Tema 4
Nivel de Transporte y Aplicación

Redes de Computadores

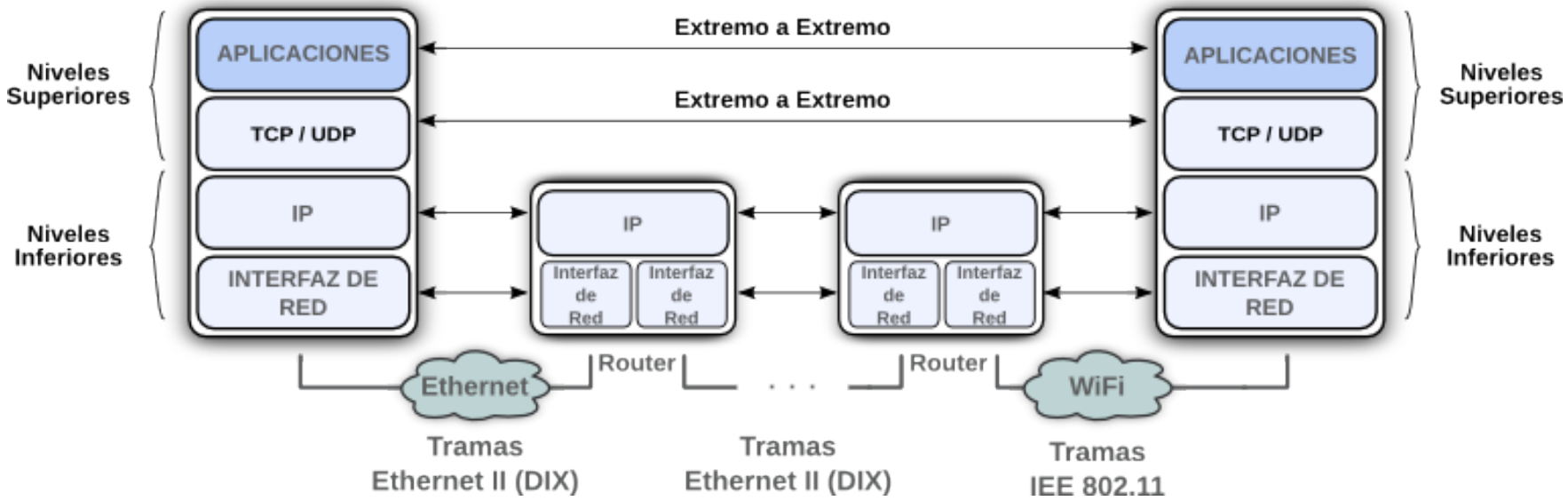
Curso 2015/2016
Primer Semestre

Índice

4.1 Nivel de transporte

Nivel de transporte
Introducción y generalidades

- El nivel de transporte y el nivel de aplicación son los niveles superiores TCP/IP extremo a extremo, independientemente del número de redes y *routers* en el trayecto entre el origen y el destino

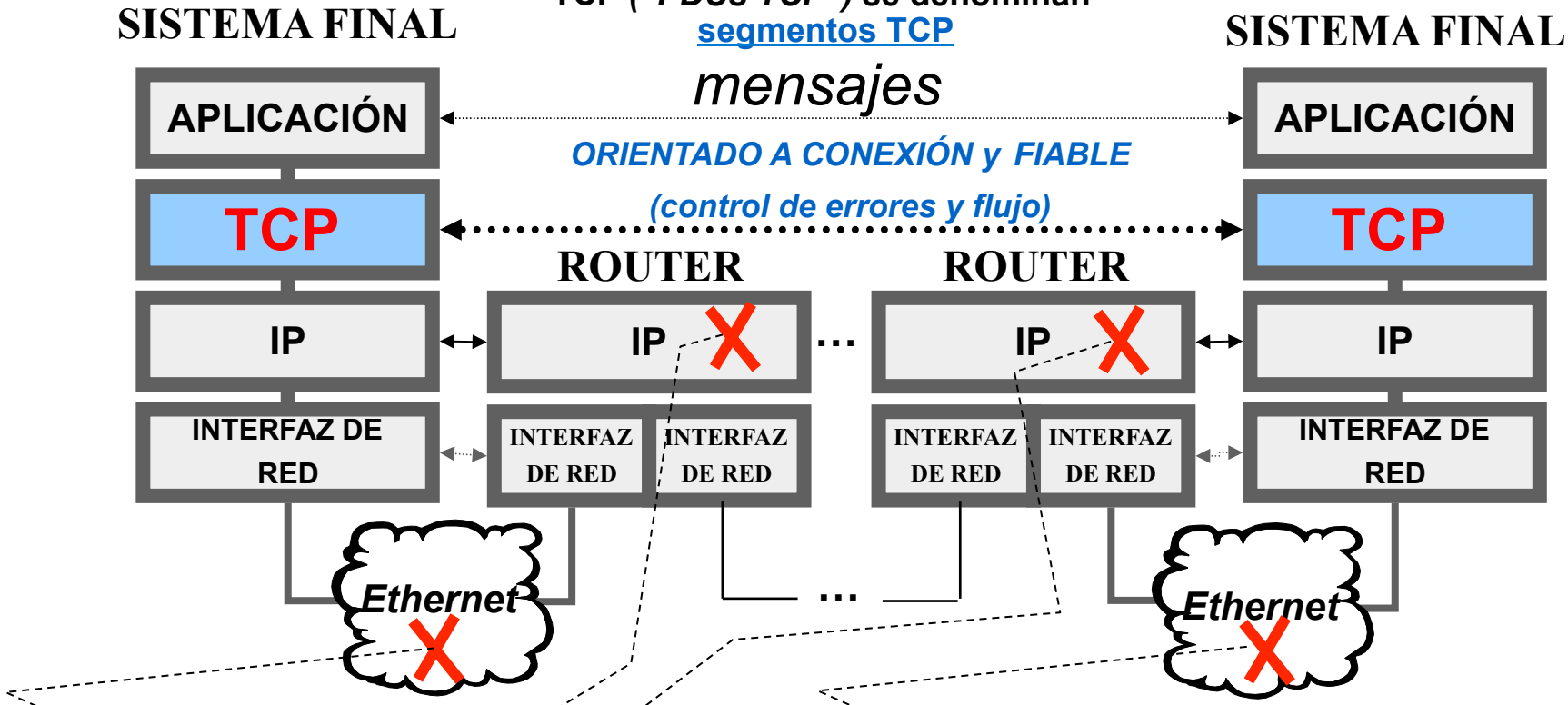


PROTOCOLO TCP (Transmission Control Protocol) RFC-793, STD 0007

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

Las unidades de datos del protocolo TCP *("PDUs TCP")* se denominan segmentos TCP



- ***ERRORES:*** *IPv4 checksum, destino inalcanzable, TTL = 0, …*
- ***CONGESTIÓN:*** *Desborde del buffer del interfaz de salida*
- *…*

FIABILIDAD TCP

- CONTROL DE ERRORES
- CONTROL DE FLUJO

Control de Errores Lógicos TCP

Fallos en la Comunicación

- *TODOS LOS OCTETOS DE DATOS contenidos en el CAMPO DE DATOS de cada segmento de información VAN NUMERADOS (cada octeto tiene su propio nº de secuencia) y dichos octetos tienen asociados*
  - *Una CONFIRMACIÓN*
    - *Las confirmaciones*, por parte de la entidad receptora TCP, *reconocen octetos de datos recibidos correctamente y no los segmentos de información que los contienen porque éstos no van numerados*
  - *Un TEMPORIZADOR de espera de CONFIRMACIÓN*
    - *Al vencimiento del temporizador al no llegar la confirmación, durante el plazo de espera establecido, se produce una retransmisión*

