

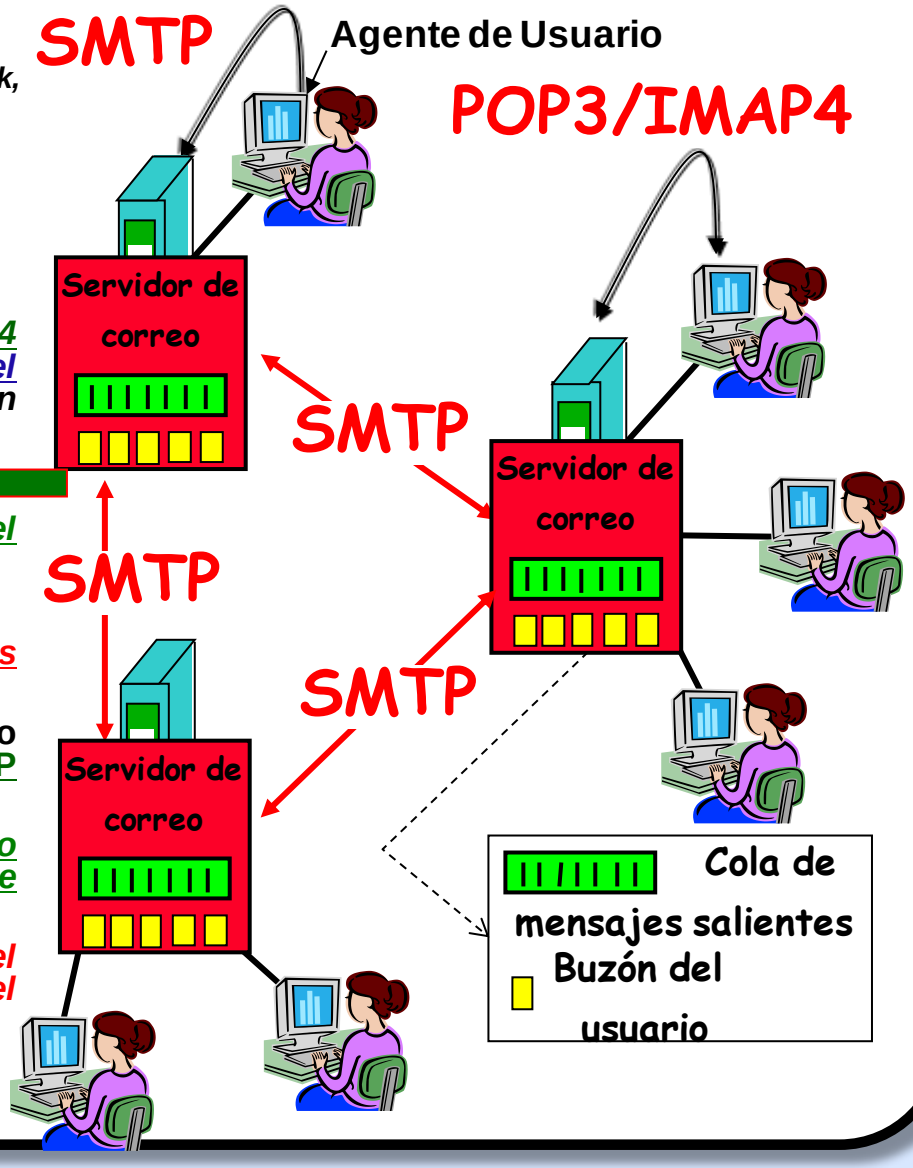
# Protocolos de Aplicación sobre TCP



# APLICACIÓN DE CORREO ELECTRÓNICO TCP/IP

## Tres Componentes Principales

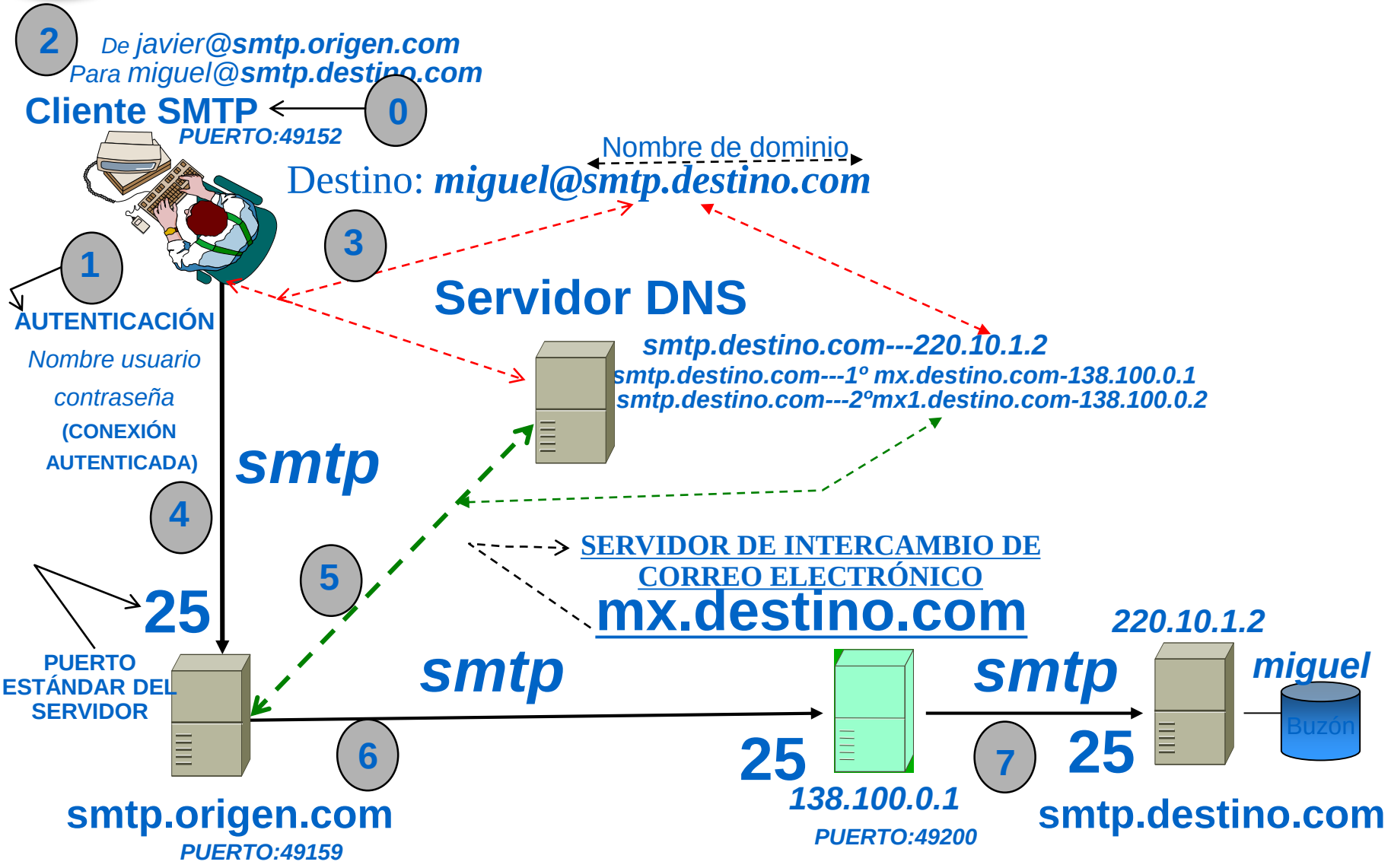
1. **AGENTE DE USUARIO** *(p.ej., MicrosofL Office Outlook, Mail de OS X, Kmail y Evolution de Linux, etc.)*
   - *Sistema de correo en el equipo del usuario*
   - *CLIENTE DE CORREO SMTP*
   - *Editor de texto*
   - *Codificador/decodificador o CODEC MIME*
   - *CLIENTE de acceso al correo POP3/IMAP4 para recuperar el correo desde el buzón del destinatario en su servidor de correo a un directorio de su disco duro*
2. **SERVIDOR DE CORREO SMTP del usuario**
   - *Se ejecuta en el equipo de la organización del usuario o en la red IP de su operador (ISP)*
   - *Buzones de los usuarios*
   - *Colas de los buffers de los mensajes salientes*
3. Protocolo Simple de Transferencia de Correo o **PROTOCOLO DE ENVÍO DE CORREO SMTP** (Simple Mail Transfer Protocol)
   - *Enviar correo desde el cliente de correo SMTP de un agente de usario a su servidor de correo SMTP*
   - *Enviar correo desde el servidor origen o del remitente al servidor destino (buzón del destinatario)*



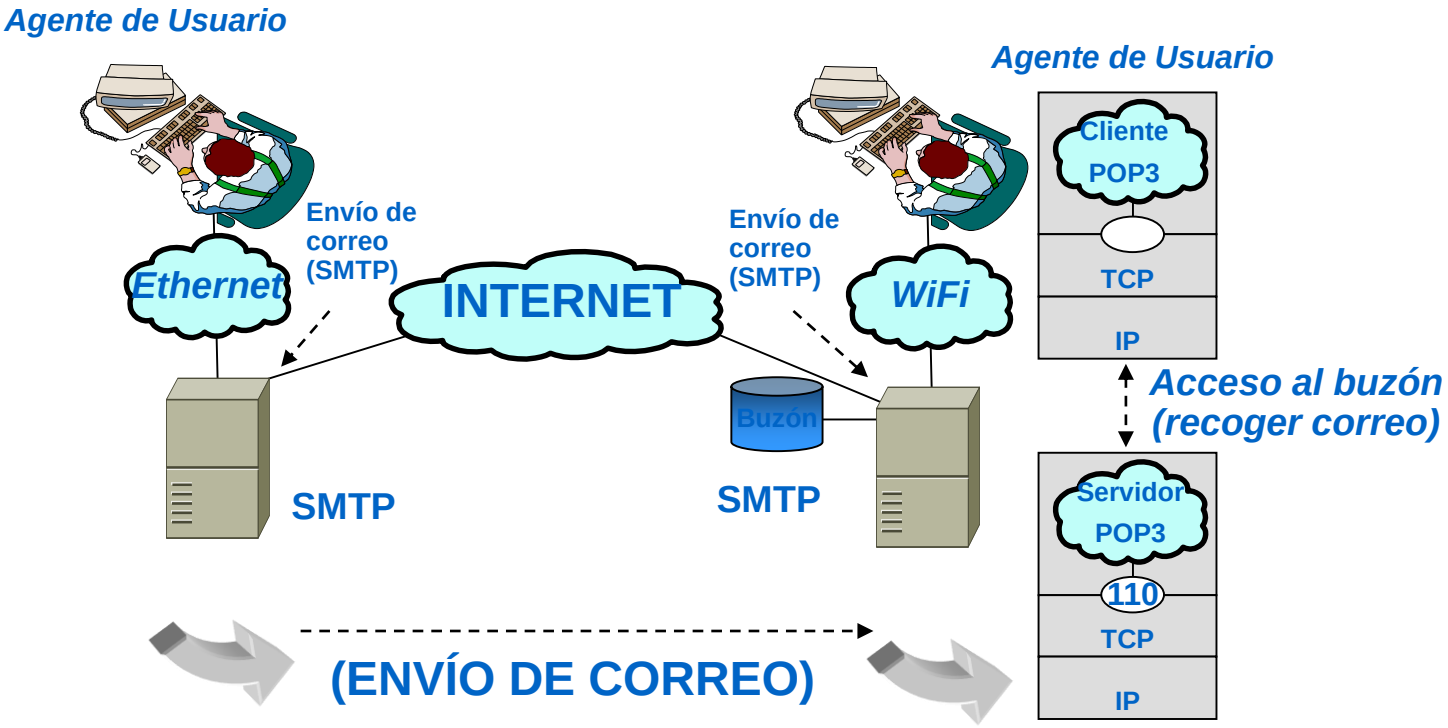
# COMPONENTE ADICIONAL
# SERVIDOR DE INTERCAMBIO DE CORREO

- *Cuando el buzón del destinatario no está localizado en el equipo en donde se ejcuta el servidor SMTP local*
  - El Servidor de Intercambio de Correo SIEMPRE ESTÁ ASOCIADO AL SERVIDOR SMTP REMOTO DEL DESTINATARIO y SIEMPRE ES INVOCADO POR EL SERVER SMTP LOCAL DEL REMITENTE VÍA SERVIDOR DNS
    - ✓ Registro MX (Mail eXchange record) o Registro de Intercambio de Correo en la Tabla DNS
  - OBJETIVO: ALMACENAR PREVIAMENTE LOS MENSAJES PARA EL SERVIDOR SMTP REMOTO DESTINATARIO SI ÉSTE NO ESTÁ EJECUTÁNDOSE o SI EL EQUIPO NO ESTÁ ENCENDIDO

# EJEMPLO DE ESCENARIO SMTP EN INTERNET



# RECOGIDA DEL CORREO ELECTRÓNICO
# POP3 (Post Office Version 3)
## Componente del AU

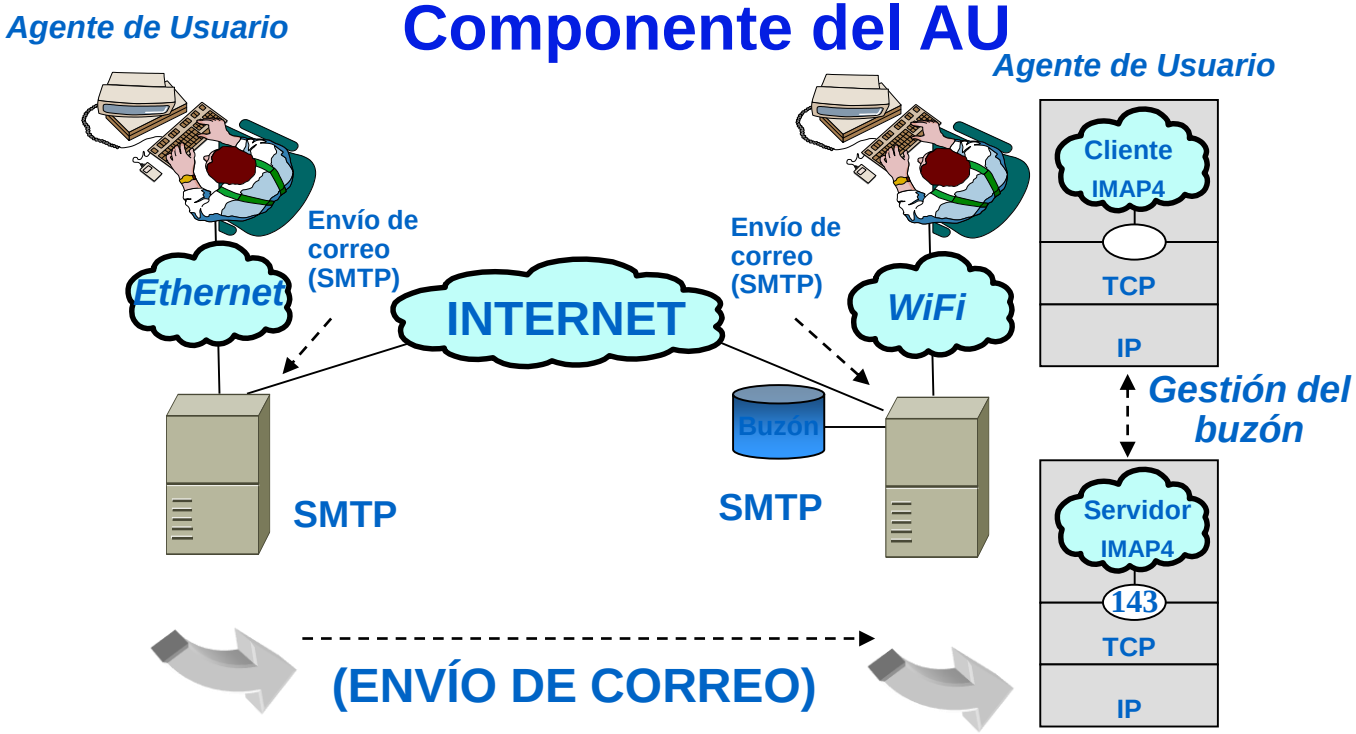



**Proporciona un SERVICIO DE RECOGIDA DE TODOS LOS MENSAJES existentes en el buzón de correo del usuario en su servidor de correo *a un directorio de un disco duro de su máquina local***

# GESTIÓN DEL CORREO ELECTRÓNICO

## IMAP4 (Internet Message Access Protocol Rev 4)

### Componente del AU




- **Proporciona un SERVICIO DE GESTIÓN DE MENSAJES en el mismo buzón de correo sin necesidad de recoger todos los mensajes y traerlos al disco duro**
- **Permite al usuario *clasificar, eliminar, y distribuir su correo en distintas carpetas en el disco duro de la propia máquina servidora de correo***
- **Asimismo, permite al usuario *copiar o mover mensajes*, previamente seleccionados, *desde su buzón hasta el disco duro de su computadora*, distribuyéndolos en las carpetas locales deseadas**