Nivel de transporte
Protocolo TCP

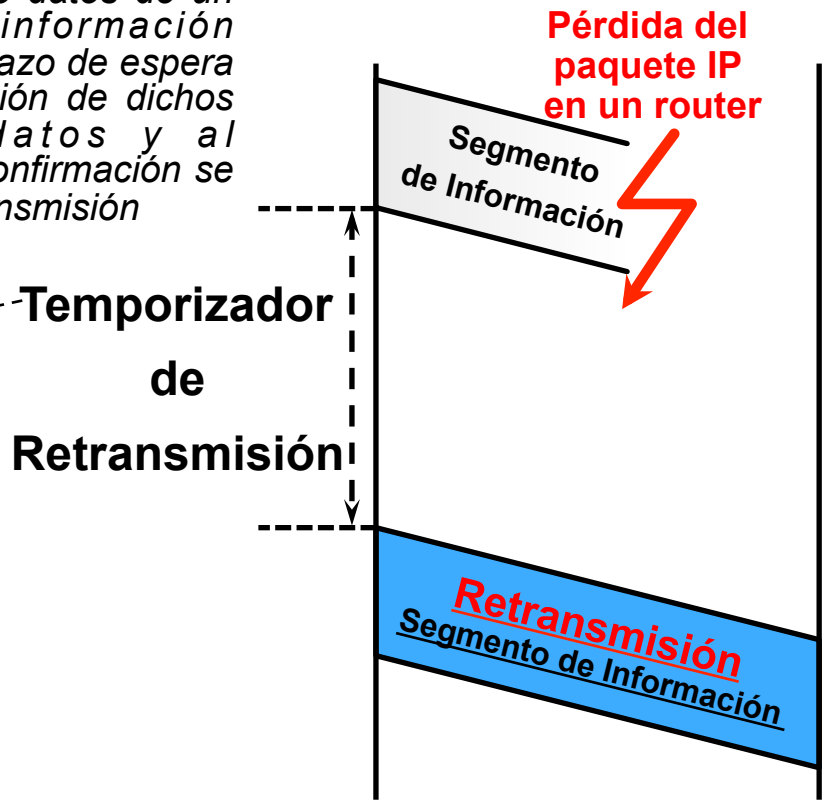
- **Diseño operacional TCP:**
  - Todo protocolo de nivel de aplicación montado sobre TCP se despreocupa de delimitar sus mensajes
  - TCP trata los mensajes , pasados por el proceso de aplicación, como un flujo de octetos (*byte-stream*) que divide en segmentos en función de su MSS (tamaño máximo del campo datos) y de la MTU (1500 octetos) de salida
    - Por esta razón, TCP no numera los segmentos sino los octetos de datos transmitidos

MECANISMOS DE CONTROL DE ERRORES LÓGICOS

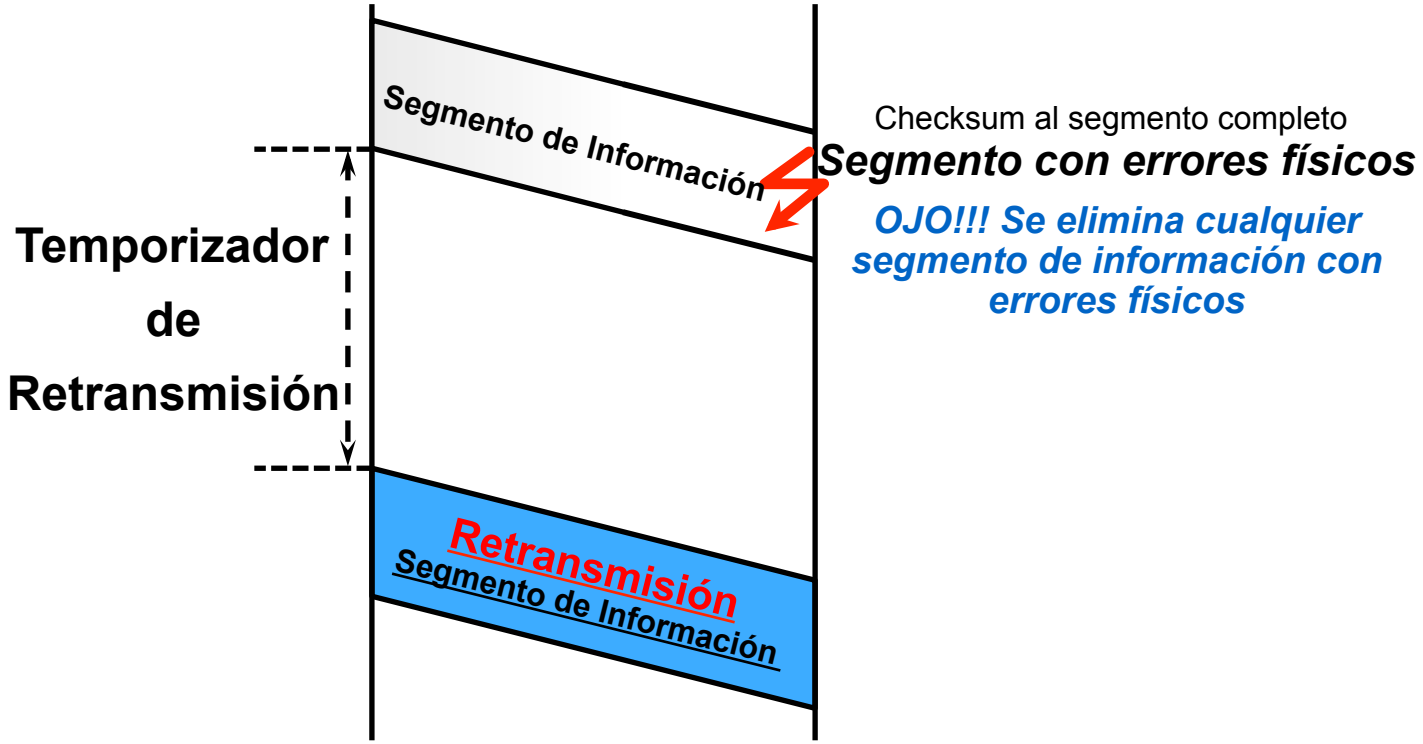
- ***Números de secuencia***: *Todos los octetos del campo datos de un segmento de información disponen de un número de secuencia*
  - *Los números de secuencia permiten pasar al nivel de aplicación los octetos de datos (cabecera de aplicación + campo datos de usuario o carga útil) ordenadamente y detectar octetos duplicados si han llegado las confirmaciones con errores físicos o simplemente no han llegado*
- ***Confirmaciones***: *Todos los octetos del campo datos de un segmento de información tienen asociados una confirmación*
- ***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

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

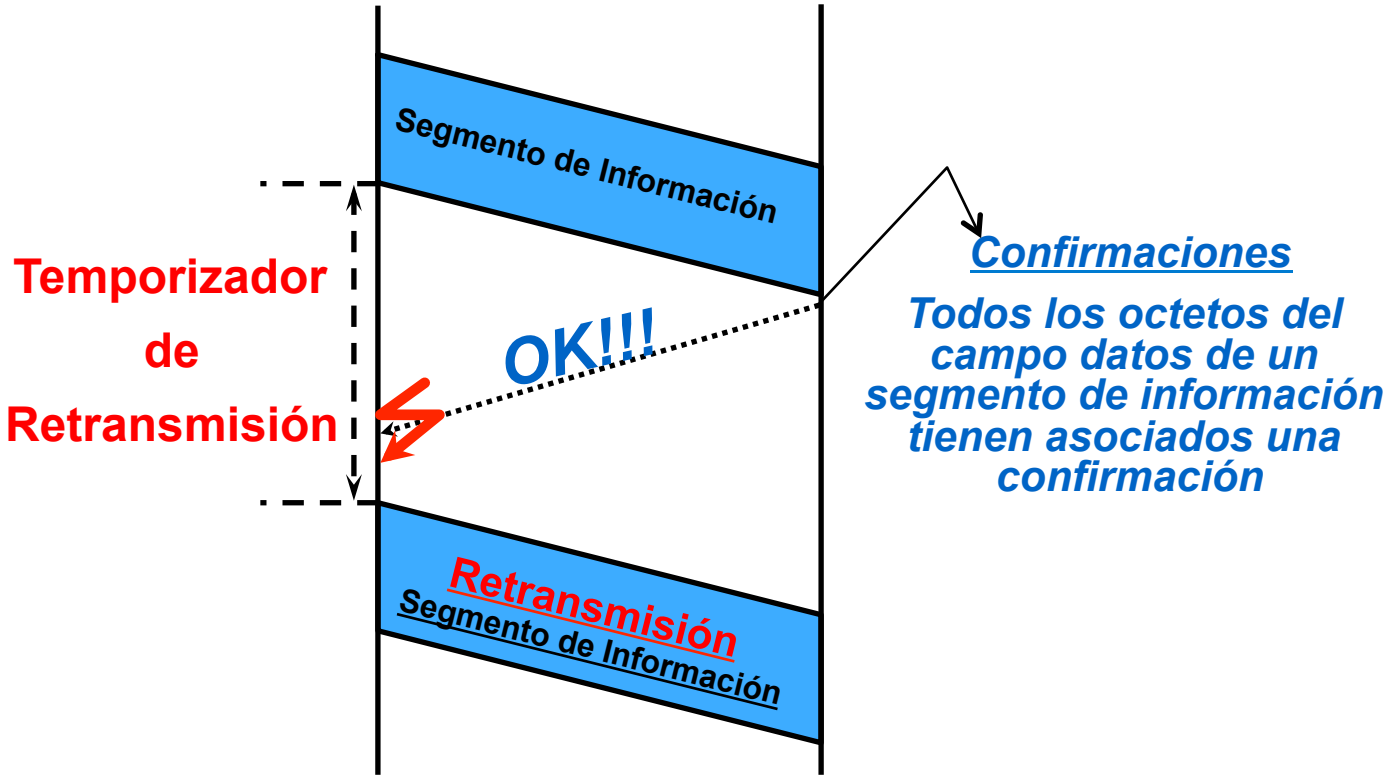


Checksum al segmento completo

Segmento con errores físicos

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

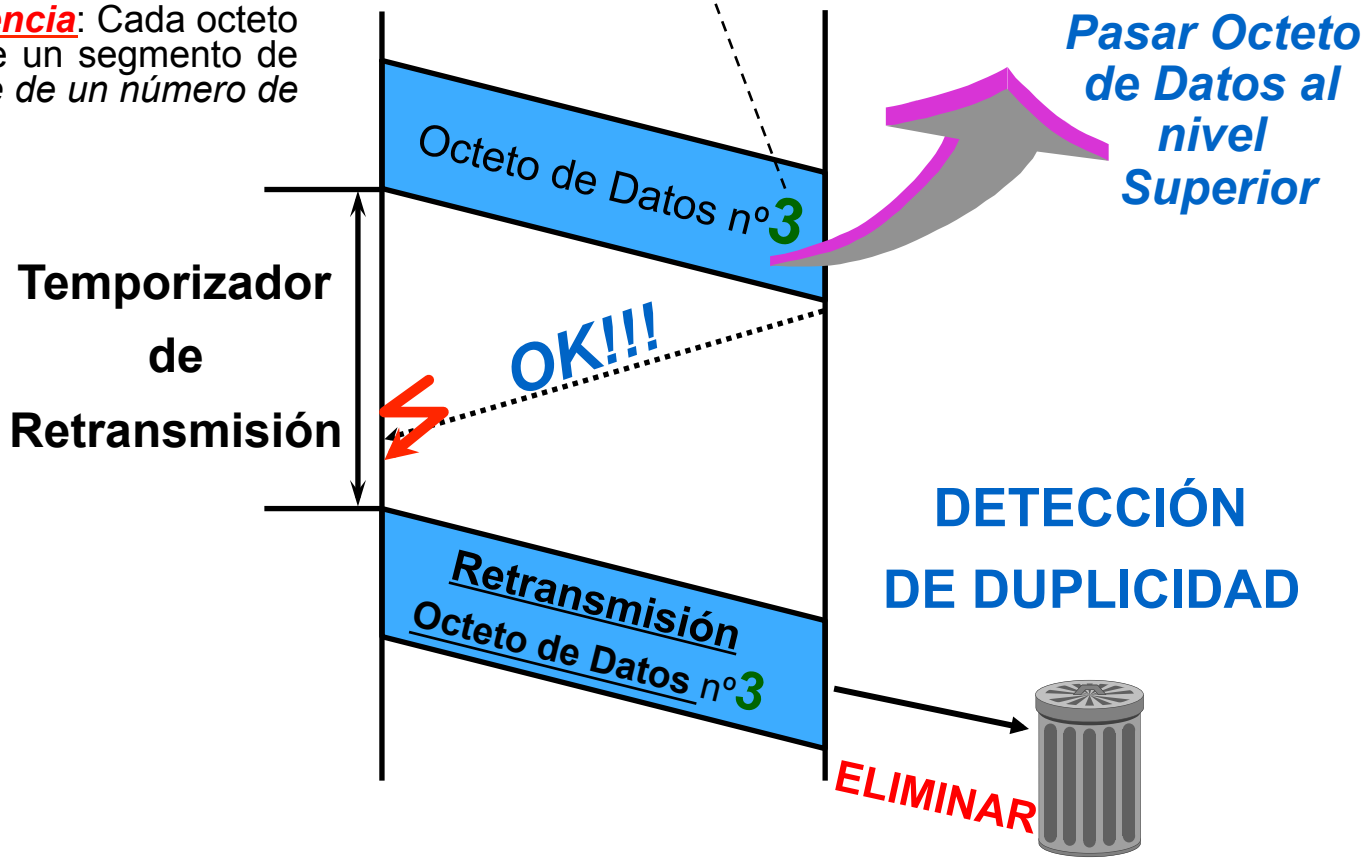
MECANISMOS DE CONTROL DE ERRORES



MECANISMOS DE CONTROL DE ERRORES

Cada octeto de datos con su propio Número de Secuencia

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



Nivel de transporte
Protocolo TCP

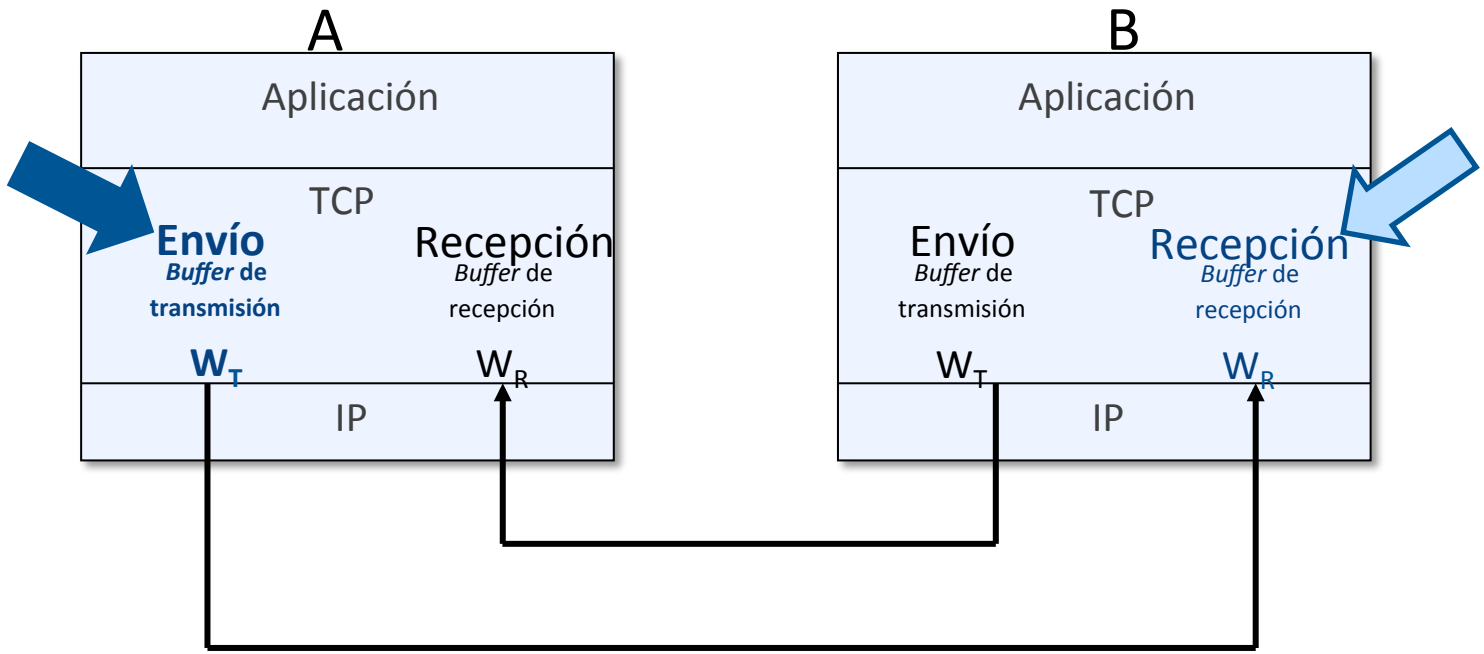
- **CONTROL DE FLUJO: Mecanismo de ventana deslizante** ejercido por el proceso TCP receptor sobre el emisor para evitar que éste desborde el *buffer* del receptor
  - Toda entidad TCP dispone de 2 ventanas:
    - W_T (***buffer* de transmisión**)
    - W_R (***buffer* de recepción**)
  - W_R garantiza que no se inunde al receptor, ejerciendo un control de flujo sobre el emisor
  - W_R controla la numeración de los octetos de datos que en un momento dado el receptor puede aceptar en función del tamaño de su *buffer* de recepción
  - W_R se calcula a partir del número de octetos libres que se pueden almacenar en el *buffer* de recepción de la entidad TCP receptora
  - W_T va variando en fase de transferencia de datos en función de W_R del otro extremo

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 transmisión)

- **Ventana de transmisión W_T:**