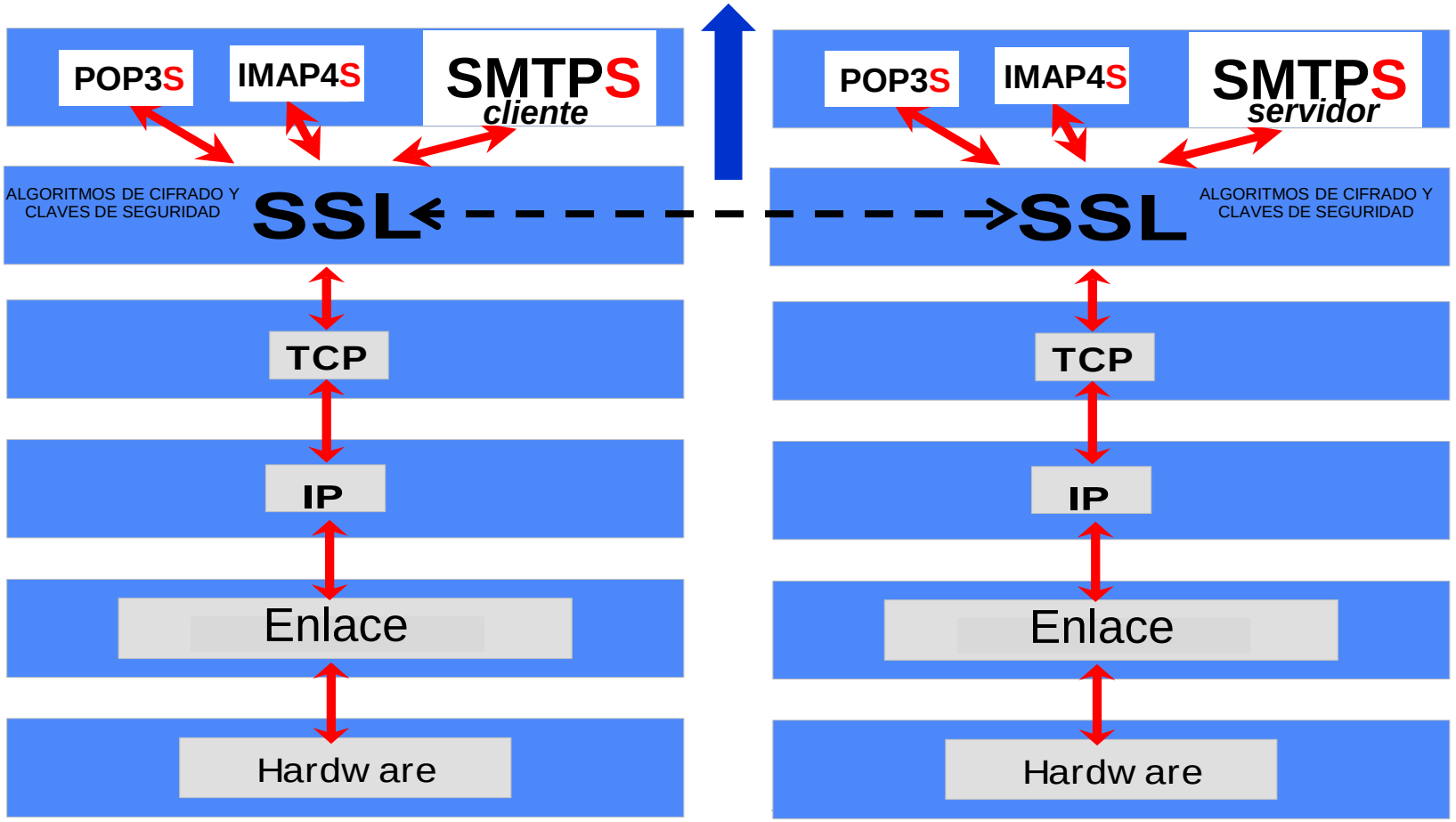
# Puertos Servidores SMTP

- 25
  - *Nº de puerto por omisión pero NO ES SEGURO (permisivo o "tragón")*
    1. *AUTENTICACIÓN NO SEGURA: Nombre de usuario y contraseña visibles*
    2. *Envía cualquier mensaje transmitido por su cliente SMTP*
       - **Peligro de que el cliente SMTP transmita, voluntariamente o involuntariamente por infección previa,** *correo spam ("basura" de tipo publicitario)* **o** *correo con virus (fichero adjunto que se ejecuta al abrirlo)*
       - **Peligro de que el** *servidor de correo origen entre en una Lista Negra de servidores SMTP de envío de correo peligrosos en Internet*
- 465
  - *Nº DE PUERTO SEGURO*
    1. *AUTENTICACIÓN SEGURA: Cifrado del nombre de usuario y contraseña*
    2. *Cifrado de todo el correo enviado del cliente SMTP al servidor SMTP*
    3. *Cifrado de todo el correo recibido por el servidor SMTP hacia el cliente SMTP vía IMAP4 o POP3*
  - *Permite el uso de FIREWALLS*
    - *Se pueden filtrar, o no, direcciones IP de clientes SMTP potencialmente peligrosos para el nº de puerto 465*
  - *FILTROS ANTISPAM*
  - *FILTROS ANTIVIRUS*
  - *LISTAS NEGRAS (mantenidas por organizaciones especializadas) de servidores SMTP de envío de correo peligrosos en Internet*

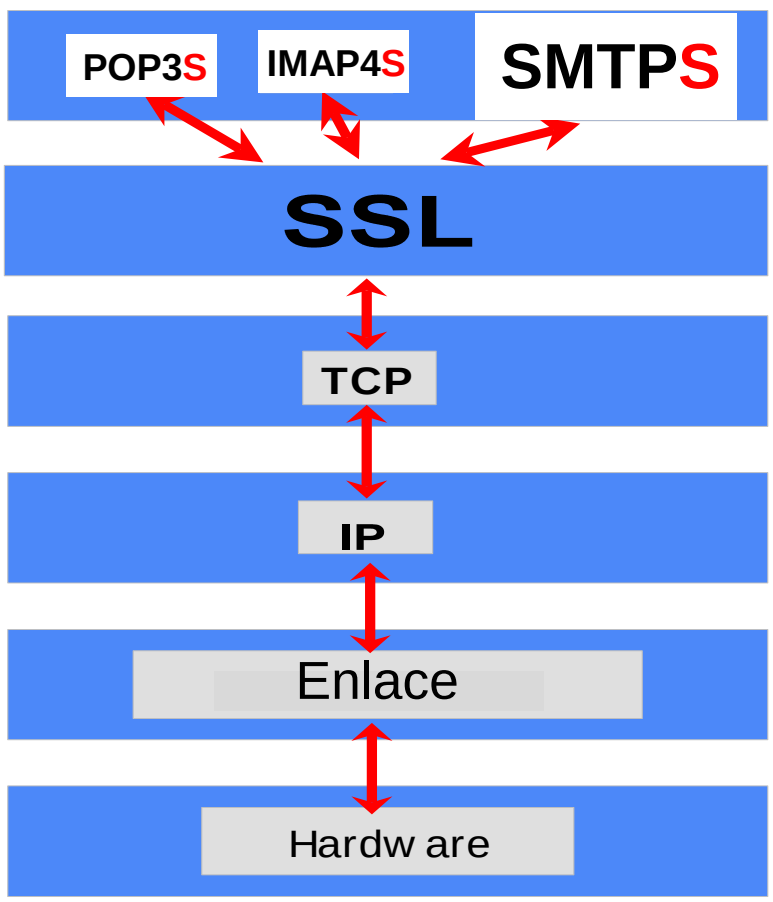
# SECURE SOCKETS LAYER (SSL)

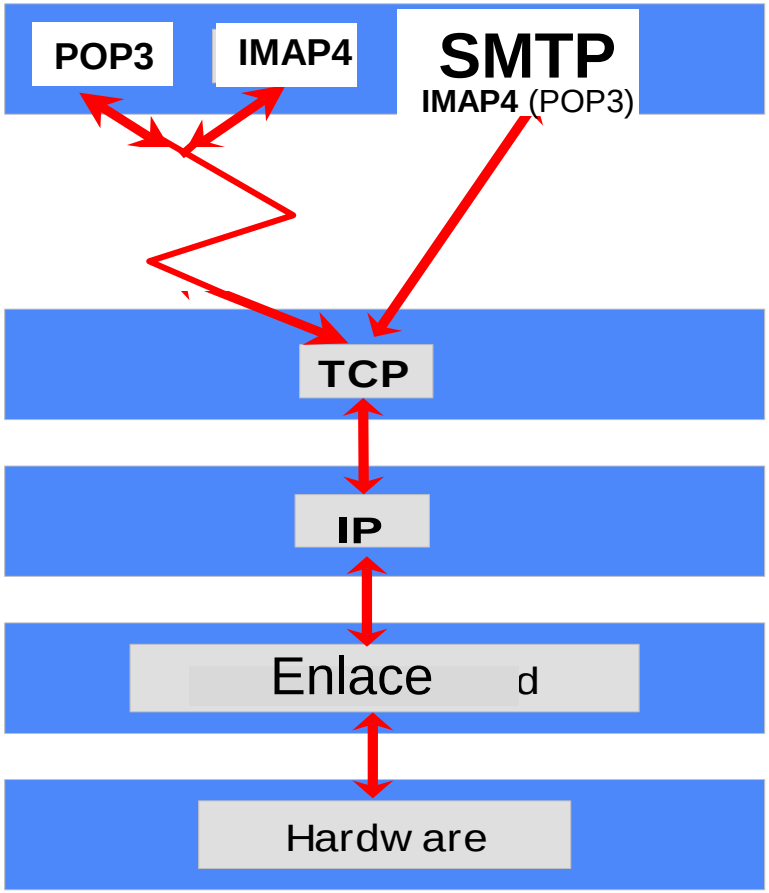
***Para poder usar el nº de puerto 465 en el envío de correo es necesario que tanto el cliente como el servidor SMTP, dispongan de un Nivel Intermedio de Seguridad (SSL) entre TCP y el proceso SMTP cliente y servidor***

***Negociación previa del algoritmo de cifrado (AES, 3DES, …) y longitud en bits de la clave compartida (128, 168, 192 y 256 bits) para cifrar y descifrar los mensajes de autenticación y correo de usuario***



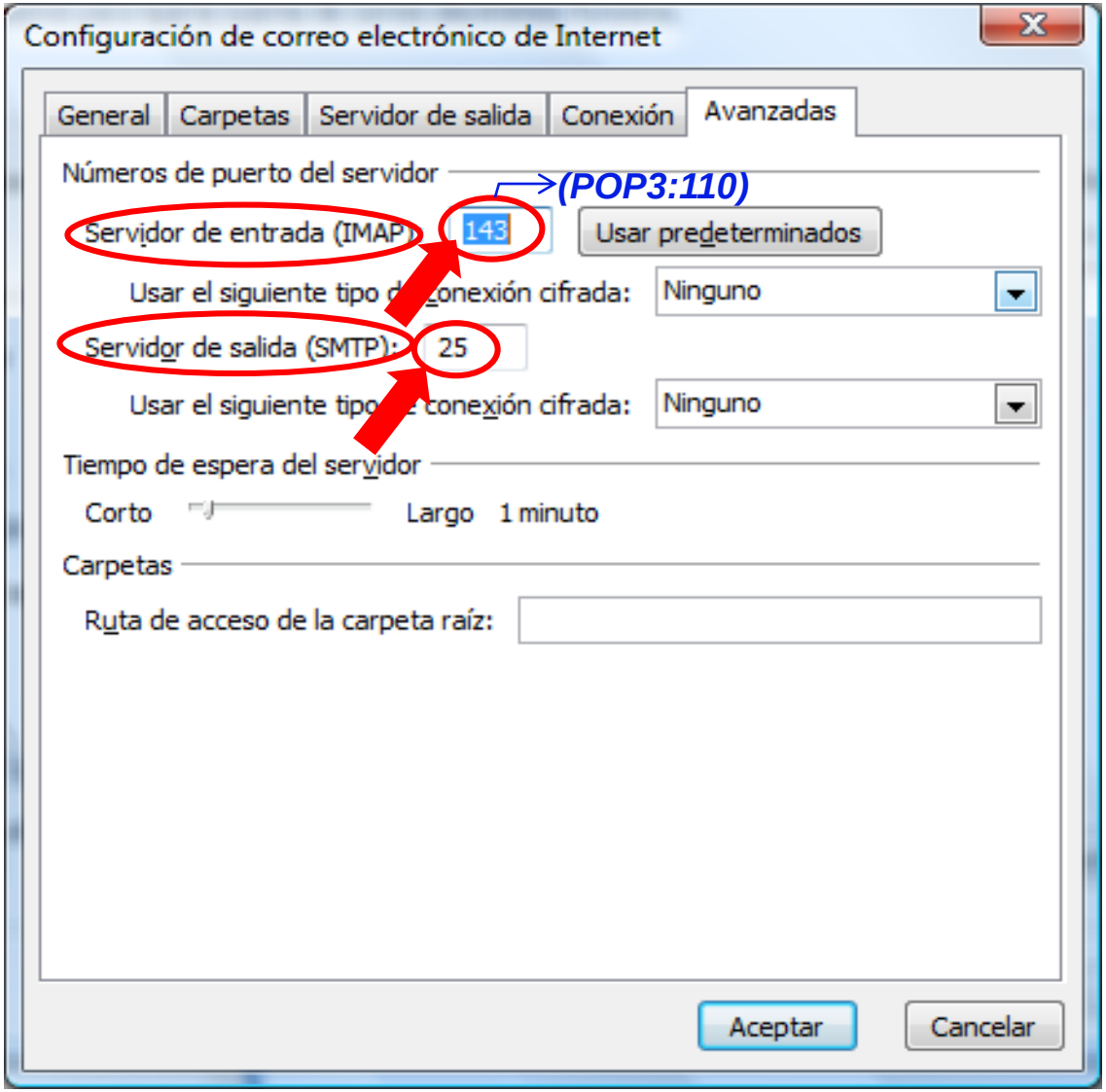
# Un Entorno de Correo (AU) Seguro vía SMTP/IMAP4 (POP3) y NO Seguro



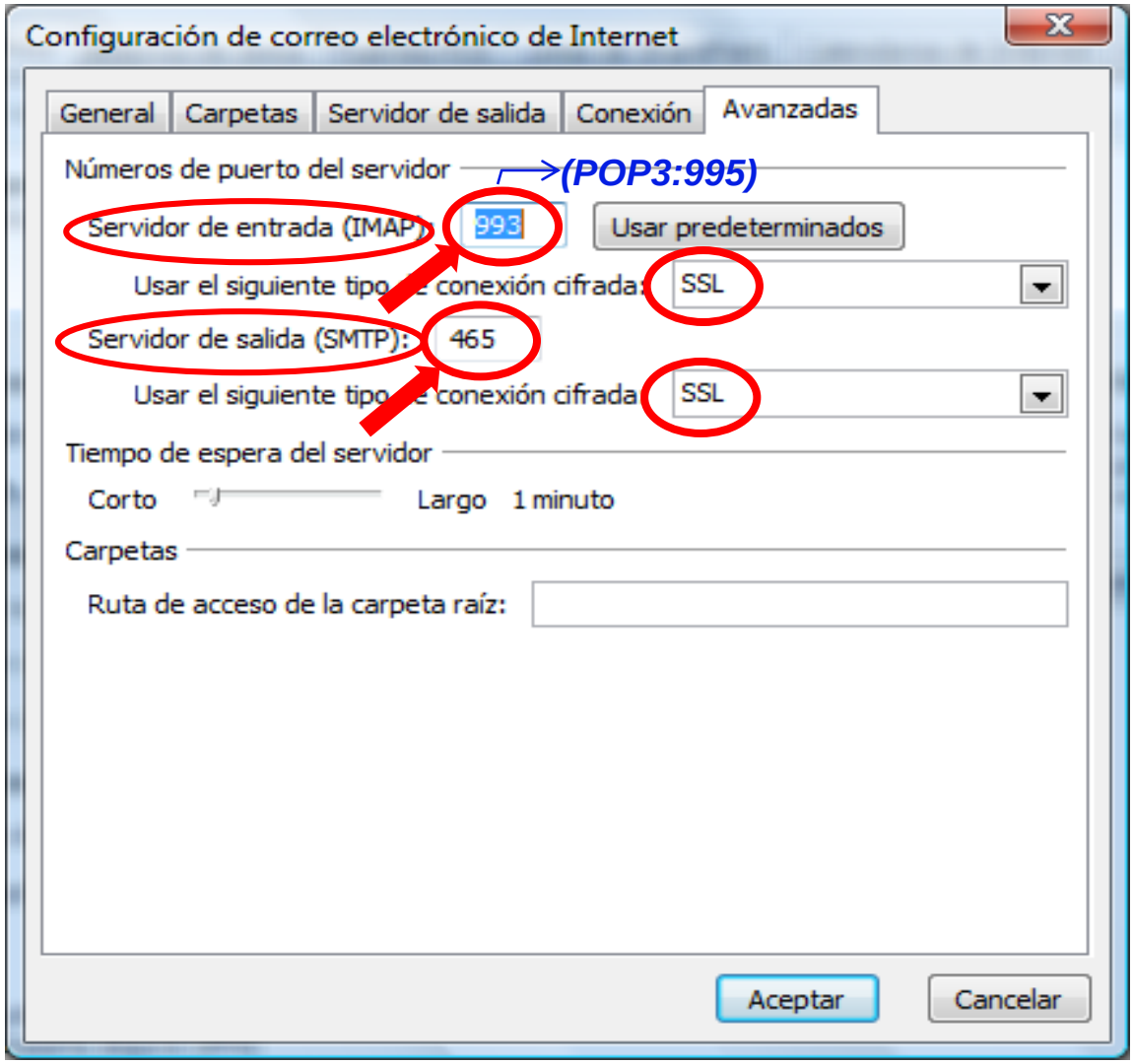
# CONFIGURACIÓN: Puerto 25 de SMTP y 143 de IMAP en Outlook

## Microsoft Office Outlook

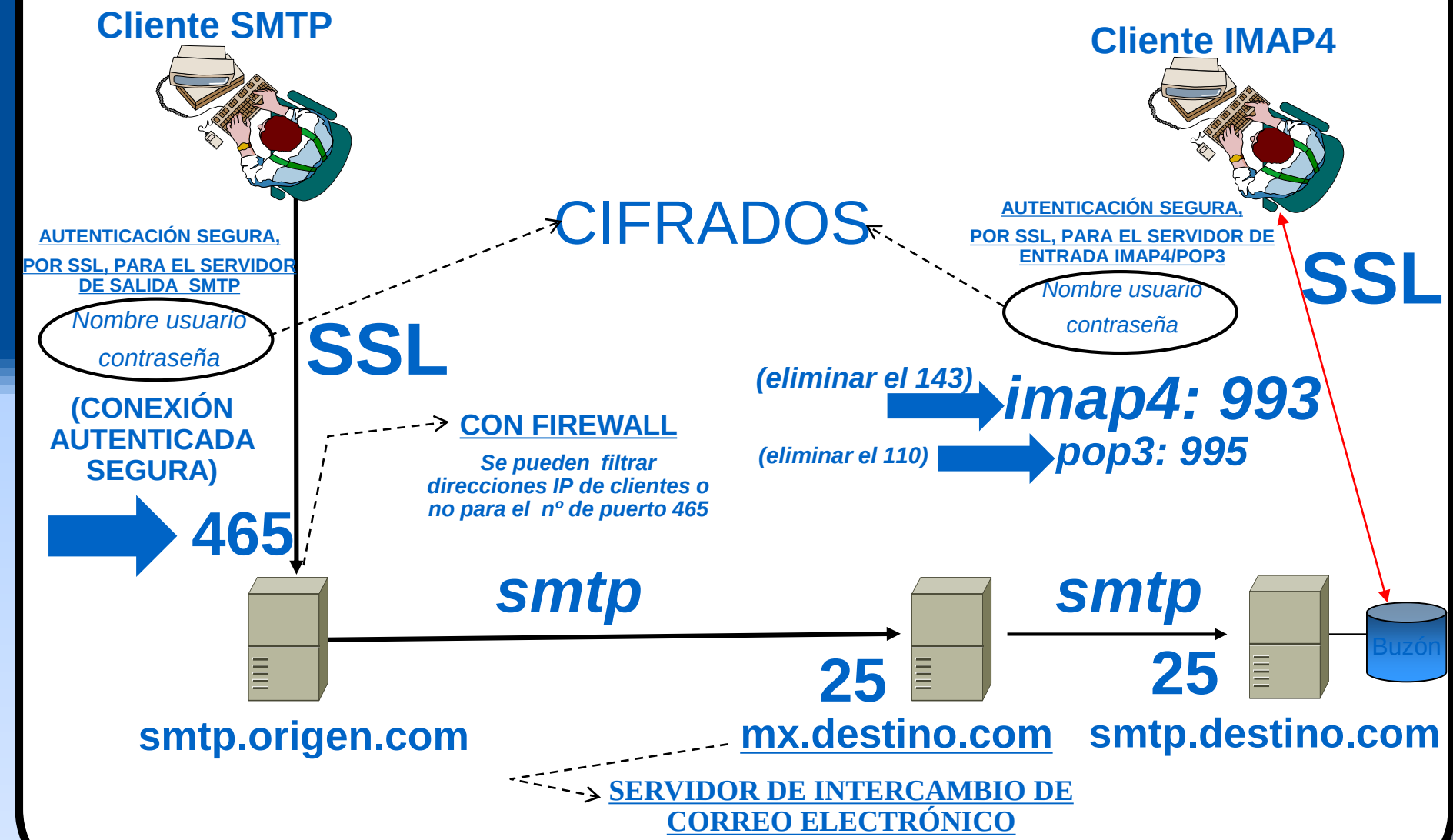


# CONFIGURACIÓN: Puertos 465 de SMTP y 993 de IMAP en Outlook

## Microsoft Office Outlook



# Un Correo Electrónico Seguro vía SSL

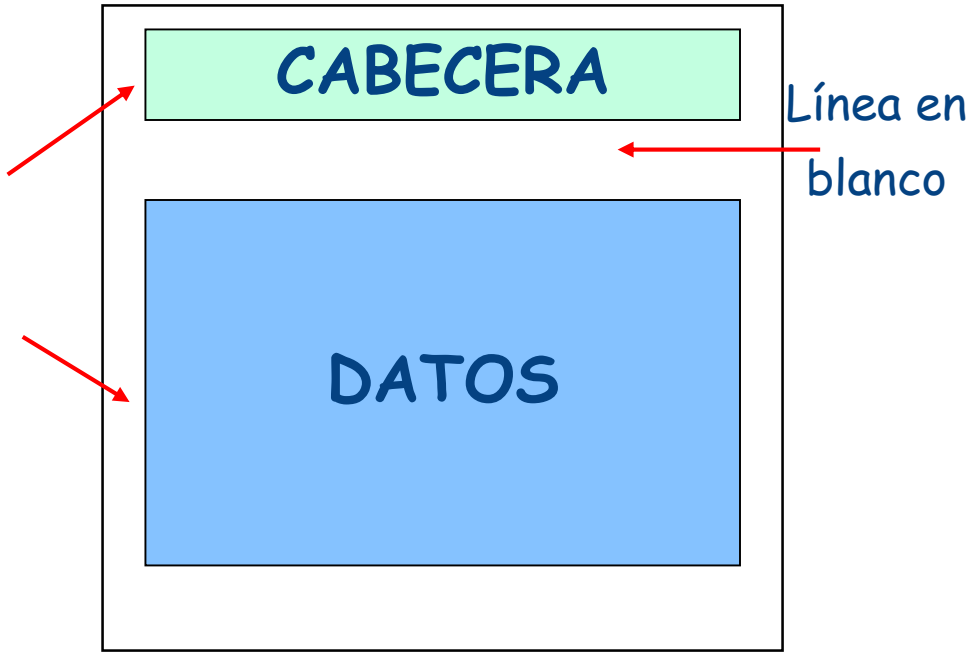


# FORMATO DE LOS MENSAJES SMTP

## RFC-822, STD-0011

*(actualizado por el draft standard RFC-5322)*

- RFC-822: Estándar para el formato de mensajes de texto. *Define las líneas o campos de la CABECERA (To:, Cc:, Bcc:, From:, Subject:, etc.), una línea en blanco y el Campo DATOS del mensaje en formato ASCII de 7 bits (español, francés, chino, etc., no soportados)*




***FORMATO DE LOS MENSAJES DE CORREO EN INTERNET***

***Para poder enviar mensajes NO ASCII (acentos, diéresis, ficheros .jpg, .mp3, .m4a oAAC, .wav, .m4v o MPEG-4, .h264 , etc.) en el cuerpo de un mensaje de correo se emplea el CODIFICADOR/DECODIFICADOR MIME***

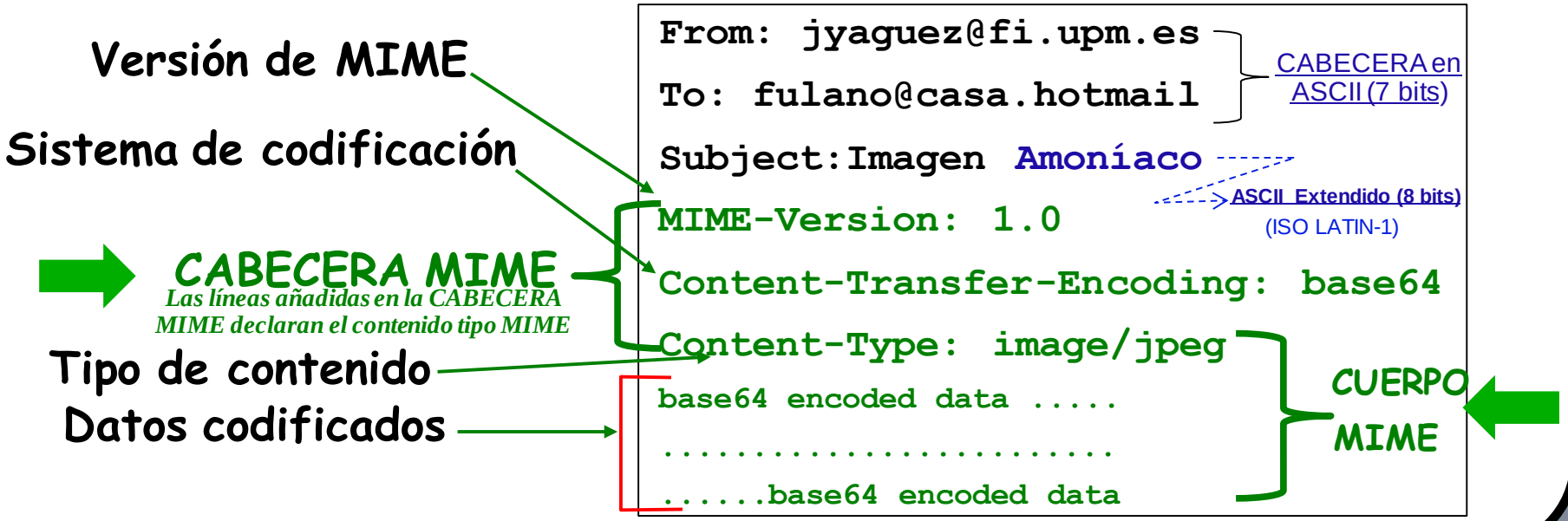
# Formato del Mensaje SMTP/MIME

## *(Multipurpose Internet Mail Extensions)*

## Extensiones Multipropósito de Correo de Internet

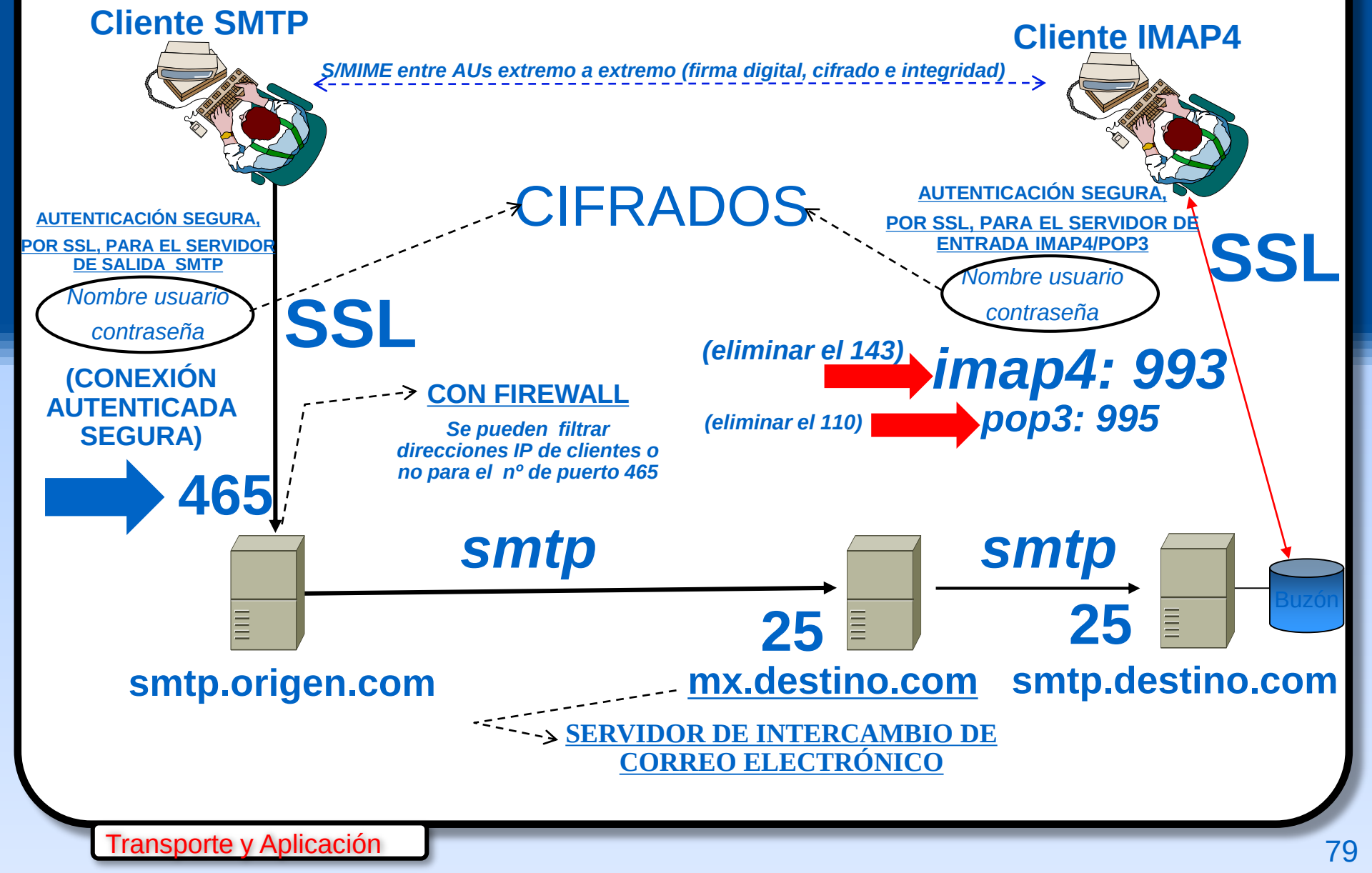
## Componente del AU

- ***Todo AU dispone de un CODIFICADOR (emisor)/DECODIFICADOR (receptor) MIME, el cual define el FORMATO DEL CAMPO DATOS DEL MENSAJE DE CORREO***
  - *Para ello, AGREGA, al CAMPO DATOS del mensaje, una CABECERA MIME y un CAMPO DATOS o CUERPO MIME de forma individual tanto para el texto como para cada uno de los ficheros incluidos en dicho mensaje*
  - *A su vez, el CODIFICADOR MIME emplea el sistema de codificación base64 o radix64 (subconjunto de 6 bits del ASCII de 7 bits; $2^6$ = 64 combinaciones o 64 caracteres base64)*
    - ✓ *El texto y los ficheros incluidos en el campo DATOS del mensaje SMTP ("attachments"), se codifican, sustituyendo grupos de 6 bits del texto o fichero original por un carácter base64*

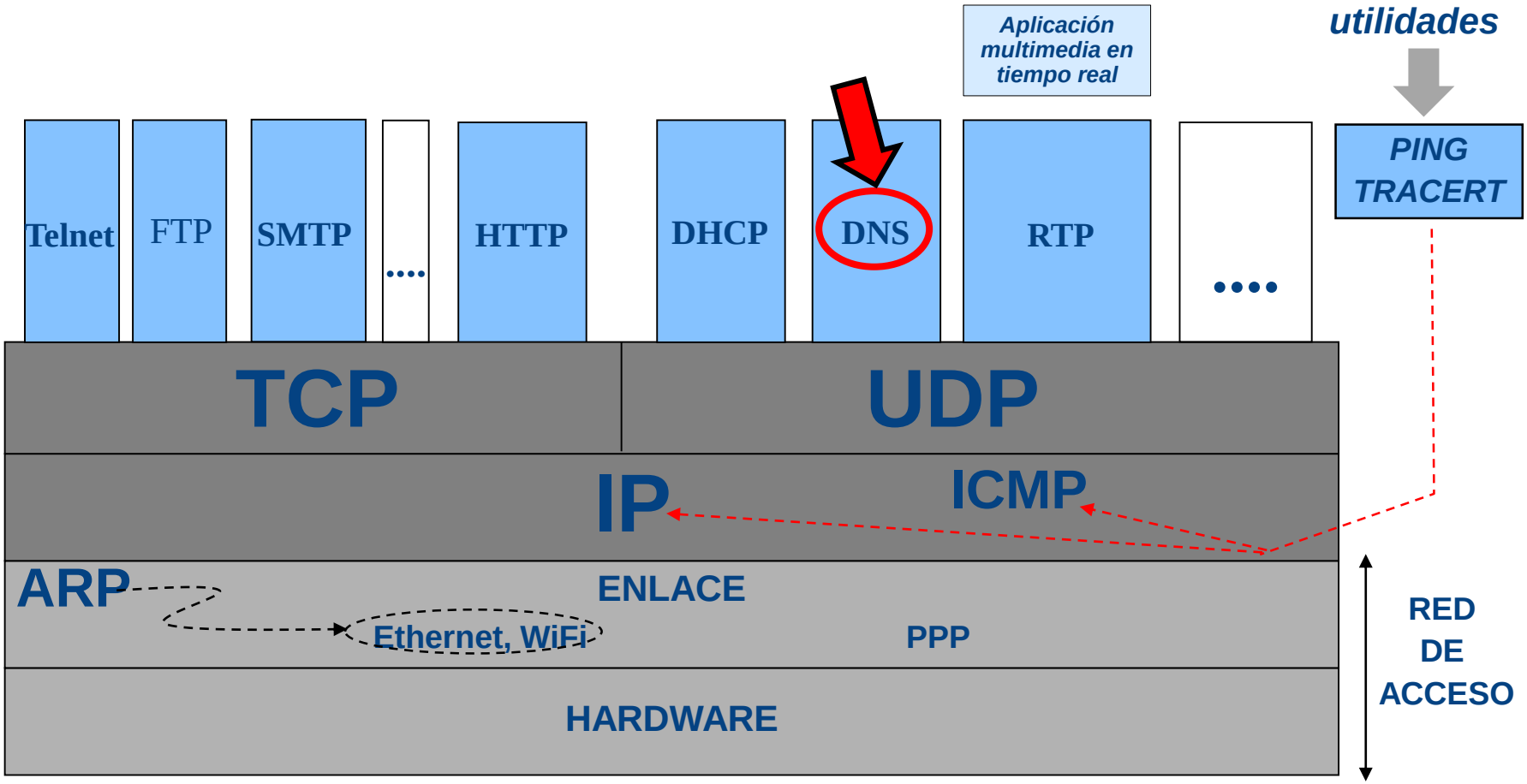


# Un Correo Electrónico Seguro vía SSL

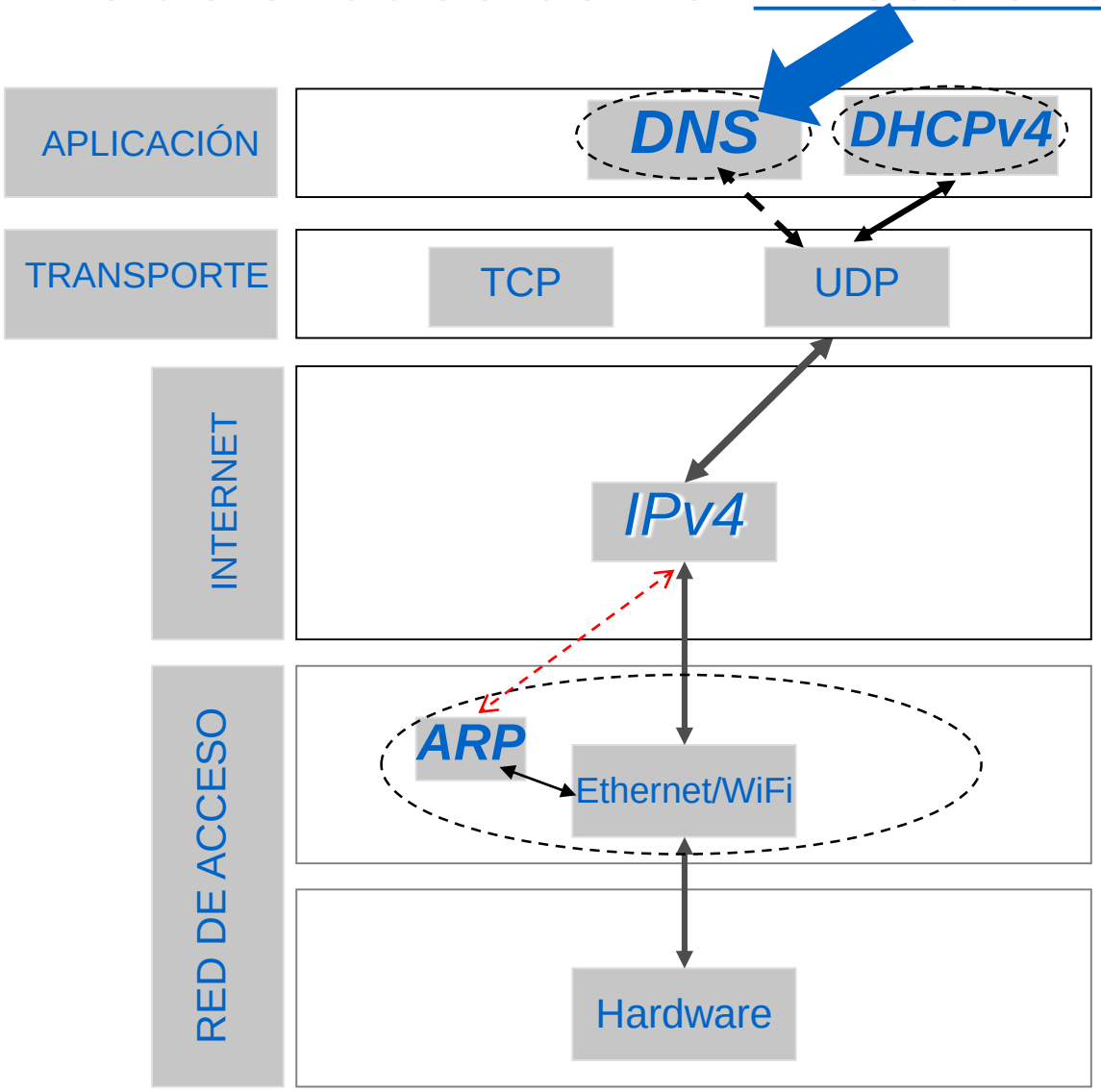
## Opción S/MIME (Extensiones seguras multipropósito al correo de Internet)