  - Lista de números de secuencia consecutivos de los octetos de datos que en un momento dado el emisor ha enviado sin haber recibido confirmación
  - Los octetos de datos procedentes del proceso de aplicación se recogen, numeran y almacenan en el *buffer* de transmisión

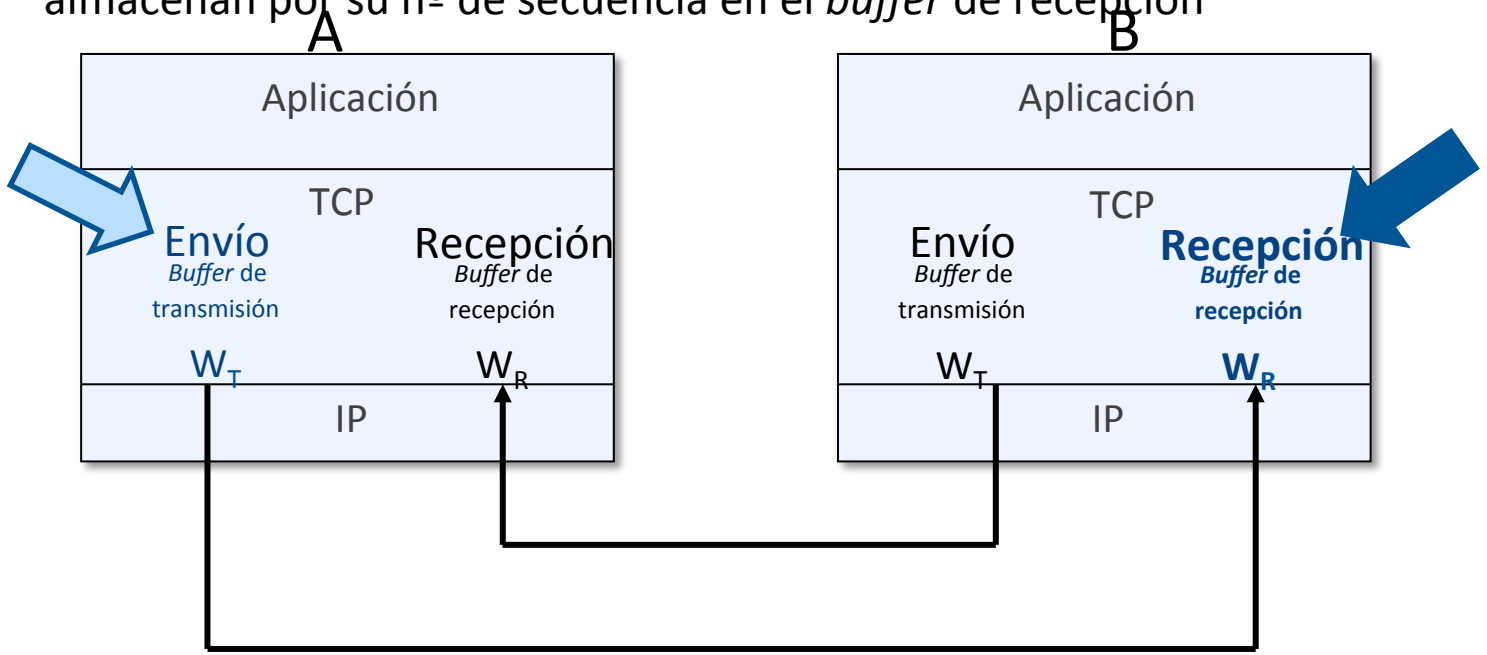


Mecanismo de Control de Flujo

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

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

- **Ventana de recepción W_R:**
  - Lista de números de secuencia consecutivos de los octetos de datos que en un momento dado el receptor puede aceptar
  - Los octetos de datos procedentes de la entidad TCP emisora se recogen y almacenan por su nº de secuencia en el *buffer* de recepción



Mecanismo de Control de Flujo

- *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 para un Correcto Control de Flujo

- *Las implementaciones TCP son muy diferentes en cuanto a los algoritmos auxiliares empleados (p.ej., algoritmos de gestión de ventanas), 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 = *El primer octeto del campo datos del PRIMER SEGMENTO DE INFORMACIÓN que se espera recibir*
  - *Límite superior* de W_R = *El último octeto del campo datos del ÚLTIMO SEGMENTO DE INFORMACIÓN que se espera recibir*
  - *CUANDO LLEGAN OCTETOS DE DATOS CORRECTOS (primer octeto coincide con el límite inferior de W_R)*
    - *Se CONFIRMAN*
    - *Se PASAN 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 para un Correcto Control de Flujo