# Protocolos de Aplicación sobre UDP



# Protocolos y Niveles TCP/IP Relacionados con el Direccionamiento IP



# Sistema de Nombres de Dominio (DNS)
# 2 Componentes Principales

1. **BASE DE DATOS DNS en Internet**: *Una **BD Distribuida mediante servidores DNS de las diferentes organizaciones conectadas a Internet y que mantienen:***

   - *Registros locales con **las asociaciones conocidas LOCALMENTE entre los nombres simbólicos y las direcciones** numéricas de la organización correspondiente*
   - Ningún servidor DNS contiene la BD completa
   - Cada organización conectada a Internet suele disponer de su propio servidor DNS que gestiona su propia BD DNS

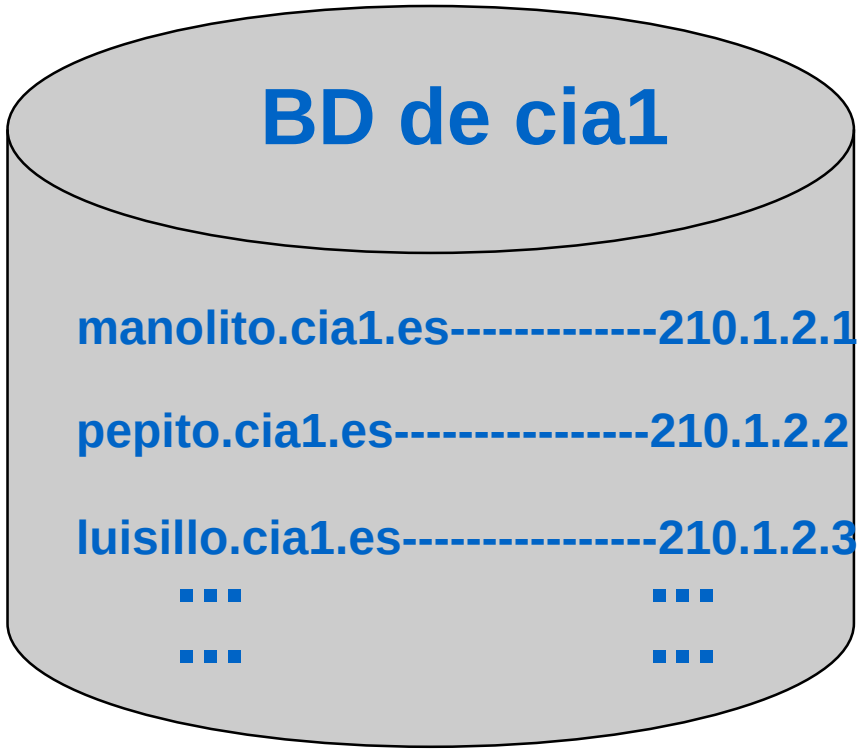
2. **PROTOCOLO DNS:** ***Protocolo del nivel de aplicación que sigue el modelo cliente y servidor para la RESOLUCIÓN DE NOMBRES SIMBÓLICOS en direcciones IP***

   - *Para acceder a un servidor DNS, se necesita un cliente DNS y un protocolo DNS*

# Un Ejemplo de una BD DNS Local gestionada por su propio Servidor DNS

## Servidor DNS

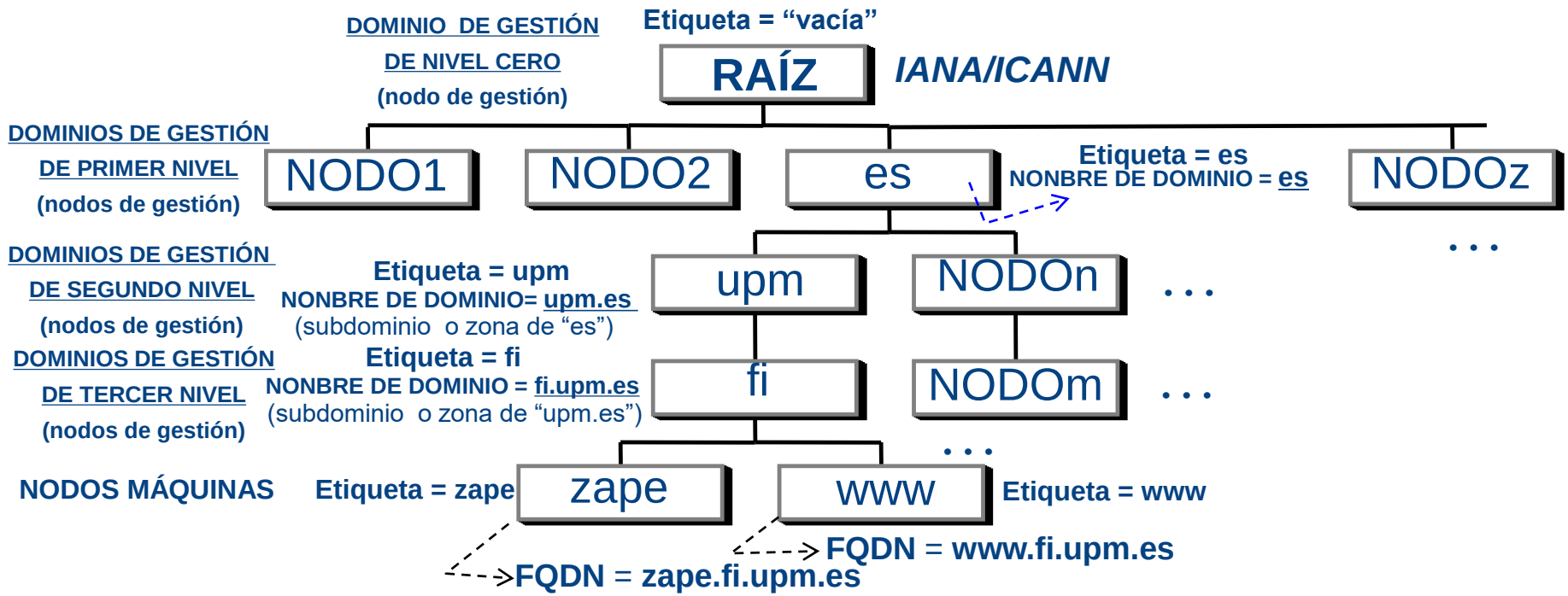
- *Cada organización conectada a Internet suele disponer de su propio servidor DNS que gestiona su BD DNS local*
- *Ningún servidor DNS contiene la BD completa*
- *BD DNS es una BD distribuida*




**BD DNS de Internet** = Internet { BD de **cia1** + BD de cia2+ … }

*Organizaciones conectadas a Internet con su propio Servidor DNS gestionando su propia BD DNS*

# La BD DNS en Internet se representa mediante una ESTRUCTURA JERÁRQUICA (ÁRBOL) DE DOMINIOS o NIVELES DE GESTIÓN




*DOMINIO DE GESTIÓN* = *ORGANIZACIONES QUE DISPONEN DE UN SERVER DNS QUE GESTIONA SU BD DNS LOCAL*

**ÁRBOL** = **Nodos de Gestión** (organizaciones con Servidor DNS local) + **Nodos Máquinas**

- **Cada NODO DE GESTIÓN** dispone de una **ETIQUETA** (nombre simbólico) y un **NOMBRE DE DOMINIO** **(secuencia de etiquetas)**

***SECUENCIA DE ETIQUETAS separadas por* puntos desde la ETIQUETA del propio NODO hacia arriba, es decir, hasta la RAÍZ *(acaba en punto que no se ve)***

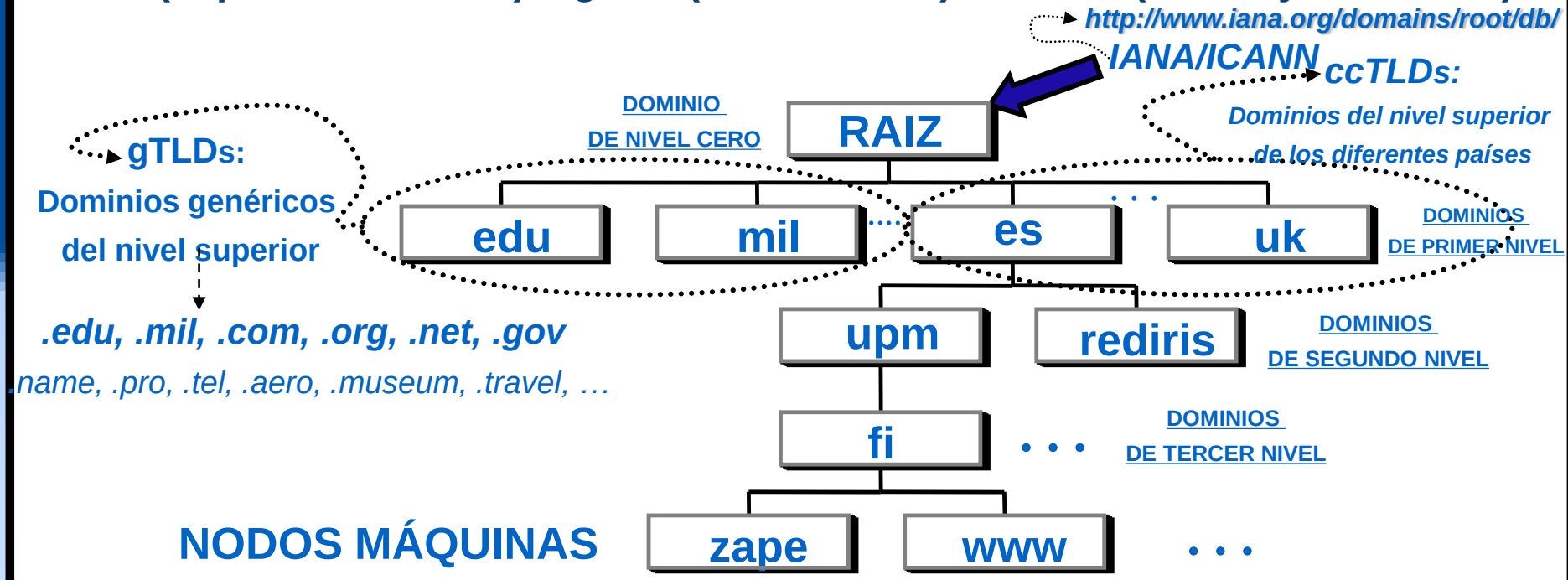
- **Cada NODO MÁQUINA disponde de una ETIQUETA** y un **FQDN** (*Fully Qualified Domain Name*)
  - ***FQDN: SECUENCIA DE ETIQUETAS separadas por puntos desde la ETIQUETA del propio NODO hacia arriba, es decir, hasta la RAÍZ (acaba en punto que no se ve)***

## EJEMPLO DE DOMINIOS DE GESTIÓN DE PRIMER NIVEL o TOP LEVEL DOMAINs (TLDs)

EL DOMINIO DE GESTIÓN DE PRIMER NIVEL, SE CORRESPONDE CON LOS DOMINIOS TOP LEVEL o TLDs distribuidos en:

# TLDs Genéricos + TLDs países

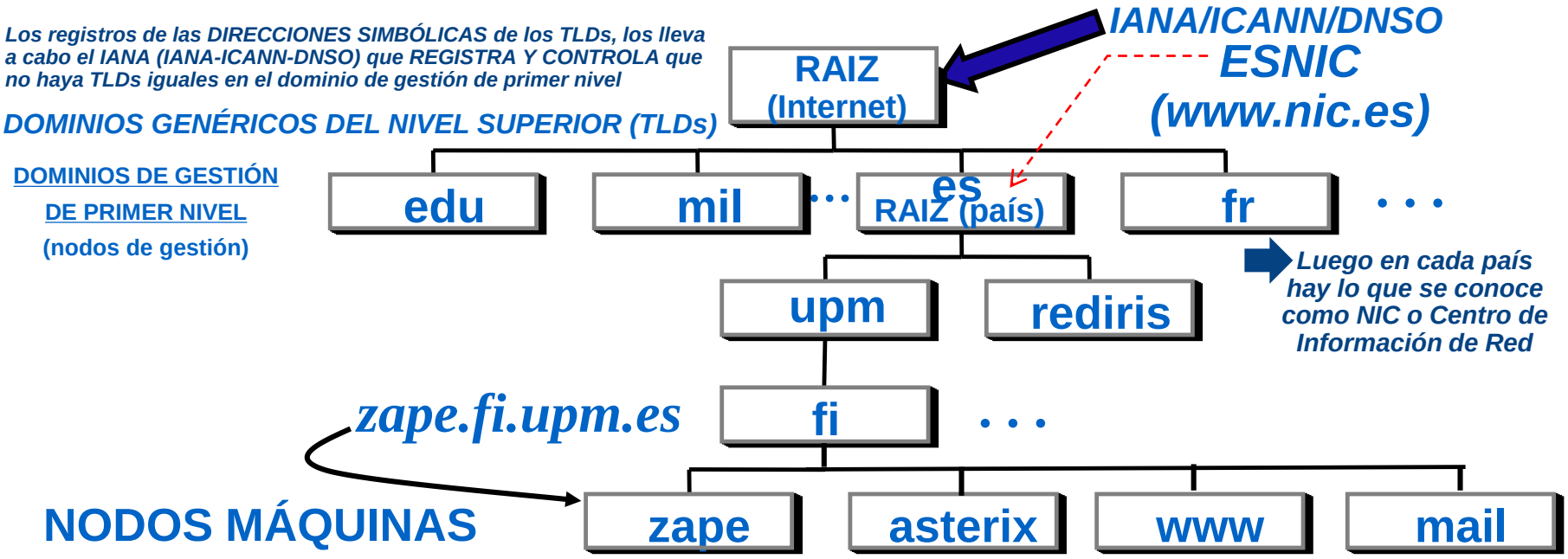
**TLDs (Top Level Domain) = gTLD (Generic TLD) + ccTLD (Country Code TLD)**



# REGISTROS DE DIRECCIONES SIMBÓLICAS

## IANA, NIC (Network Information Center), Agentes Registradores y Nuevos Dominios

*Los registros de las DIRECCIONES SIMBÓLICAS de los TLDs, los lleva a cabo el IANA (IANA-ICANN-DNSO) que REGISTRA Y CONTROLA que no haya TLDs iguales en el dominio de gestión de primer nivel*




- **Agentes Registradores acreditados por ESNIC/IANA-ICANN:**
  - *Registros de nombres simbólicos de primer nivel (x.es, x.com, x.org,…) y segundo nivel (x.com.es, x.org.es, …)*
- Asimismo, existen dominios particulares TLD (aprobados por el IANA) con un coste superior:
  - *p. ej.: .madrid, .empresa,..., .apellido*

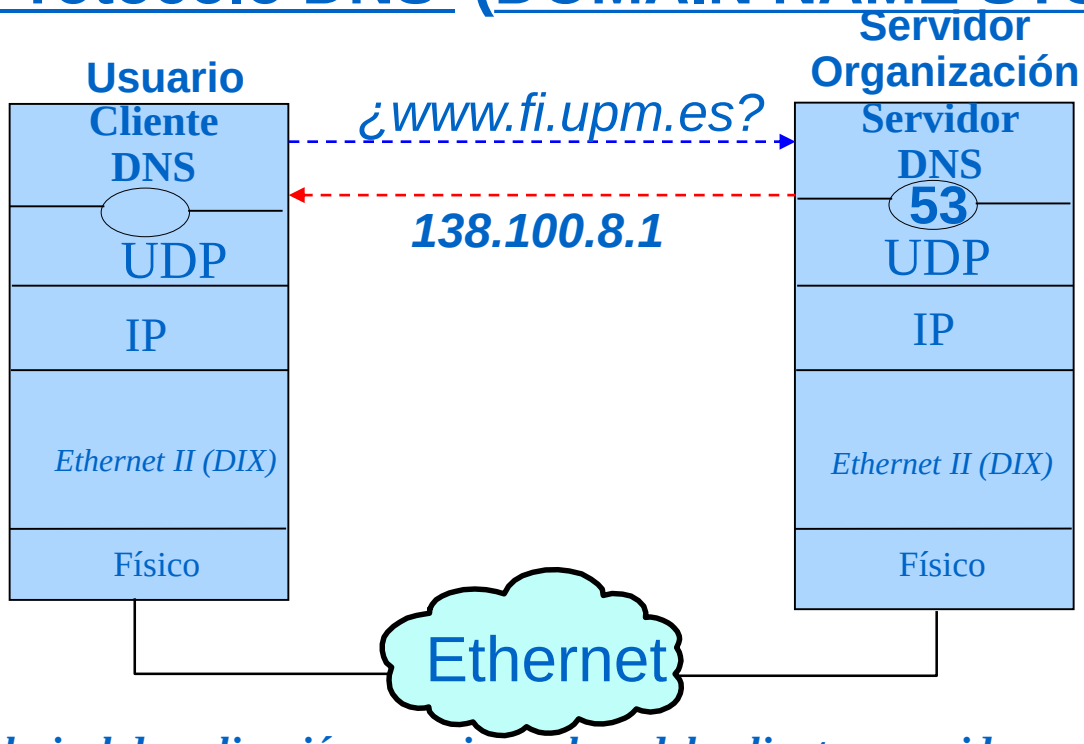
# El Sistema DNS
componentes

# El Protocolo DNS

***Protocolo soporte de otros protocolos o aplicaciones***

## Resolución de Nombres de Dominio

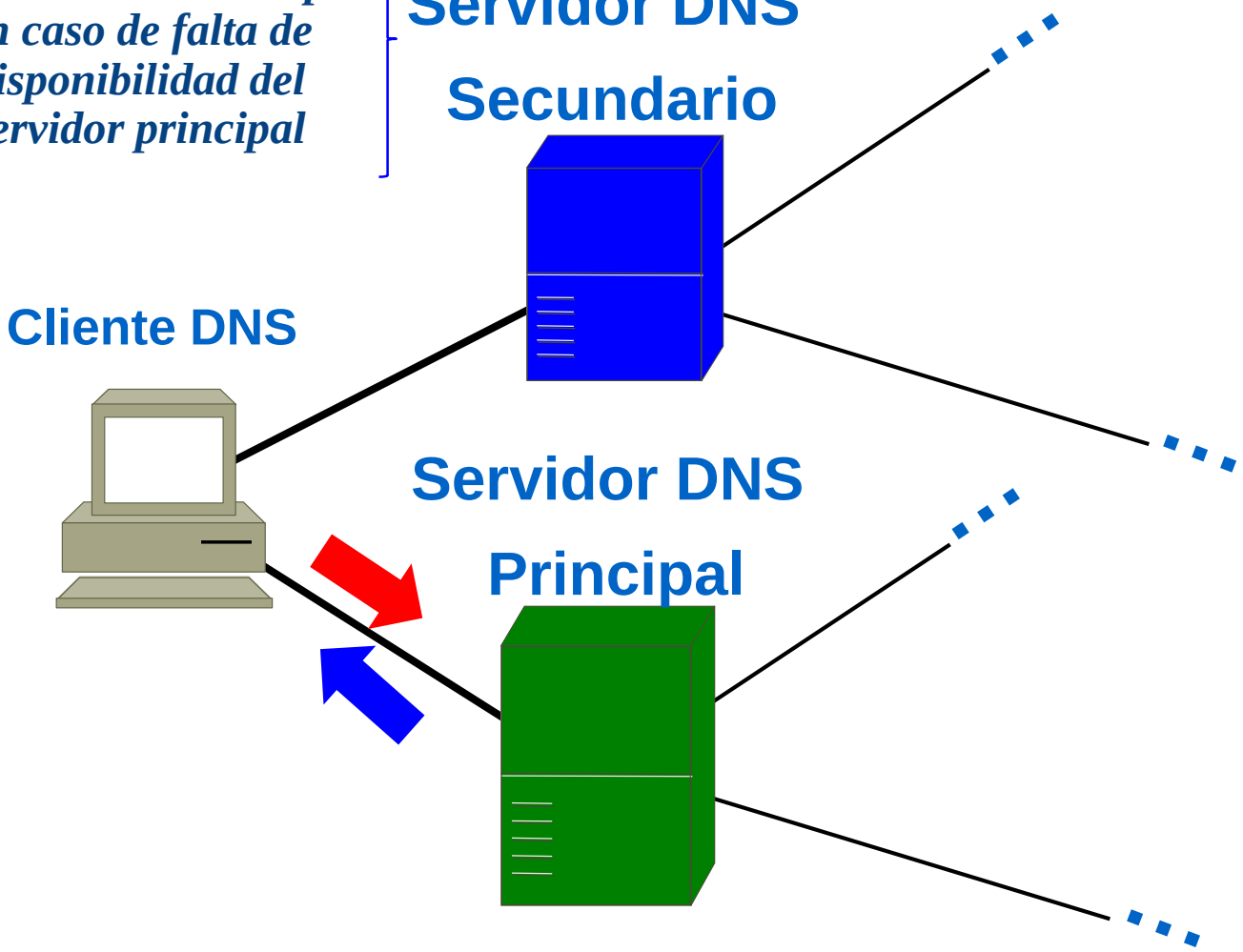
# El Protocolo DNS (DOMAIN NAME SYSTEM)



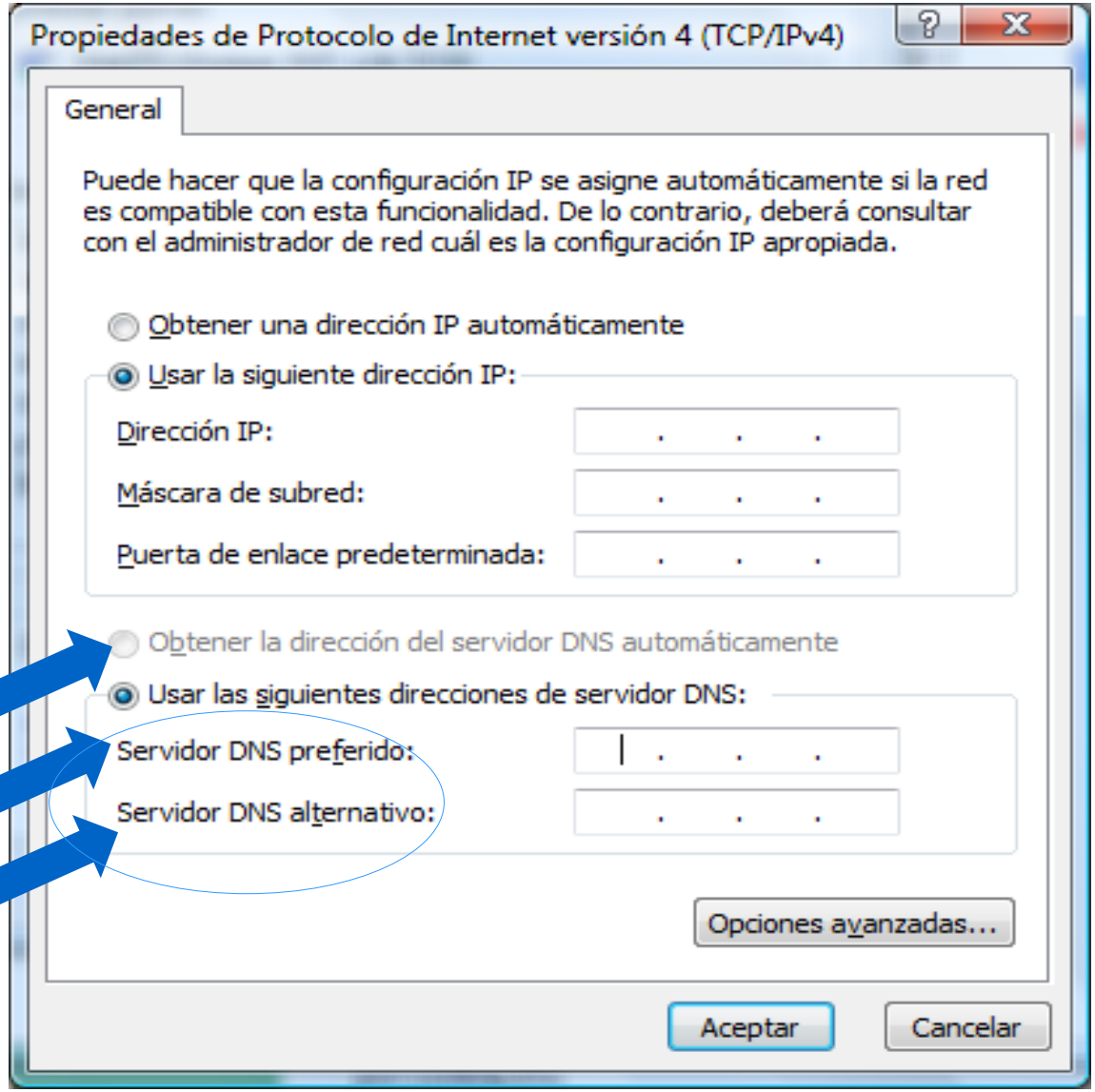

- ***Protocolo del nivel de aplicación que sigue el modelo cliente y servidor para la RESOLUCIÓN DE NOMBRES DE DOMINIO en direcciones numéricas***
  - ✓ Los **programas cliente y servidor** se pueden dividir en **dos categorías**
    - Protocolos que pueden ser **usados directamente por el usuario**: *HTTP*, *SMTP*, *etc.*
    - **Protocolos como DNS que, a su vez,** *da soporte, a otros protocolos* **o aplicaciones como *HTTP* y *SMTP* entre otros**
  - **Un cliente DNS comienza resolviendo un Nombre de Dominio**, interrogando a su servidor DNS
  - *Si un servidor DNS no tiene la RESOLUCIÓN SIMBÓLICA-NUMÉRICA solicitada, se convierte en cliente de otro servidor DNS* en la JERARQUÍA DNS establecida previamente en Internet

# Dos Tipos de Servidores DNS

*Puede convertirse en un Servidor DNS Principal en caso de falta de disponibilidad del servidor principal*



# Configuración de los Servidores DNS



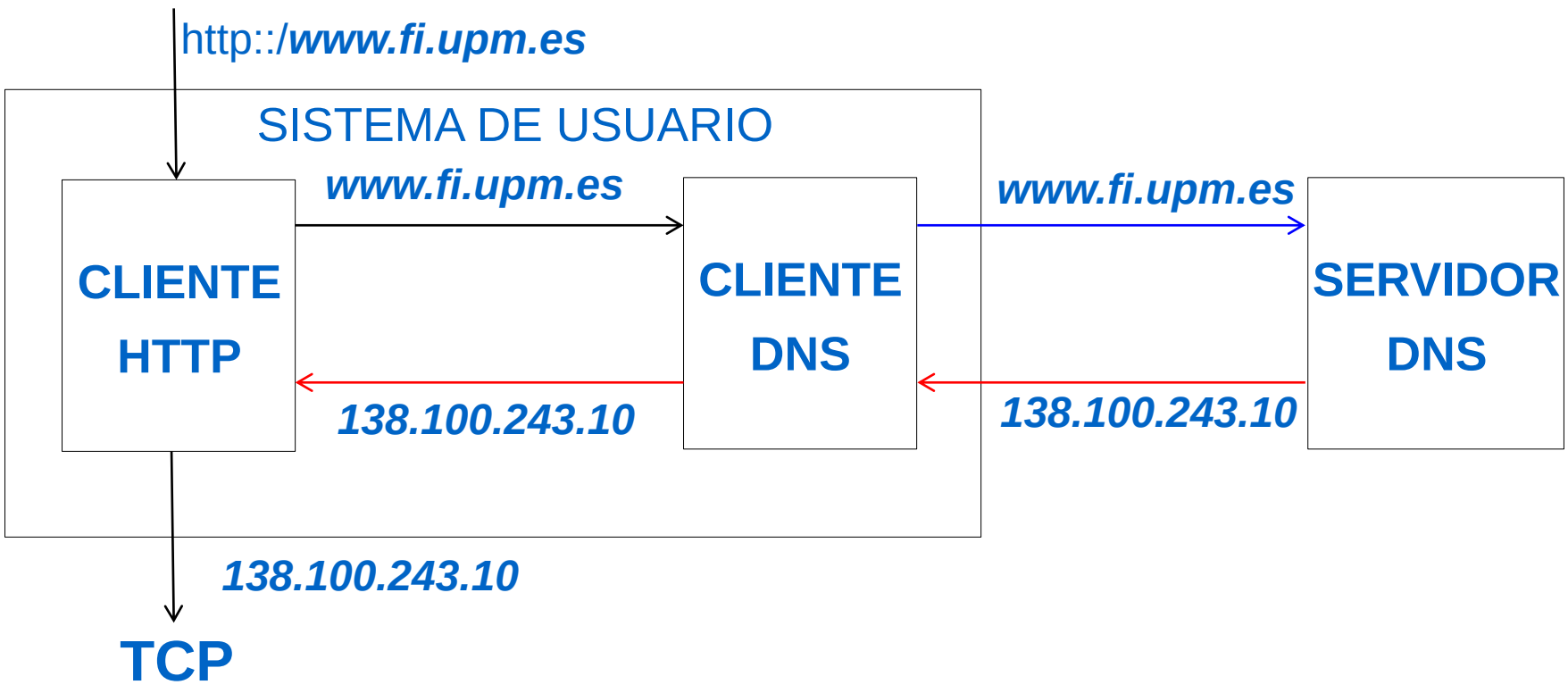

**Por DHCP**

**"A mano"**

# Ejemplo de Uso del Servicio DNS Solicitado Previamente por HTTP

*Protocolo DNS da soporte, a otros protocolos* ***o aplicaciones como HTTP y SMTP entre otros***

***La resolución de nombres se hace de forma transparente por las aplicaciones del cliente***

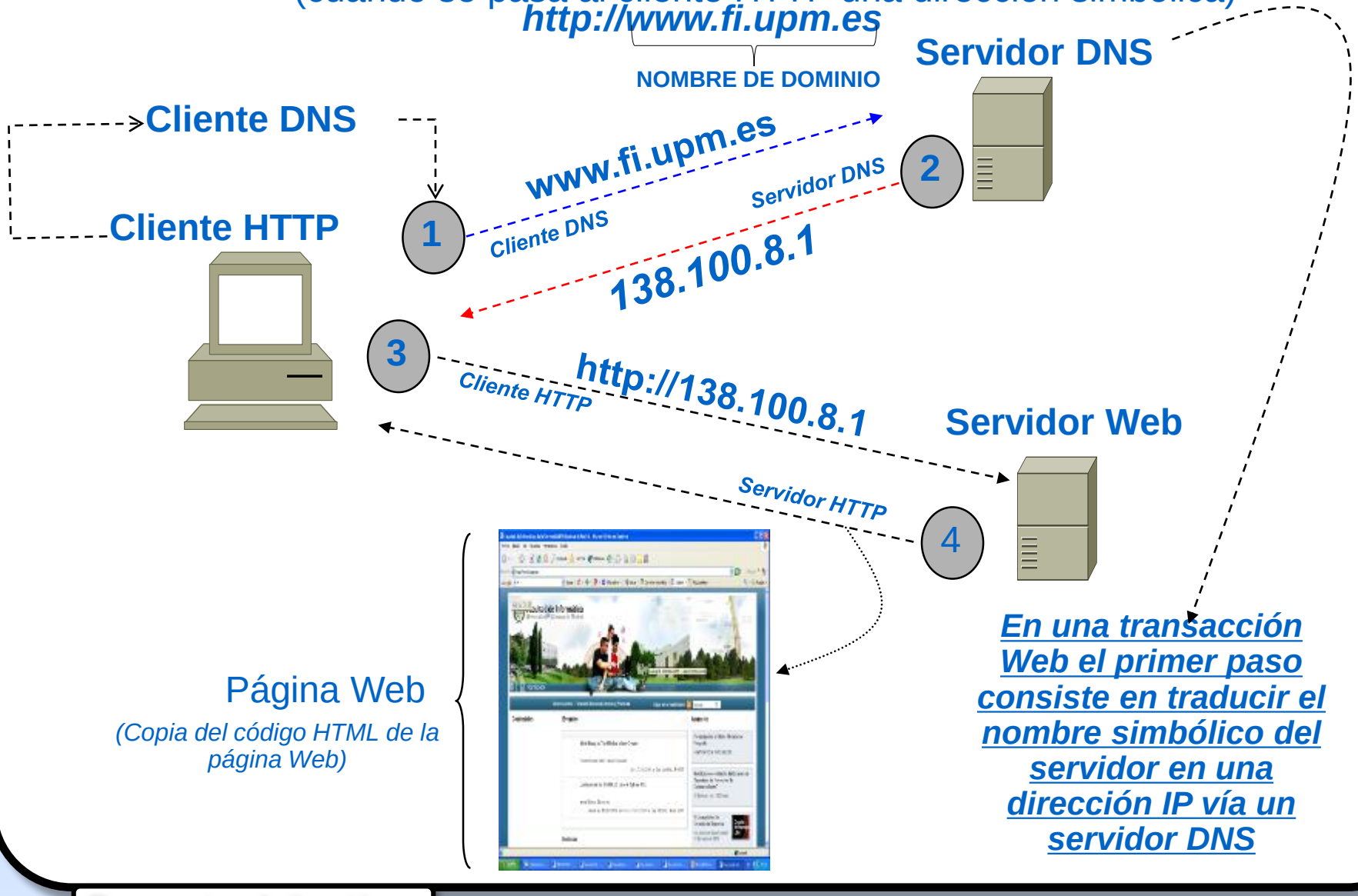


# Un Ejemplo de Resultado Final previa Consulta al Sistema DNS

(cuando se pasa al cliente HTTP una dirección simbólica)

***http://www.fi.upm.es***

NOMBRE DE DOMINIO



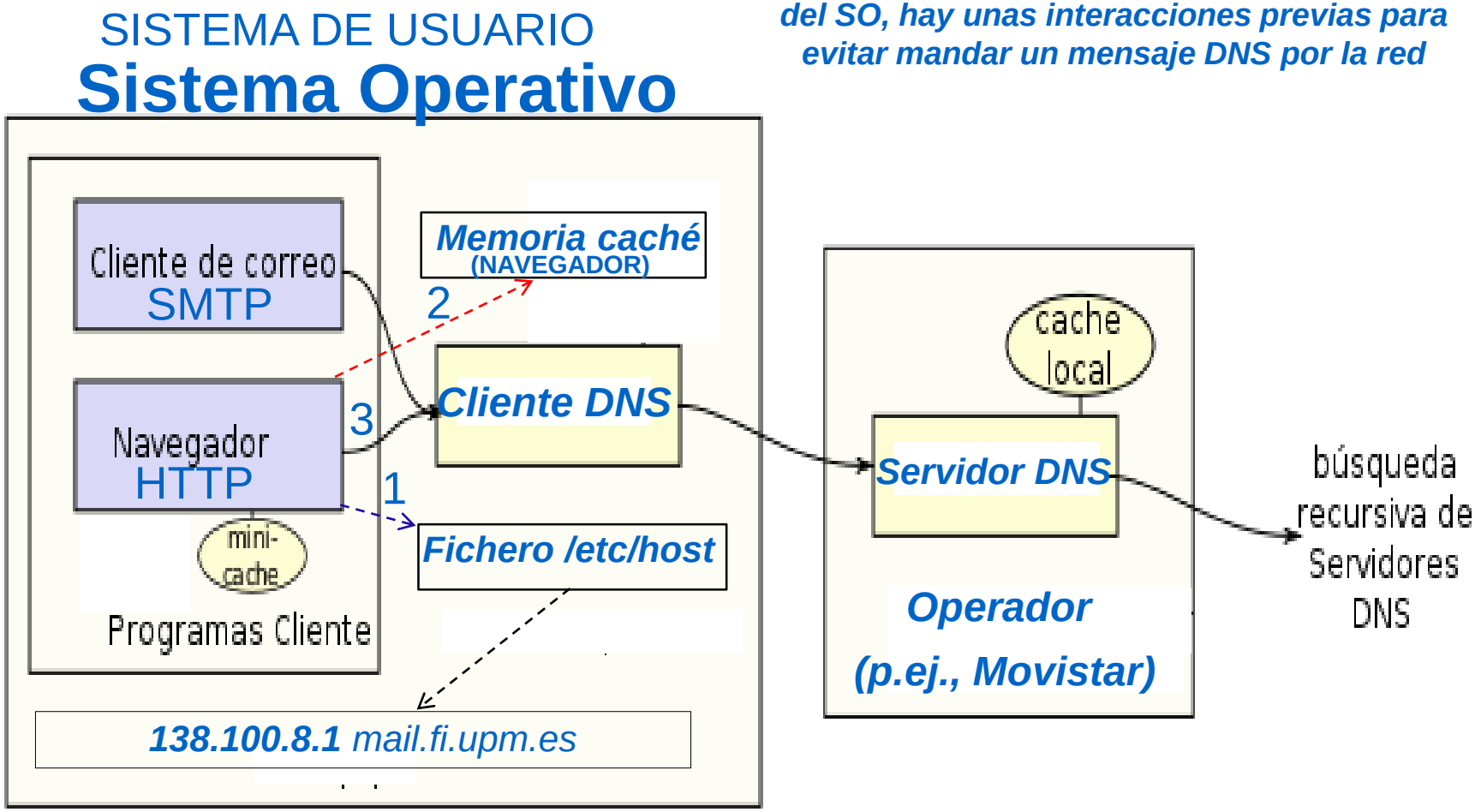

Página Web

*(Copia del código HTML de la página Web)*

***En una transacción Web el primer paso consiste en traducir el nombre simbólico del servidor en una dirección IP vía un servidor DNS***