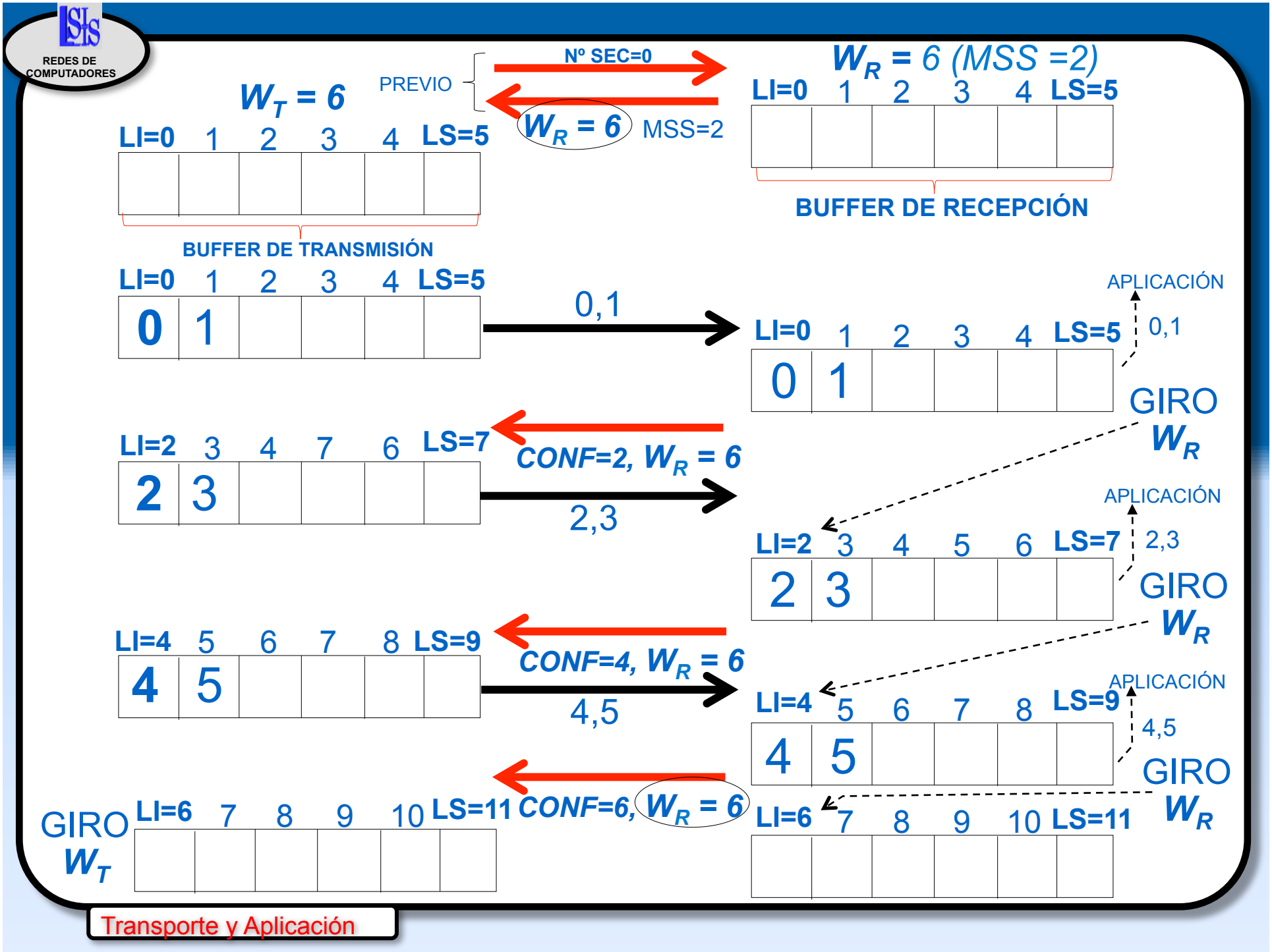
- *W_T inicial = Tamaño máximo del buffer de transmisión*
- *W_T inicial = W_R inicial* (tamaño máximo del buffer de recepción del otro extremo)
  - *Límite inferior* de *W_T* = *El primer octeto pendiente de confirmación del campo datos del PRIMER SEGMENTO DE INFORMACIÓN transmitido*
  - *Límite superior* de *W_T* = *El último octeto pendiente de confirmación del campo datos del ÚLTIMO SEGMENTO DE INFORMACIÓN transmitido*
  - *W_T variando en fase de transferencia de datos en función de la W_R del otro extremo*
  - *CONFIRMACIÓN (ACK) de la entidad TCP receptora: Primer octeto del campo de datos del siguiente segmento de información que se espera recibir, con lo cual los octetos anteriores están todos confirmados y la entidad emisora ajusta el* LÍMITE INFERIOR DE *W_T* al NUEVO valor de CONFIRMACIÓN recibido
  - *CUANDO SE RECIBE UNA CONFIRMACIÓN DEL LÍMITE INFERIOR de W_T*
    - *Es una confirmación de todos los octetos, es decir, desde el primer octeto (límite inferior) hasta el último octeto del campo datos del primer segmento de información transmitido*
    - *Se DESACTIVA el temporizador asociado a los octetos confirmados*
    - *Se ELIMINA copia de los octetos confirmados en el buffer de transmisión*
    - *Se DESLIZA el Límite Inferior Y Superior de W_T en función de los octetos confirmados*