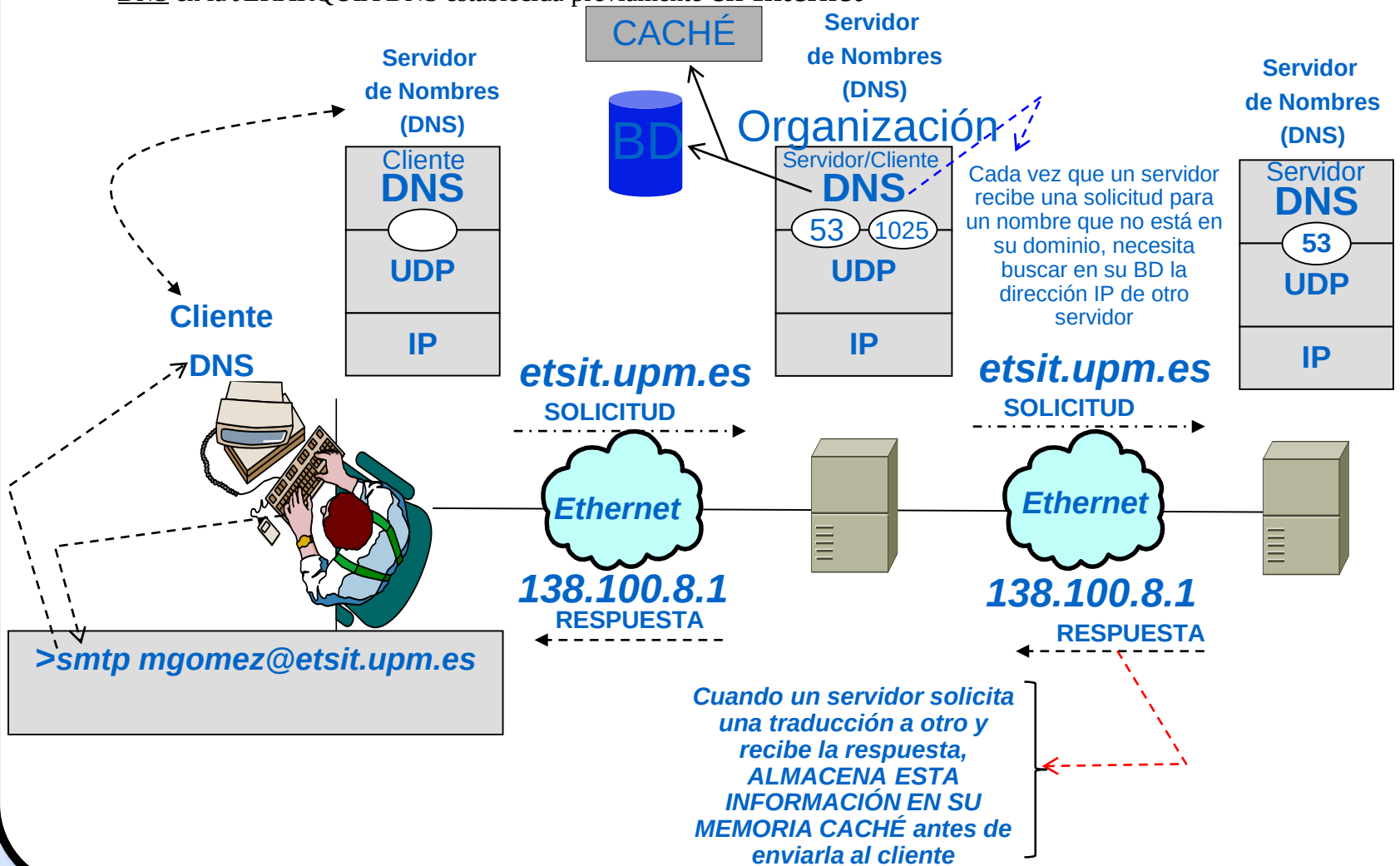
# Para Evitar un Excesivo Tráfico por la Red, existen unas Interacciones Previas al Servicio DNS

*Antes de lanzar el cliente DNS y dependiendo del SO, hay unas interacciones previas para evitar mandar un mensaje DNS por la red*



# Un Ejemplo de Resultado Final previa Consulta al Sistema DNS

- **Un cliente DNS comienza resolviendo un Nombre de Dominio**, interrogando a su servidor DNS
- *Si un servidor DNS no tiene la RESOLUCIÓN SIMBÓLICA-NUMÉRICA solicitada, se convierte en cliente de otro servidor DNS* en la JERARQUÍA DNS establecida previamente en Internet



# El Sistema DNS

## Tipos de Traducciones

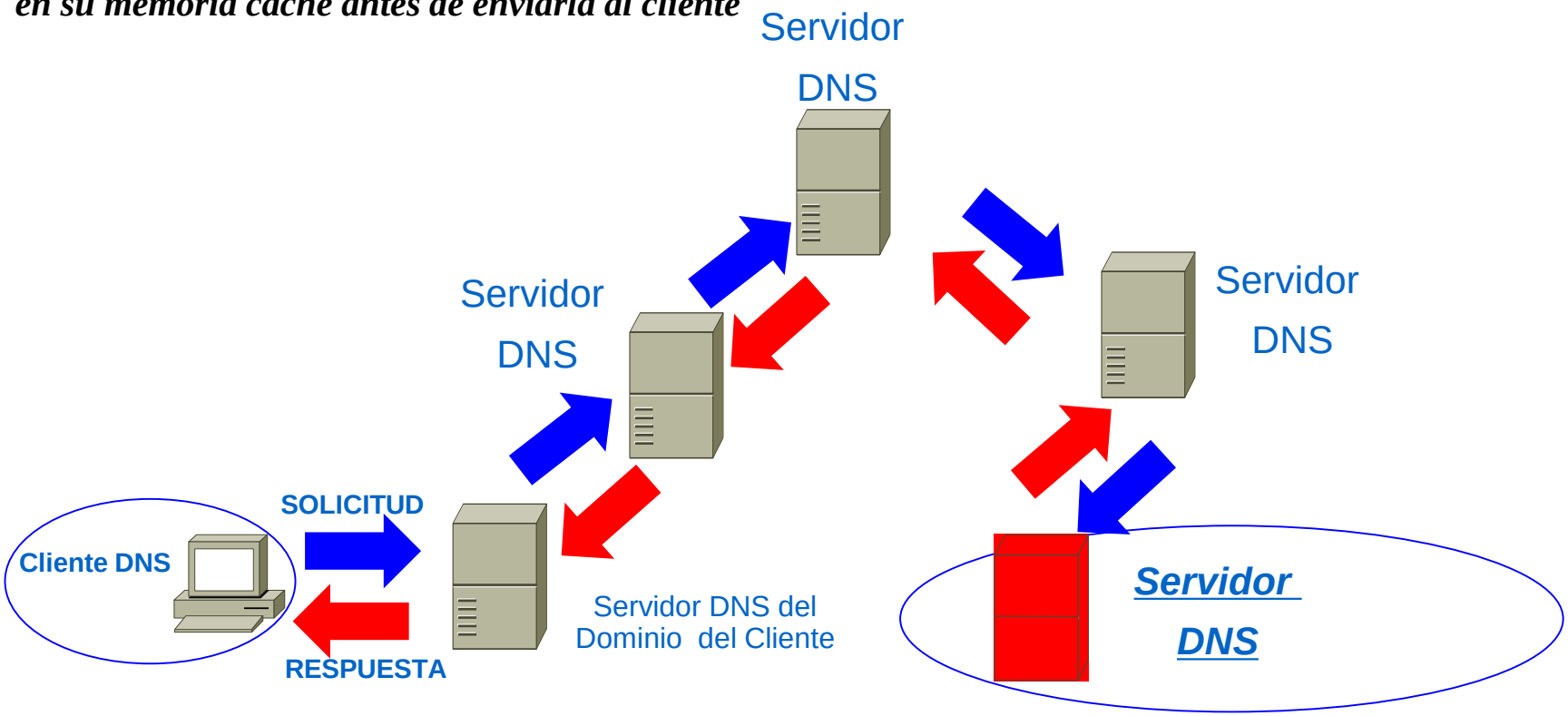
- **Resoluciones**

# Dos Tipos de Traducciones o Resoluciones DNS

1. Recursiva
   - *Generalmente a petición del cliente*
2. Iterativa
   - *Generalmente a petición del servidor*

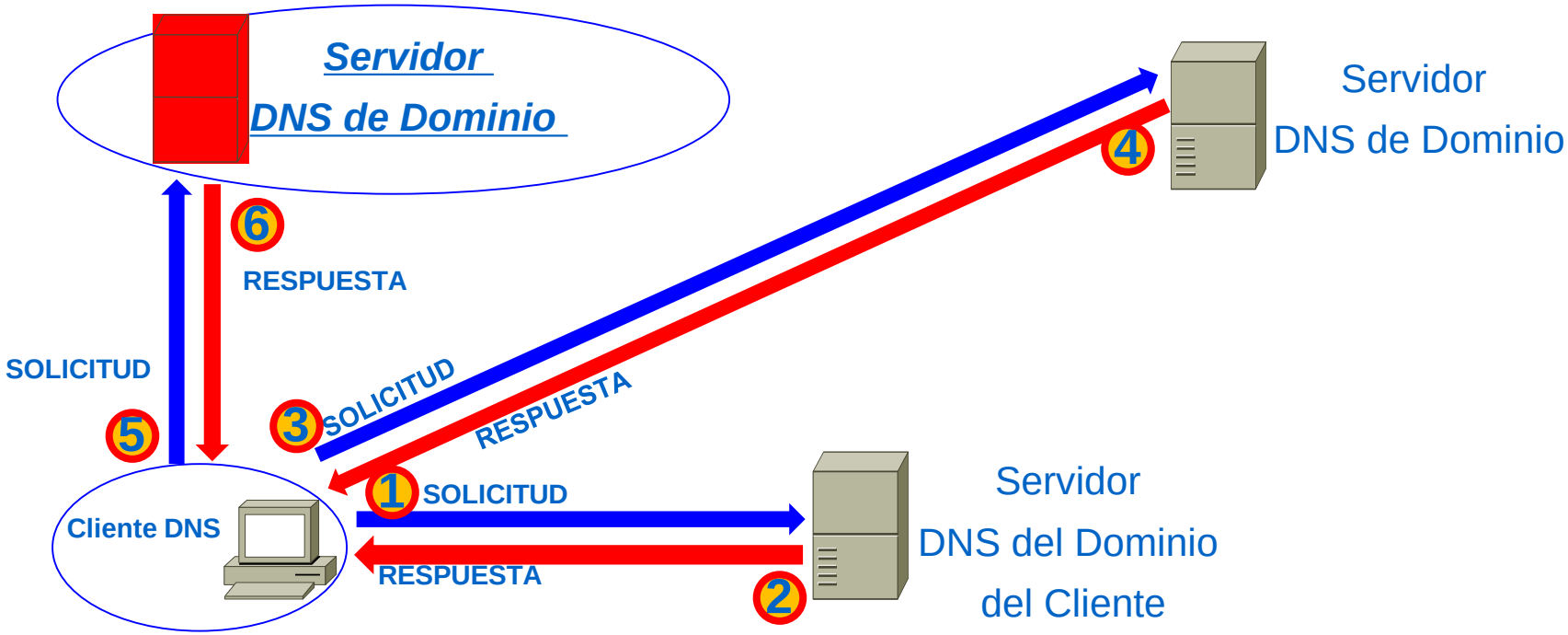
# Traducciones DNS Recursivas

- **Un cliente DNS comienza resolviendo un Nombre de Dominio**, interrogando a su servidor DNS
- *Si un servidor DNS no tiene la **RESOLUCIÓN SIMBÓLICA-NUMÉRICA** solicitada, se convierte en cliente de otro servidor DNS* en la JERARQUÍA DNS establecida previamente en Internet
- ***Cuando un servidor solicita una traducción a otro y recibe la respuesta, almacena esta información en su memoria caché antes de enviarla al cliente***



# Traducciones DNS Iterativas

**La traducción puede hacerse de forma iterativa a PETICIÓN DEL SERVIDOR que devuelve al cliente la dirección IP del servidor que cree que puede resolver el nombre**



# El Sistema DNS

## Registros DNS

# Contenido de un Registro DNS: 5 ATRIBUTOS en ASCII

| NOMBRE DE DOMINIO (FQDN) | TTL | CLASE | TIPO | DATOS |
|---|---|---|---|---|
| mail.fi.upm.es | 84600 (24h) | IN | A | 138.100.8.1 |

- **FQDN**: Clave primaria de búsqueda = NOMBRE SIMBÓLICO del equipo + NOMBRE DE DOMINIO de la organización de dicho equipo
- **TTL**: Tiempo de Vida del Registro en segundos
- **CLASE**: La clase puede ser **IN (relacionada con los protocolos de Internet)**, o **CH** (para un sistema no relacionado con Internet)
- **TIPO**: Tipo de recurso descrito por el registro
- **DATOS**: **Datos relacionados con el registro**. Aquí se encuentra la información esperada según el tipo de registro. Puede ser un número, un FQDN o una cadena ASCII

# Contenido del Campo Tipo: A

| NOMBRE DE DOMINIO (FQDN) | TTL | CLASE | TIPO | DATOS |
|---|---|---|---|---|
| zape.fi.upm.es | 3600 (1h) | IN | A | 138.100.8.1 |
| mail.fi.upm.es | 3600 | IN | A | 138.100.243.11 |
| www.fi.upm.es | 86400 (24h) | IN | A | 138.100.243.10 |

- **TIPO**: Tipo de recurso descrito por el registro
  - **A**: ***Registro que hace corresponder el FQDN con la dirección IP***

# Contenido de un Registro DNS: PTR

| NOMBRE DE DOMINIO | TTL | TIPO | CLASE | DATOS |
|---|---|---|---|---|
| *1.8.100.138.in-addr.arpa* | 3600 | IN | **PTR** | **zape.fi.upm.es** |

- **TIPO**: Tipo de recurso descrito por el registro
  - **PTR**: **Puntero a un FQDN**. *Permite realizar búsquedas inversas y* ***obtener un FQDN a partir de una dirección IP***
    - *P.ej., nslookup 138.100.8.1*

# Contenido del Campo Tipo: CNAME

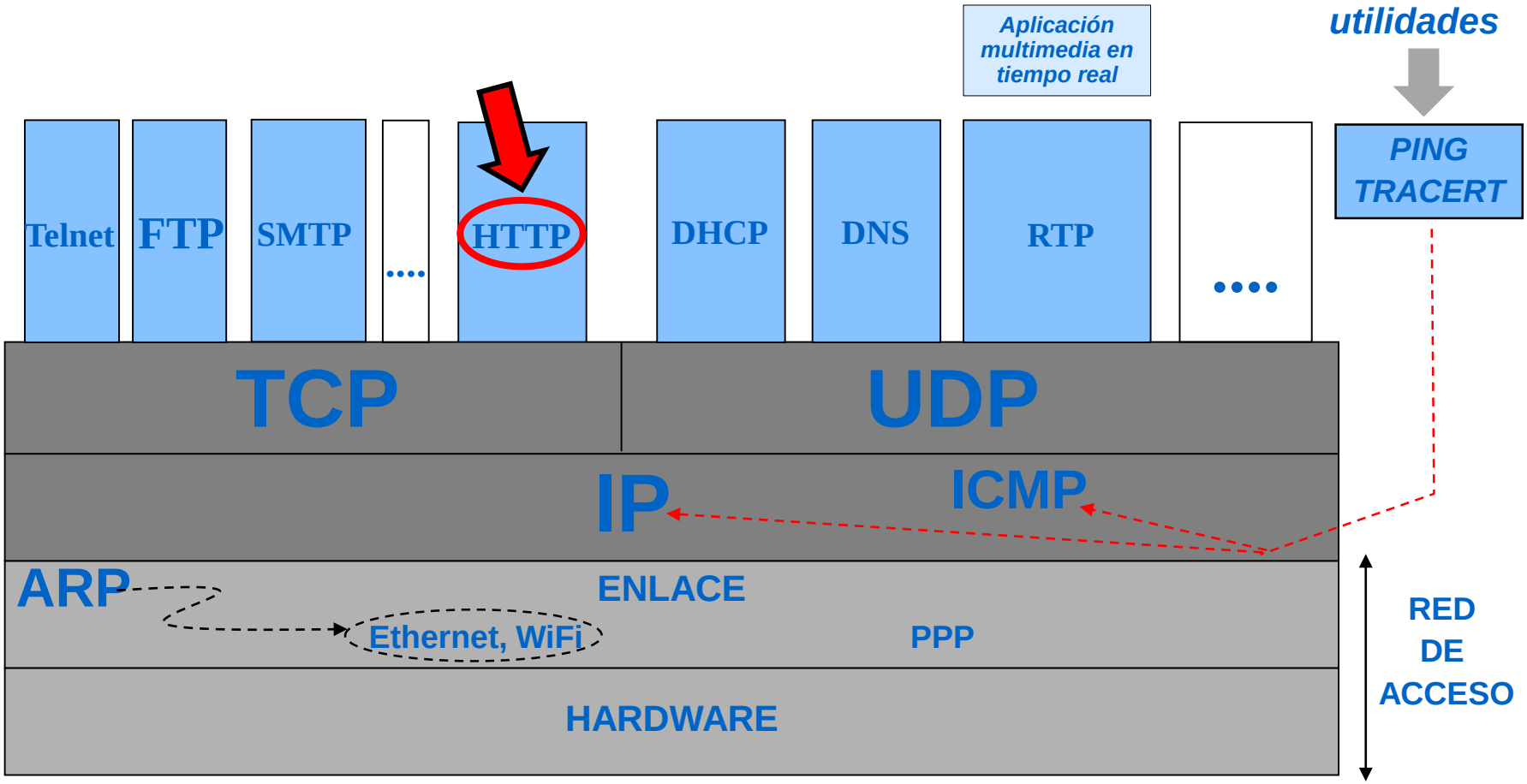
| NOMBRE DE DOMINIO | TTL | CLASE | TIPO | DATOS |
|---|---|---|---|---|
| fi.upm.es (alias) | 86400 | IN | **CNAME** | www.fi.upm.es |
| www.fi.upm.es (FQDN) | 86400 | IN | A | 138.100.243.10 |

# Contenido del Campo Tipo: MX

| NOMBRE DE DOMINIO (FQDN) | TTL | CLASE | TIPO | DATOS |
|---|---|---|---|---|
| mail.fi.upm.es | 3600 | IN | **MX** | 1 mx.fi.upm.es |
| mail.fi.upm.es | 3600 | IN | **MX** | 2 mx1.fi.upm.es |
| mx.fi.upm.es | 3600 | IN | A | 138.100.0.1 |
| mx1.fi.upm.es | 3600 | IN | A | 138.100.0.2 |