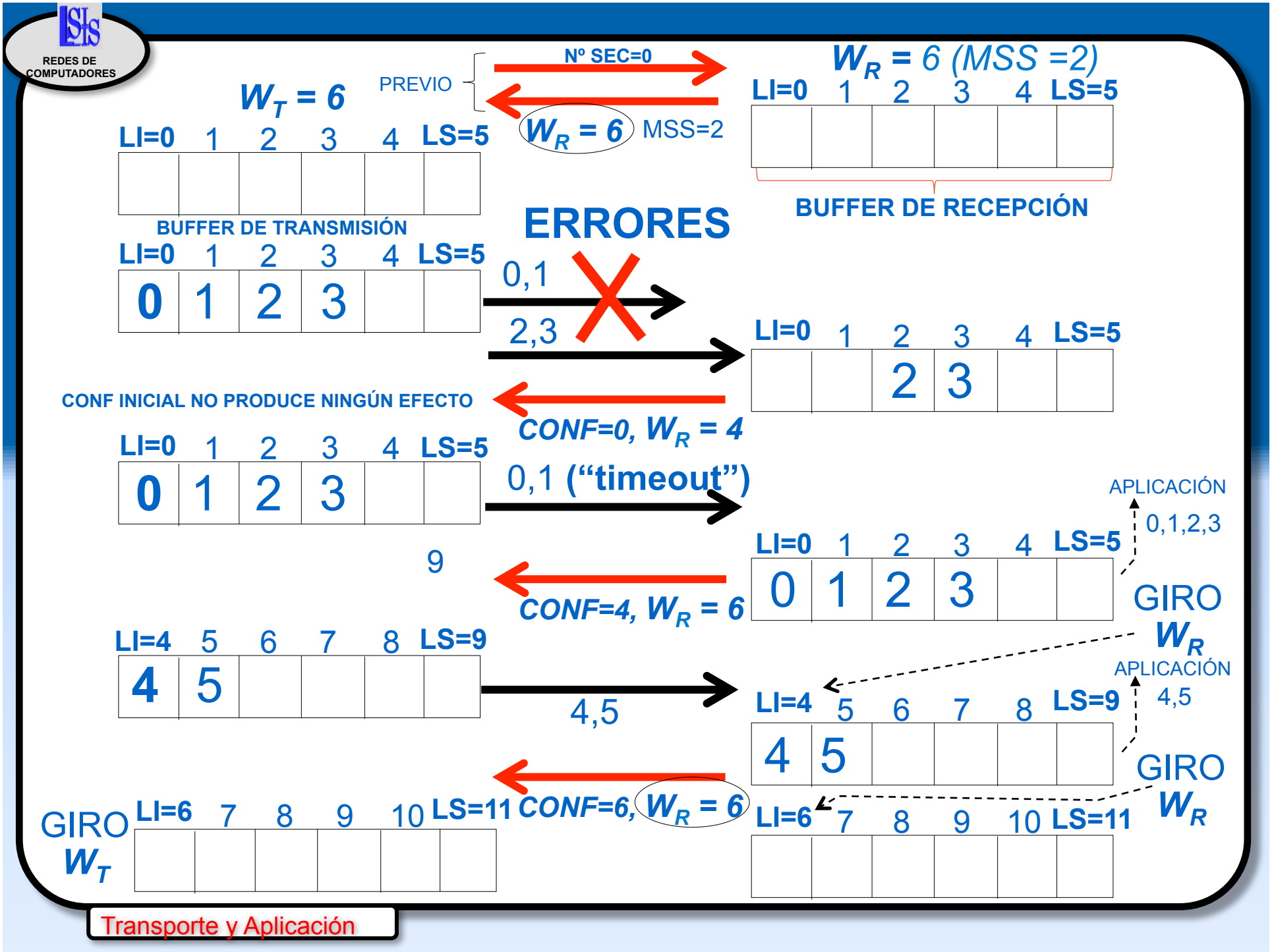
Funcionamiento Actual de W_T

- *RFC-793: CUANDO SE LLENA EL BUFFER DE TRANSMISIÓN, LA ENTIDAD TCP EMISORA VA CREANDO SEGMENTOS DE DATOS*
- *Excepciones:*
  - *Proliferación en Internet de aplicaciones en tiempo real*
    - *Aplicaciones interactivas (VoIP, juegos en red, etc) y no interactivas (streaming de video en HTTP) en tiempo real*
      - ➢ *Si la aplicación pasa, de vez en cuando, 1 bloque reducido de datos en un "chorro de octetos" (byte stream)*
        - ✓ *TCP CREA un segmento de datos y ACTIVA el bit PUSH (= 1), sin esperar a que se llene el buffer de transmisión*
      - ➢ *Si la aplicación pasa de forma continua "n" bloques de datos en un "chorro de octetos" (byte stream)*
        - ✓ *TCP va creando segmentos de datos y los va transmitiendo sin esperar a que se llene el buffer de transmisión*

Funcionamiento Actual de W_R

- *RFC-793: CUANDO SE LLENA EL BUFFER DE RECEPCIÓN, LA ENTIDAD TCP RECEPTORA PASA EL CONTENIDO AL PROCESO DE APLICACIÓN*
- *Excepciones:*
  - *Proliferación en Internet de aplicaciones en tiempo real*
    - *Aplicaciones interactivas (VoIP, juegos en red, etc) y no interactivas (streaming de video en HTTP) en tiempo real*
      - *TCP PASA los octetos de datos recibidos al proceso de aplicación sin esperar a que se llene el buffer de recepción, CONFIRMA y DESLIZA el límite inferior y superior de su W_R*
        - *Si recibe, de vez en cuando, 1 bloque reducido de datos con el bit PUSH = 1 y el primer octeto coincide con el límite inferior de W_R*
        - *Si recibe de forma continua "n" bloques de datos, y el primer octeto coincide con el límite inferior de W_R*


$W_T = 6$
PREVIO
Nº SEC=0
$W_R = 6$ MSS=2
$W_R = 6$ (MSS =2)
LI=0 1 2 3 4 LS=5
BUFFER DE TRANSMISIÓN
BUFFER DE RECEPCIÓN
0 1
0,1
APLICACIÓN
GIRO W_R
2 3
CONF=2, $W_R = 6$
2,3
4 5
CONF=4, $W_R = 6$
4,5
CONF=6, $W_R = 6$
GIRO W_T
LI=6 7 8 9 10 LS=11