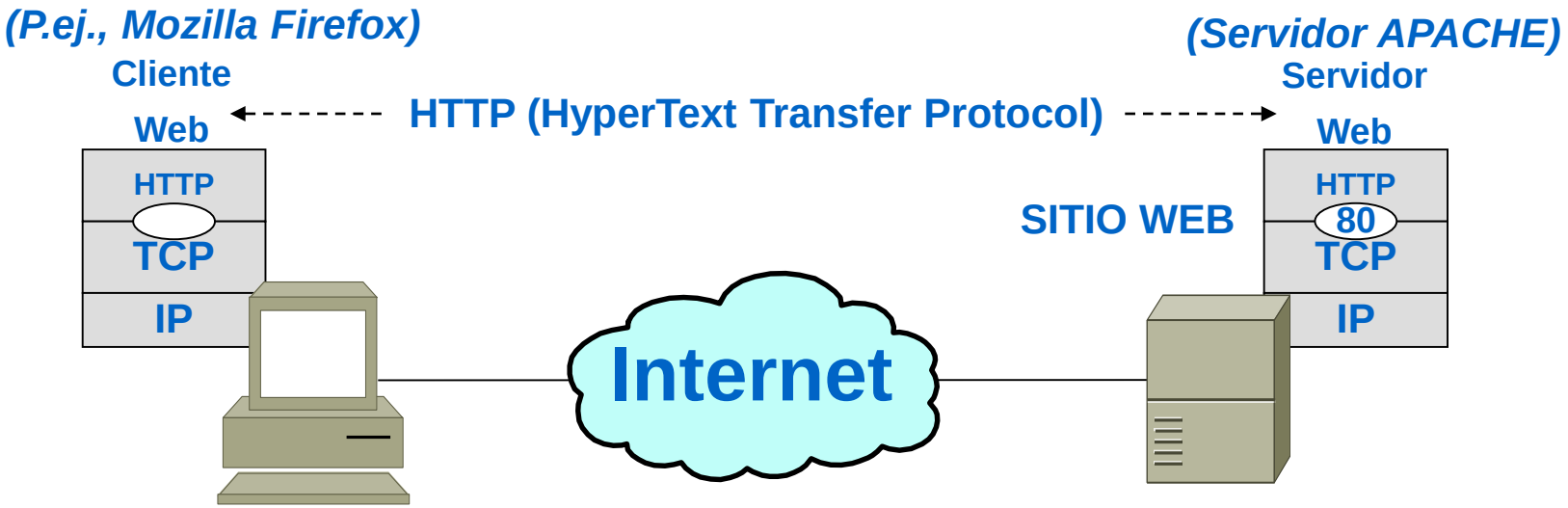
- **TIPO**: Tipo de recurso descrito por el registro
  - **MX** (*Mail eXchange*): **Registro del Servidor de Intercambio de Correo**. Cuando un usuario envía un correo electrónico a una dirección (user@domain), el servidor de correo saliente interroga al servidor de nombre de dominio con autoridad sobre el dominio para obtener el registro MX. **Pueden existir varios registros MX** por dominio, para así suministrar una repetición en caso de fallos en el servidor principal de correo electrónico. De este modo, *el registro MX permite definir una prioridad con un valor entre 0 y 65.535*

# Protocolos de Aplicación sobre TCP



# HTTP (HyperText Transfer Protocol)

## *PROTOCOLO de PRESENTACIÓN Y DISTRIBUCIÓN DE INFORMACIÓN EN INTERNET*




- ***Para descargar desde un cliente HTTP o navegador (Mozilla Firefox, Safari, Google Chrome, Internet Explorer, etc.) cualquier tipo de fichero mantenido por un servidor HTTP o Servidor Web (Apache y Nginx) e incluso para acceder a cualquier tipo de servicio (p.ej., Webmail o correo Web o cliente/servidor SMTP remoto vía HTTPs)***

# APACHE
## *Típico Servidor Web en Internet (85,5%)*

- *Implementación LIBRE de un Servidor Web/HTTP (protocolo HTTP 1.1) de CÓDIGO ABIERTO MULTIPLATAFORMA PARA CUALQUIER KERNEL*
  - *Unix (BSD, GNU/Linux, etc.), Microsoft Windows, OS X y otras, que implementa el protocolo HTTP/1.1*
  - El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) *de la Apache Software Foundation*
  - *Configurable fácilmente para sus distintas funcionalidades (p.ej., HTTPS)*
  - Número de puerto por omisión = 80
    - Reservado para el administrador de la máquina
    - Cualquier otro usuario, con cuenta en dicha máquina, que quiera disponer de su propio Servidor Web, debe arrancar dicho servidor en un número de puerto diferente al 80
      - Generalmente, *el 8080*, aunque se puede poner cualquier otro número no reservado como el *3123*, *etc.*
        - *Debe ser conocido previamente*

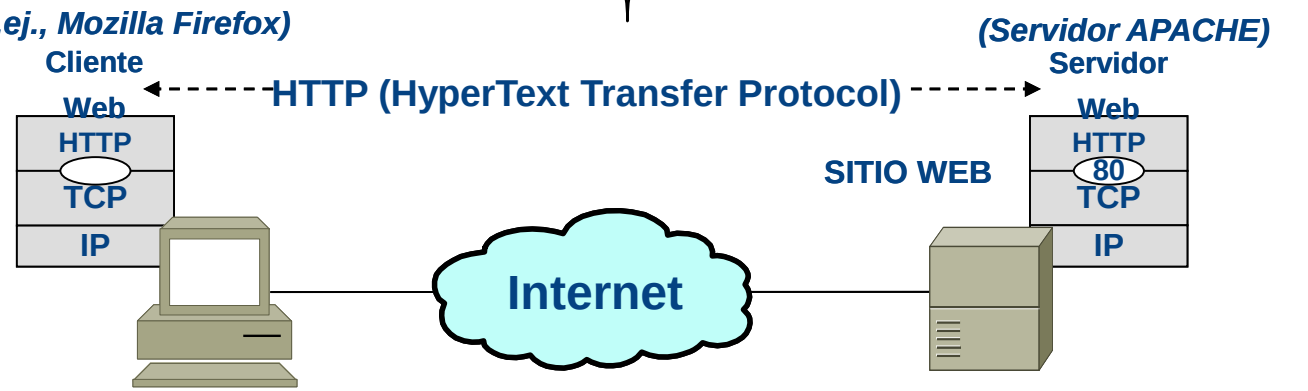
# Ginx (gineX)

## *Segundo Servidor Web en Internet (14,5 %)*

- Alternativa a Apache
- *Implementación LIBRE de un Servidor Web/HTTP (protocolo HTTP 1.1) de CÓDIGO ABIERTO MULTIPLATAFORMA PARA CUALQUIER KERNEL*
- Igor Sysoev creador del nginx Web server y fundador de NGINX, Inc.

# Inicio de los Contenidos de un Servidor Web

***PROTOCOLO de PRESENTACIÓN Y DISTRIBUCIÓN DE INFORMACIÓN EN INTERNET para descargar desde un cliente HTTP o navegador (Mozilla Firefox, Safari, Google Chrome, Internet Explorer, etc.) cualquier tipo de fichero mantenido por un servidor HTTP o Servidor Web (configuración del código de libre distribución Apache o Nginx)***




- ***LA DESCARGA DESDE EL CLIENTE DE LOS CONTENIDOS DE UN SERVIDOR WEB COMIENZA POR EL TÍPICO Fichero HTML (HyperText Markup Language) de la página Web inicial para su interpretación y visualización por un intérprete HTML en el cliente HTTP o NAVEGADOR***

  - ✓***index.html o default.html el cual dispone de texto e hiperenlaces (enlaces HTTP) LOCALES o REMOTOS a:***

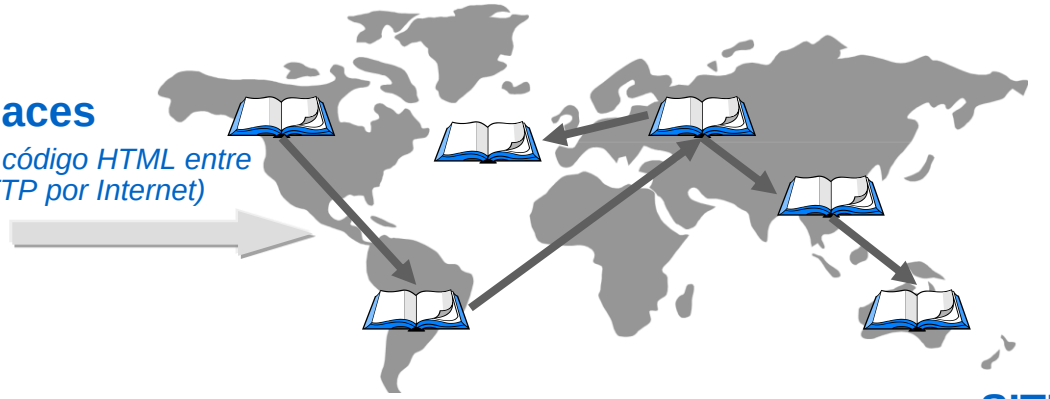
    - ***Ficheros LOCALES o REMOTOS de texto (pdf, txt, doc, …), audio (AAC o .m4a, mp3, WMA, AC-3, WAV, AIFF,…), vídeo (MPEG-4 o m4v, H.264, …), imagen (.jpg, .png, .gif, …), etc.***
    - ***Accesos a cualquier tipo de servicio (p.ej., Webmail: Google, Gmail, Yahoo Mail, Outlook.com, etc.)***
      - ***Webmail o correo Web o cliente/servidor SMTP remoto vía HTTPs/SSL)***

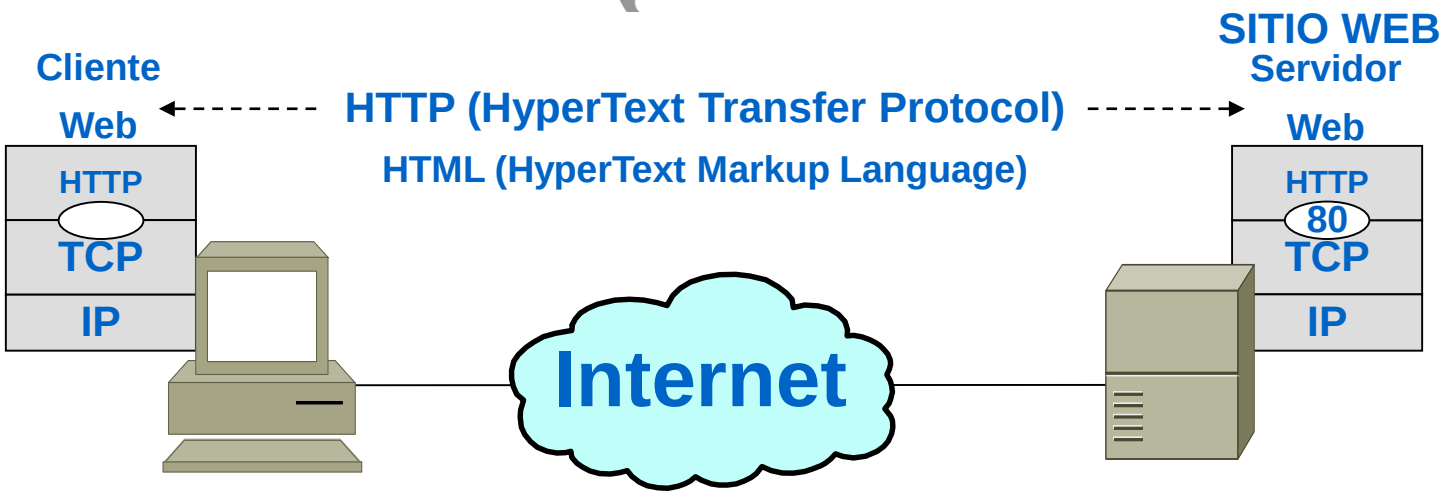
# HTTP (HyperText Transfer Protocol)

**El protocolo HTTP PERMITE, POR EXTENSIÓN, UN SERVICIO DISTRIBUIDO *(WORLD WIDE WEB = WWW = WEB = TELARAÑA)* DE INFORMACIÓN MEDIANTE UN SISTEMA DISTRIBUIDO DE SERVIDORES WEB ENLAZADOS a través de HIPERENLACES (enlaces HTTP)**

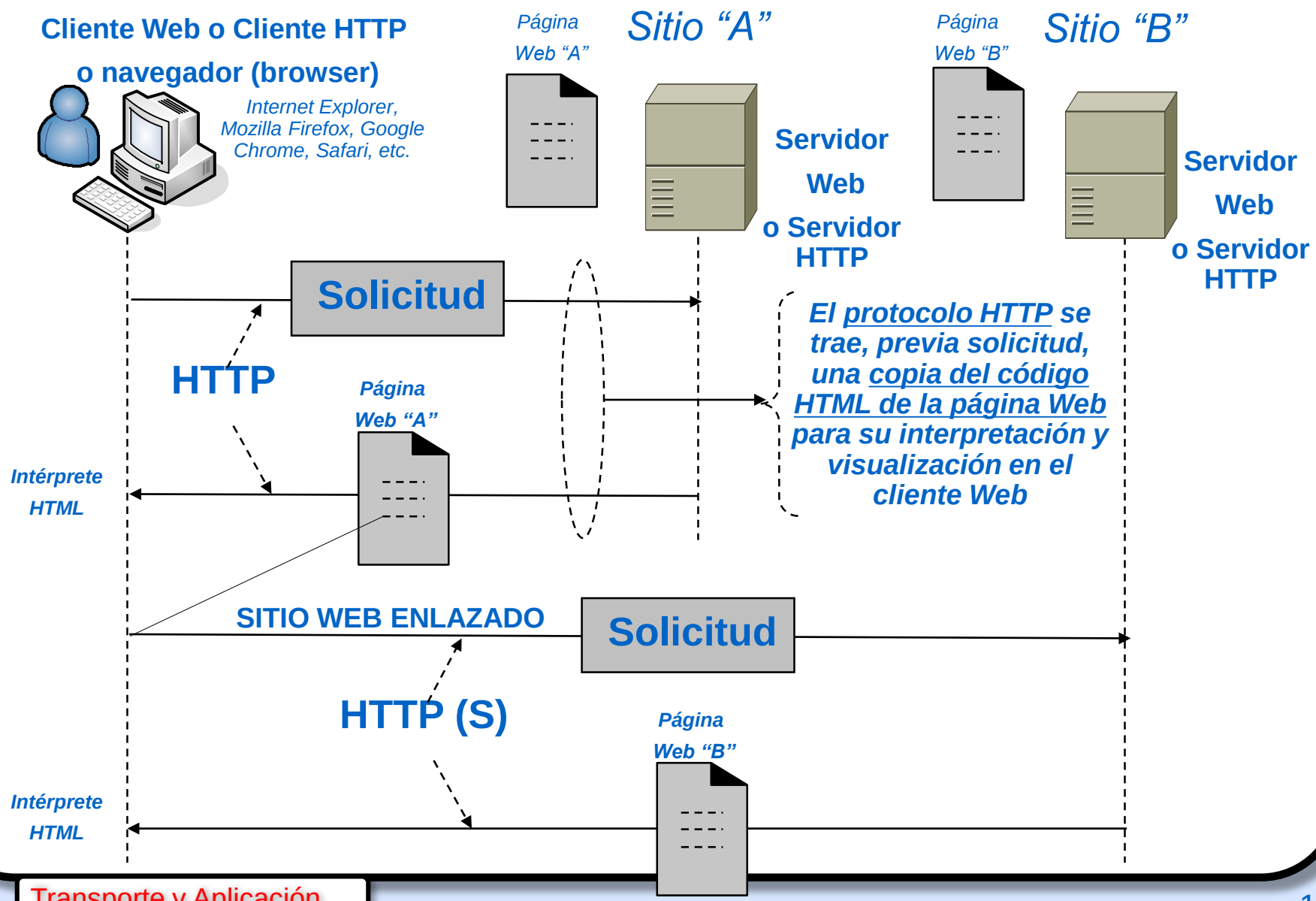
**Hiperenlaces**

*(enlaces HTTP incluidos en código HTML entre diferentes servidores HTTP por Internet)*





# Ejemplo de Servidores Web enlazados vía Hiperenlaces



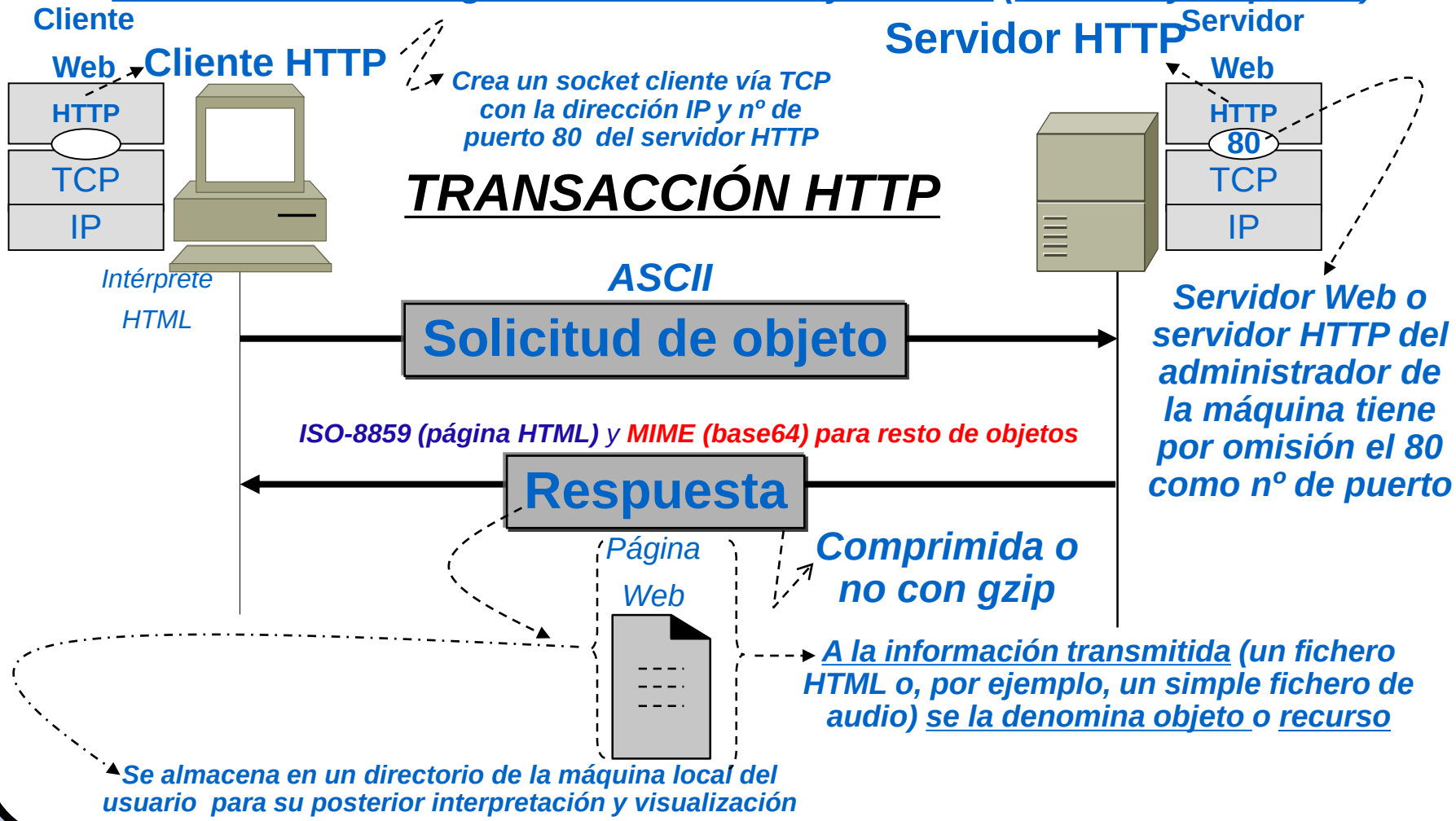
# El Localizador Uniforme de Recursos o Formato URL *(Uniform Resource Locator)*