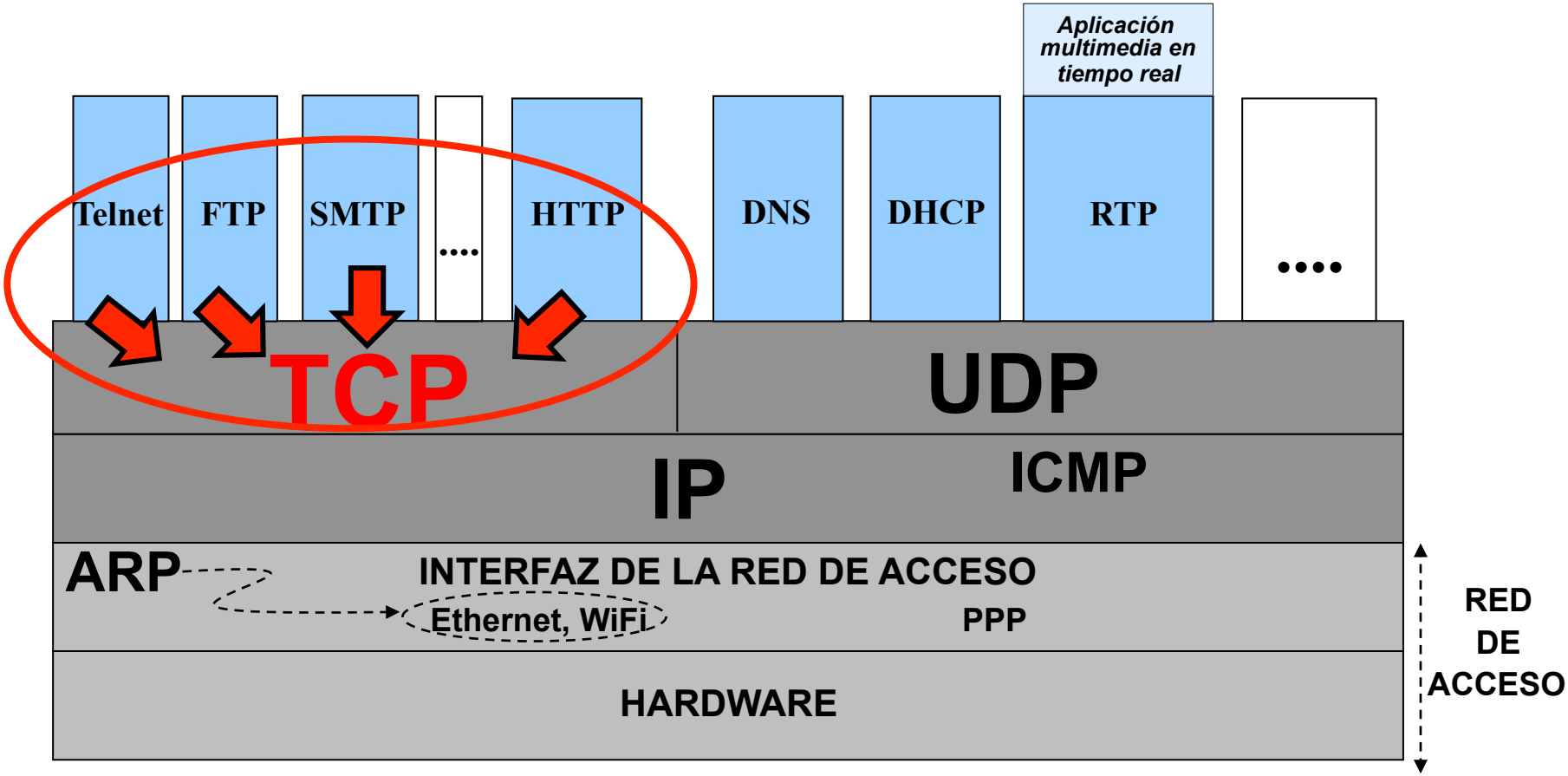

$W_T = 6$
PREVIO
Nº SEC=0
$W_R = 6$ MSS=2
$W_R = 6$ (MSS =2)
LI=0 1 2 3 4 LS=5
BUFFER DE TRANSMISIÓN
BUFFER DE RECEPCIÓN
ERRORES
0,1
2,3
CONF INICIAL NO PRODUCE NINGÚN EFECTO
CONF=0, $W_R = 4$
0,1 ("timeout")
APLICACIÓN
0,1,2,3
9
CONF=4, $W_R = 6$
GIRO W_R
LI=4 5 6 7 8 LS=9
4,5
APLICACIÓN
4,5
GIRO W_R
CONF=6, $W_R = 6$
LI=6 7 8 9 10 LS=11
GIRO W_T

Nivel de transporte
Protocolo TCP

- **Servicios:**
  - **Flujo de octetos** (*byte-stream*): permite al proceso de aplicación (emisor) transmitir un flujo continuo de octetos a su entidad TCP emisora para que ésta los recoja, numere y agrupe en segmentos. La entidad TCP emisora calcula un MSS (*Maximum Segment Size*) para que los datagramas IP se correspondan con la MTU de la red de acceso.
  - **Orientado a conexión** (tres fases) y fiable
    - Control de errores:
      - Detección: Mecanismo de suma de comprobación
      - Corrección: Mecanismo temporizadores y retransmisión
    - Control de flujo: Mecanismo de ventana deslizante ejercido por el receptor sobre el emisor para evitar que éste desborde el *buffer* del receptor
  - **Multiplexado/demultiplexado**: a través de los números de puerto
  - **Full-duplex**: transferencia bidireccional y simultánea

Servicio Multiplexado TCP

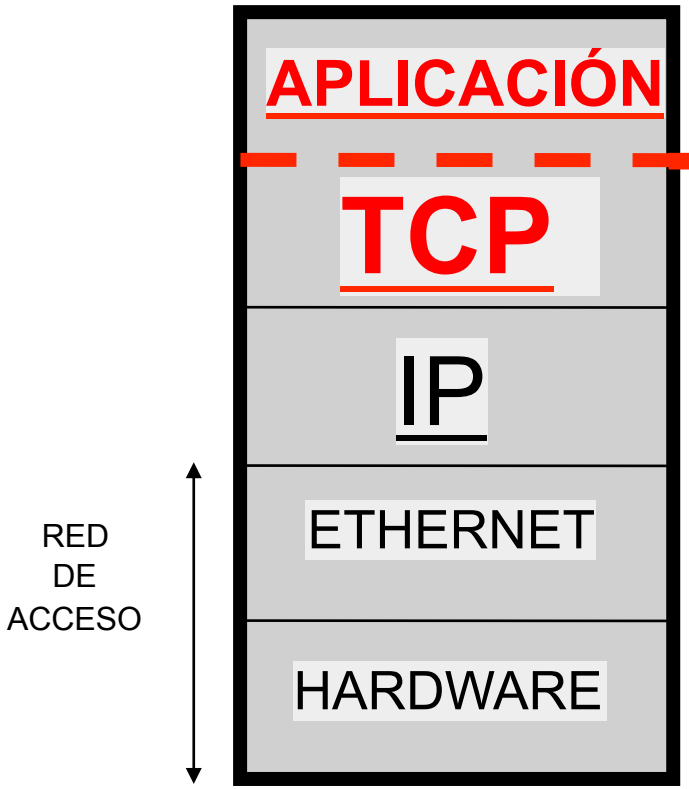
SIMULTÁNEO y DIFERENCIADO a través de los números de puerto, aplicando mecanismos y recursos de fiabilidad por separado



TCP ofrece un SERVICIO ORIENTADO A CONEXIÓN FIABLE a los Procesos del Nivel de Aplicación

- Un SERVICIO ORIENTADO A CONEXIÓN dispone de TRES FASES
  1. ESTABLECIMIENTO DE LA CONEXIÓN
  2. TRASFERENCIA DE DATOS
  3. LIBERACIÓN DE LA CONEXIÓN
- *¿Cómo accede la entidad de aplicación al servicio TCP?*
  - Haciendo uso tanto en el *código del cliente* como en el *código del servidor* de unas *funciones de comunicaciones estándares* pertenencientes a un *API de Programación de Aplicaciones en red* (API de Red o API de Sockets) denominado *INTERFAZ DE SOCKETS*