- *Para que el protocolo HTTP localice en Internet un determinado fichero (objeto o recurso) mantenido por un determinado servidor HTTP o servidor Web, se utiliza un FORMATO conocido como URL que consta de 4 parámetros*
- *protocolo://máquina:puerto/ruta*
  - protocolo: *HTTP*
  - máquina: *Alias o dominio del sitio Web (generalmente, comienza por www) o dirección IP*
    - *En una transacción Web el primer paso consiste en traducir el nombre simbólico del servidor en una dirección IP vía un servidor DNS*
  - puerto: *Número entero que identifica al proceso servidor (campo opcional, si no aparece se asume que es el 80)*
  - ruta: *Nombre del fichero o camino (path) de directorios (separados por "/") para acceder al fichero*

# Protocolo HTTP 2.0 (HyperText Transfer Protocolo) o Protocolo de Transferencia de Hipertexto versión 2.0 RFC-2616/RFC-7230 al RFC-7235

***HTTP versión 2.0 es la versión actual del protocolo HTTP en Internet***

***HTTP 2.0 funciona según el modelo cliente y servidor (solicitud y respuesta)***



# Características de HTTP 2.0

- PERSISTENTE
- CON PIPELINING
- SIN ESTADO

# Características de HTTP 2.0

- PERSISTENTE:
  - *Permite descargar dos o más o ficheros por una única conexión TCP*
    - *En HTTP 1.0 sólo se podía enviar un fichero en una conexión TCP*
  - Con PIPELINING (con "tubería")
    - Por omisión en HTTP 2.0
    - *Permite enviar una solicitud de fichero sin esperar el fichero de la solicitud anterior, es decir, enviar tantas solicitudes, sin esperar a los objetos de invocaciones previas, como objetos o ficheros haya referenciados en el código HTML*
      - Un solo RTT para todas las referencias
  - Sin PIPELINING (sin "tubería")
    - *El cliente envía una nueva solicitud sólo cuando ha recibido el objeto de la anterior solicitud*
      - Un RTT por cada objeto referenciado

# Características de HTTP 2.0

- SIN ESTADO: *El servidor HTTP NO mantiene el estado o la información sobre las solicitudes o acciones (historial o huella) de los clientes HTTP al cerrarse la conexión TCP*. Para mantener el estado, *el programador de la aplicación Web tendrá que gestionar el estado fuera o por encima de HTTP* en base a
  - Cookies *("galletas")*: *Ficheros de texto* o *fragmentos de información (trozos de datos) diferentes que contienen las acciones del usuario para cada servidor Web visitado y que se almacenan en el disco duro del cliente (visitante de una página Web), a través de su navegador, a petición del servidor Web. Esta información puede ser recuperada luego por el servidor en posteriores visitas para diferenciar usuarios y actuar de diferente forma dependiendo del usuario.*
  - *Usos frecuentes*:
    - *Control de usuarios*: *Cuando un usuario introduce su nombre de usuario y contraseña, se almacena una cookie para que no tenga que estar introduciéndolas por cada página del servidor*
    - *Seguimientos de usuarios: Estadísticas de usos, aficiones, cestas virtuales de compras, etc.*
    - *Personalización del sitio Web en función de los hábitos o preferencias del usuario: Particularizar el aspecto en cuanto a presentación y funcionalidad*
    - *…*

# EJEMPLO DE HTTP 2.0 PERSISTENTE

***Se pueden enviar multiples ficheros por una única conexión TCP***

***El usuario introduce el URL: www.periodico.es/deportes/index.html***

***TCP es un servicio orientado a conexión: 3 FASES***

*(o default.html) contiene la página Web inicial o fichero HTML inicial con texto y 10 referencias a imágenes jpg*

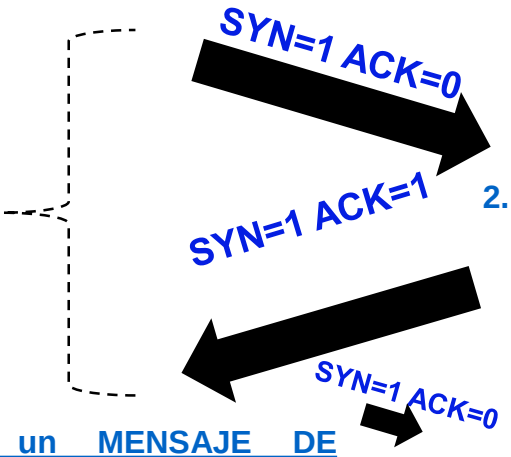
1. Cliente HTTP obtiene un nº de puerto, se conecta a TCP y solicita que establezca una conexión con el socket remoto

**FASE I (TCP):**

**ESTABLECIMIENTO DE LA CONEXIÓN TCP**

**(entre el socket del cliente y el socket del servidor)**

**(3 pasos)**



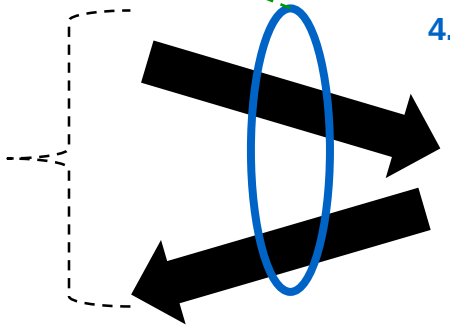

2. Servidor HTTP en *www.periodico.es* que está a la espera de conexiones TCP en el puerto 80, "ACEPTA" la conexión

3. Cliente HTTP envía un MENSAJE DE SOLICITUD (conteniendo el URL) EN FASE DE TRANSFERENCIA DE DATOS

***Se pueden enviar "n" ficheros por la misma conexión sin necesidad de cerrarla***

**FASE II (TCP):**

**TRANSFERENCIA DE DATOS**



4. Servidor HTTP recibe el mensaje de solicitud, y envía el MENSAJE DE RESPUESTA que contiene el objeto requerido y envía dicho mensaje por su socket

(*Aquí, HTTP 1.0 NO PERSISTENTE hubiera solicitado el cierre de la conexión TCP)*

# EJEMPLO DE HTTP 2.0 PERSISTENTE SIN PIPELINING

5. Cliente HTTP recibe el MENSAJE DE RESPUESTA que contiene el fichero HTML, lo interpreta y encuentra *10 referencias a objetos jpeg*

***(continuación)***

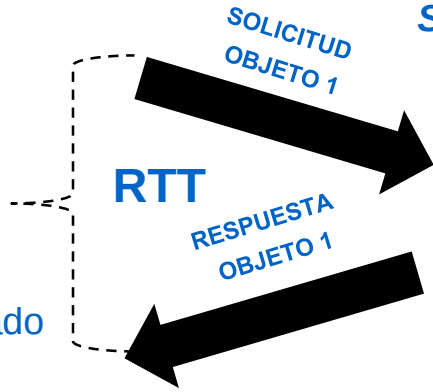
***EL CLIENTE NO PUEDE ENVÍAR UNA NUEVA SOLICITUD HASTA QUE NO HAYA RECIBIDO EL FICHERO DE LA SOLICITUD ANTERIOR***

**FASE II (TCP):**

**TRANSFERENCIA DE DATOS**

***(continuación)***

Un RTT por cada objeto referenciado




6. Servidor HTTP recibe el mensaje de solicitud del objeto 1 y envía un mensaje de respuesta con el objeto 1

7. Cliente HTTP recibe el MENSAJE DE RESPUESTA, lo interpreta, lo visualiza y envía un nuevo mensaje de solicitud de objeto

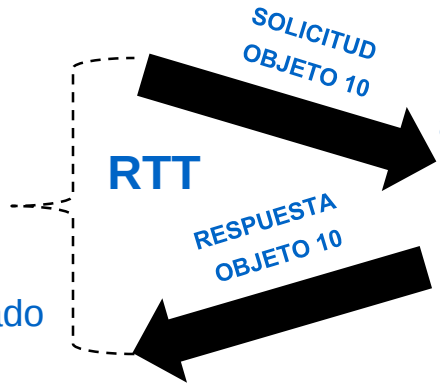
■ ■ ■

**FASE II (TCP):**

**TRANSFERENCIA DE DATOS**

***(continuación)***

Un RTT por cada objeto referenciado




8. Servidor HTTP el mensaje de solicitud, forma el mensaje de respuesta que contiene el objeto requerido, envía dicho mensaje por su socket

**EL SERVIDOR DEJA LAS CONEXIONES ABIERTAS DESPUÉS DE ENVIAR LA RESPUESTA**

9. Cliente HTTP recibe el último objeto y cierra su navegador o escribe un nuevo URL

**FASE III (TCP):**

**LIBERACIÓN DE LA CONEXIÓN**

# EJEMPLO DE HTTP 2.0 PERSISTENTE CON PIPELINING

*(continuación)*

5. Cliente HTTP recibe el MENSAJE DE RESPUESTA que contiene el fichero HTML, lo interpreta y encuentra *10 referencias a objetos jpeg*

## PERSISTENCIA CON PIPELINING:

***EL CLIENTE ENVÍA SOLICITUDES TAN PRONTO ENCUENTRA UN OBJETO REFERENCIADO***

## FASE II (TCP):

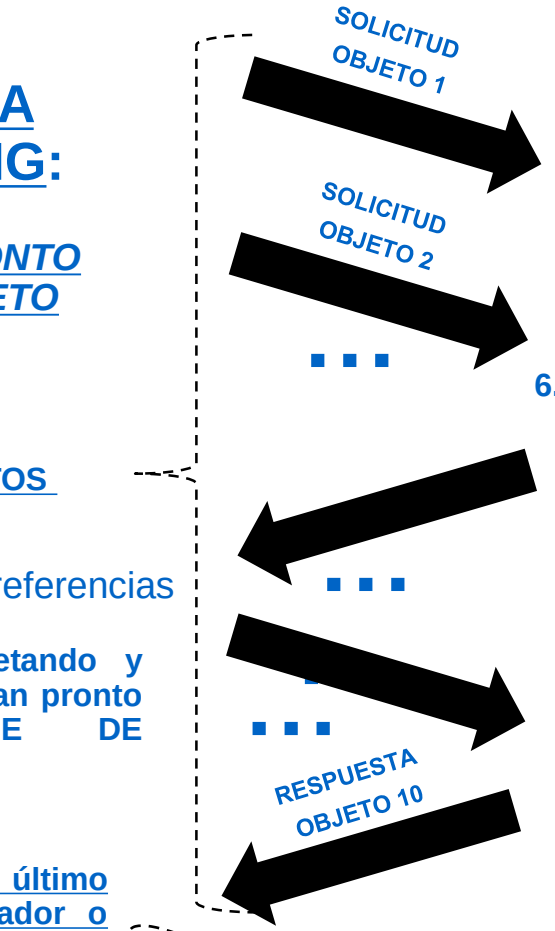
**TRANSFERENCIA DE DATOS**

**1 RTT**

Un solo RTT para todas las referencias

7. Cliente HTTP va interpretando y visualizando cada objeto tan pronto recibe un MENSAJE DE RESPUESTA, de objeto

8. Cliente HTTP recibe el último objeto y cierra su navegador o introduce un nuevo URL




***Se puede solicitar la descarga de un fichero sin esperar a que haya llegado el fichero de la solicitud anterior***

6. Servidor HTTP va enviando un mensaje de respuesta de objeto n tan pronto recibe un mensaje de solicitud de objeto n

## FASE III (TCP):

**LIBERACIÓN DE LA CONEXIÓN**

# Formato de los Mensajes HTTP de Solicitud y Respuesta

## RFC-2616/RFC-7230 al RFC-7235

### CABECERA DE INFORMACIÓN DE CONTROL y CAMPO DATOS o CUERPO DEL MENSAJE

### Mensaje de Solicitud

**Línea de Solicitud**
*GET HTTP/2.0*

**Cabeceras**
*PARES NOMBRE:VALOR*
*idioma, navegador/versión/SO, etc.*
- *Contenido del objeto enviado al servidor*
- *Longitud del objeto enviado (presente sólo en algunos mensajes)*

**Una línea en blanco**

**Cuerpo**
*(objeto que el cliente envía al servidor)*
*(presente sólo en algunos mensajes)*

### Mensaje de Respuesta

**Línea de Estado**
*HTTP 2.0 código-frase*

**Cabeceras**
*PARES NOMBRE:VALOR*
- *Servidor APACHE y versión*
- *Fecha y hora*
- *Contenido del objeto solicitado*
- *Longitud del objeto solicitado.*

**Una línea en blanco**

**Cuerpo**
*(objeto solicitado)*

# Código de Estado Frase de Estado
# Mensaje de Respuesta del Servidor

- **1xx Mensajes**

  **Del 100 al 111 *Conexión rechazada***
- **2xx Operación con éxito**

  ***200 OK***

  **201-203 Información no oficial**

  **204 Sin Contenido**

  **205 Contenido para recargar**

  **206 Contenido parcial**
- **3xx Redirección a otro URL**

  **301 Mudado permanentemente**

  **302 Encontrado**

  **303 Vea otros**

  **304 No modificado**

  **305 Utilice un proxy**

  **307 Redirección temporal**
- **4xx Error por parte del cliente**

  **400 Solicitud incorrecta**

  **401 No autorizado**

  **402 Pago requerido**

  **403 Prohibido**

  **404 No encontrado**

  **409 Conflicto**

  **410 Ya no disponible**

  **412 Falló precondición**
- **5xx Error del servidor**

  **500 Error interno**

  **501 No implementado**

  **502 Pasarela incorrecta**

  **503 Servicio no disponible**

  **504 Tiempo de espera de la pasarela agotado**

  **505 Versión de HTTP no soportada**

# Ejemplo de un Diálogo HTTP

- **Para obtener un recurso vía formato URL**
  - ***http://www.miapache.com/index.html***

***Se abre una conexión al host www.miapache.com vía puerto 80*** *(puerto, por omisión, para HTTP)*

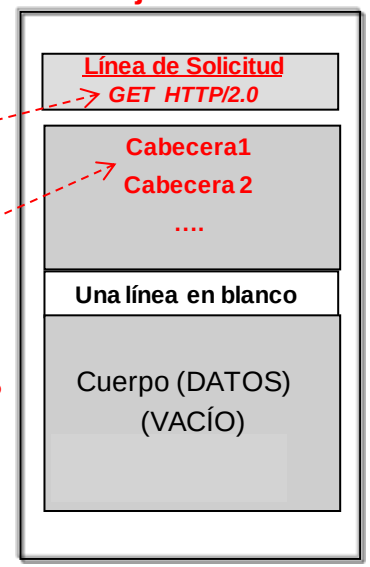
## 1. Solicitud del cliente HTTP

***GET /HTTP/2.0*** ----------> **LÍNEA DE SOLICITUD**

***Host:*** ***www.miapache.com***
*User-Agent:* ***Mozilla Firefox 56.0.2, Windows 10***
*Accept:* ***text/html, application/xml***
*Accept Language:* ***en-US, en***
*Accept Encoding:* ***gzip***
…

**CABECERAS**
***PARES NOMBRE:VALOR***

**Mensaje de Solicitud**




## 2. LA RESPUESTA DEL SERVIDOR

***HTTP/2.0 200 OK*** ----------> **LÍNEA DE ESTADO** (*protocolo/versión* ***código y frase de estado***)

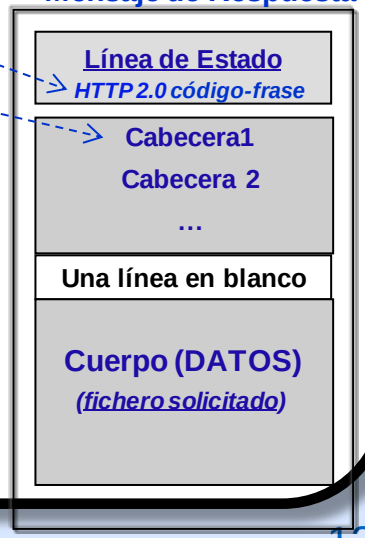
*Server:* ***Apache/2.0.3***
*Date:* ***Fri, 31 Oct 2017 23:59:59 UT***
*Content-Type:* ***text/html , charset =iso-8859***
*Content-Encoding:* ***gzip***
• • •

**CABECERAS**
***PARES NOMBRE:VALOR***

***<html>***
***<body>***
***<h1>Página principal de tuHost</h1>***
***(Contenido)***
. . .
***</body>***
***</html>***

***CUERPO (Datos)*** *(index.html)*

**Mensaje de Respuesta**



# Tipo de un Mensaje de Solicitud

- ***HTTP define 8 métodos implementados en el código del cliente HTTP*** **que indican las acciones sobre el correspondiente fichero,** ***de los cuales los 3 métodos o solicitudes más relevantes son:***
  - **GET**: ***Método HTTP asociado a un enlace en el código HTML de la página web descargada (aplicación) para solicitar un fichero***
    - ***Se ejecuta, generalmente, cuando el usuario hace un "clic" en un enlace o, previamente, cuando el intérprete HTML encuentra una referencia, p.ej., un logo o imagen, mientras dibuja la página***
    - **La mayoría de las solicitudes HTTP son mediante GET**
      - **A veces, GET incluye parámetros visibles que se encapsulan en el LOCALIZADOR URL en una búsqueda de información**
        - **/index.php?page=main&lang=es**
          - ***P.ej., obtención de un objeto llamado logo.png***
            - ✓ ***GET /images/logo.png HTTP/1.1***
  - **POST**: ***Típico método HTTP utilizado en formularios para enviar parámetros o datos del usuario y que no son visibles en el localizador URL***
    - ***El código del formulario exige que los datos introducidos se envíen vía post***
  - **PUT:** ***Método HTTP para enviar un objeto al servidor***
  - ***…***