INTERFAZ DE SOCKETS
IMPLEMENTACIÓN DEL SERVICIO TCP



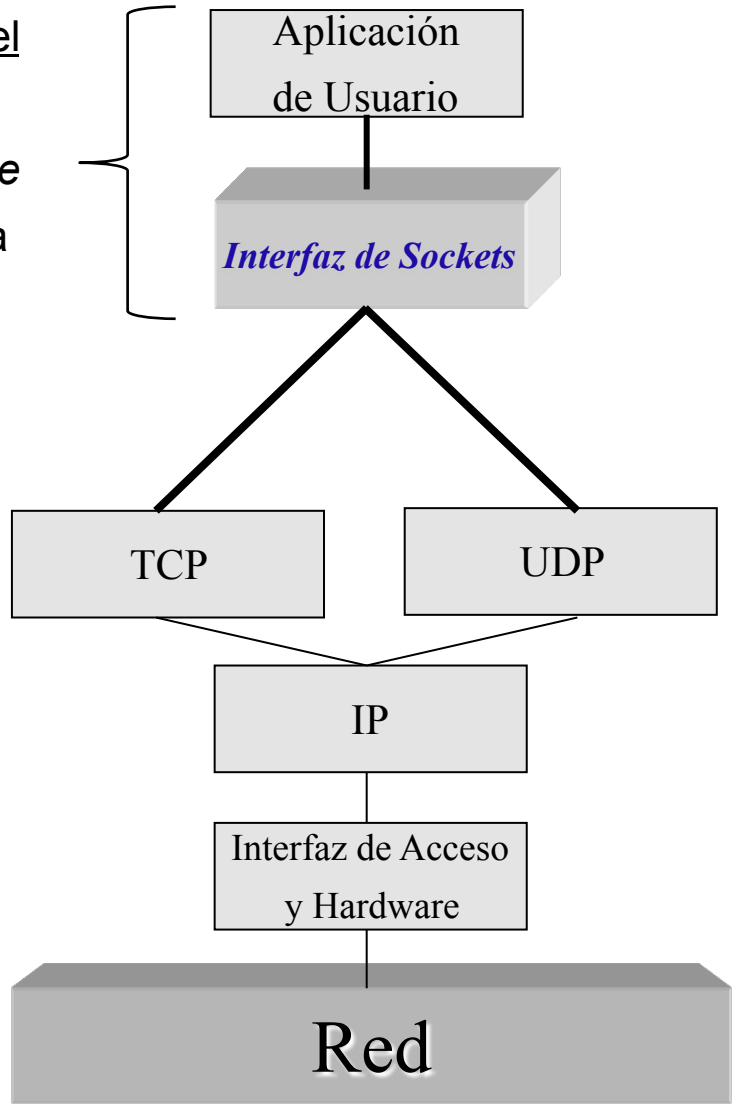
Localización conceptual

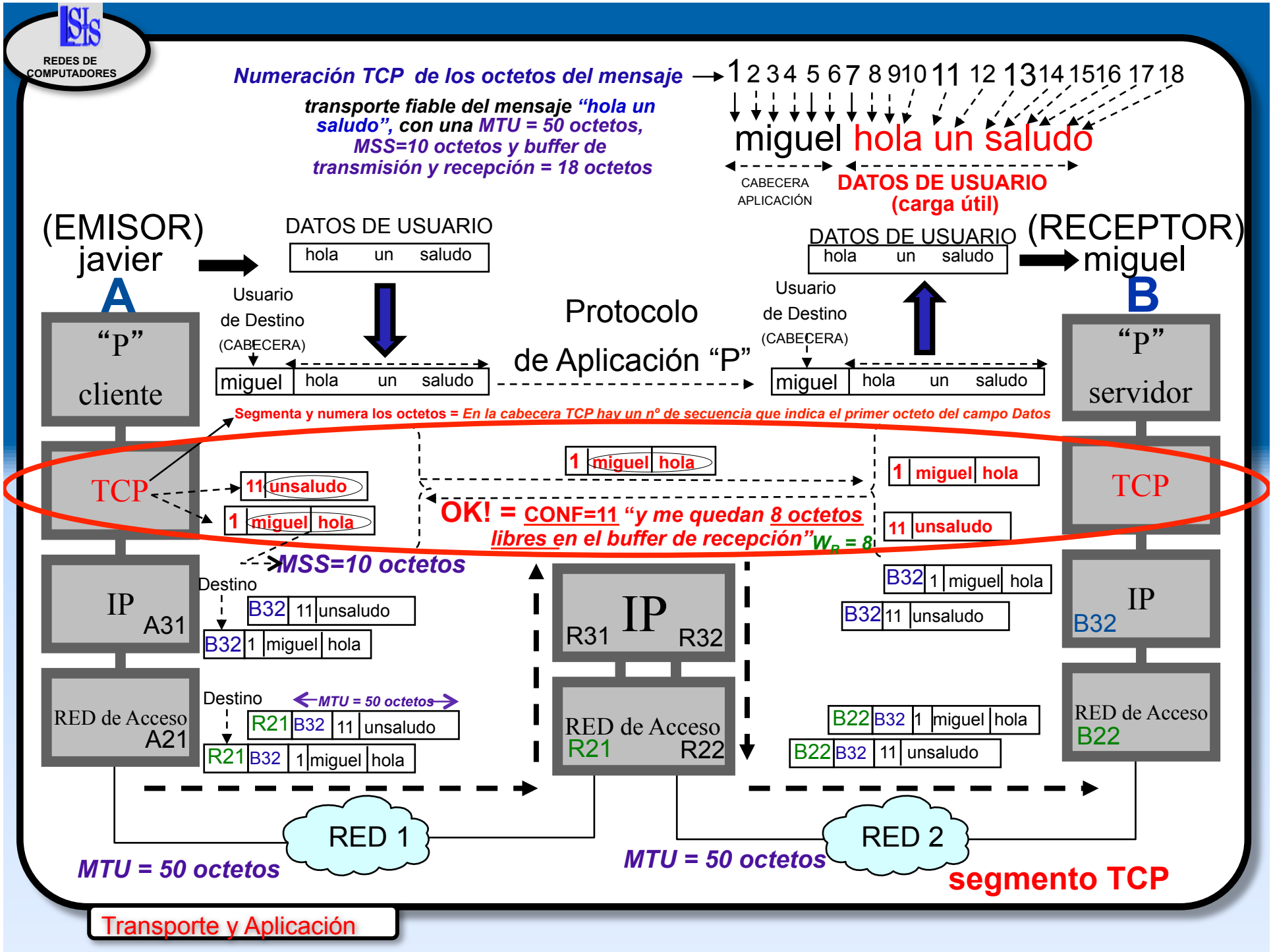
Interfaz de sockets

INTERFAZ DE SOCKETS

Universidad de Berkeley

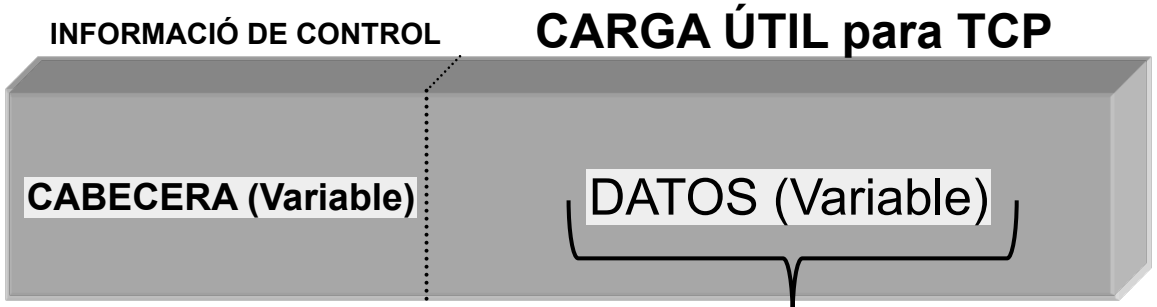
El programador tiene que trabajar a bajo nivel y llamar directamente a las funciones de comunicaciones del *API de programación* proporcionado por un sistema basado en un interfaz de sockets




Numeración TCP de los octetos del mensaje
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
transporte fiable del mensaje "hola un saludo", con una MTU = 50 octetos, MSS=10 octetos y buffer de transmisión y recepción = 18 octetos
miguel hola un saludo
CABECERA APLICACIÓN
DATOS DE USUARIO (carga útil)
(EMISOR) javier
A
DATOS DE USUARIO
hola un saludo
Usuario de Destino (CABECERA)
miguel hola un saludo
Protocolo de Aplicación "P"
DATOS DE USUARIO
hola un saludo
(RECEPTOR) miguel
B
"P" cliente
"P" servidor
Segmenta y numera los octetos = En la cabecera TCP hay un nº de secuencia que indica el primer octeto del campo Datos
TCP
11 unsaludo
1 miguel hola
MSS=10 octetos
1 miguel hola
OK! = CONF=11 "y me quedan 8 octetos libres en el buffer de recepción" $W_R = 8$
1 miguel hola
11 unsaludo
TCP
IP A31
Destino
B32 11 unsaludo
B32 1 miguel hola
IP R31 R32
B32 1 miguel hola
B32 11 unsaludo
IP B32
RED de Acceso A21
Destino
MTU = 50 octetos
R21 B32 11 unsaludo
R21 B32 1 miguel hola
RED de Acceso R21 R22
B22 B32 1 miguel hola
B22 B32 11 unsaludo
RED de Acceso B22
RED 1
RED 2
MTU = 50 octetos
MTU = 50 octetos
segmento TCP

Formato de un Segmento TCP

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



20 octetos
sin opciones

Hasta un máximo conocido como MSS

TCP calcula un MSS de tal forma que los datagramas IP resultantes se correspondan con la MTU de la red de acceso

Maximum Segment Size (MSS) = Carga Útil =
= SDU del Nivel de Aplicación = ***1024 octetos (MSS por omisión)***

MSS = MTU (1500 octetos)- cabecera IP - cabecera TCP

Cabecera TCP + DATOS

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 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)

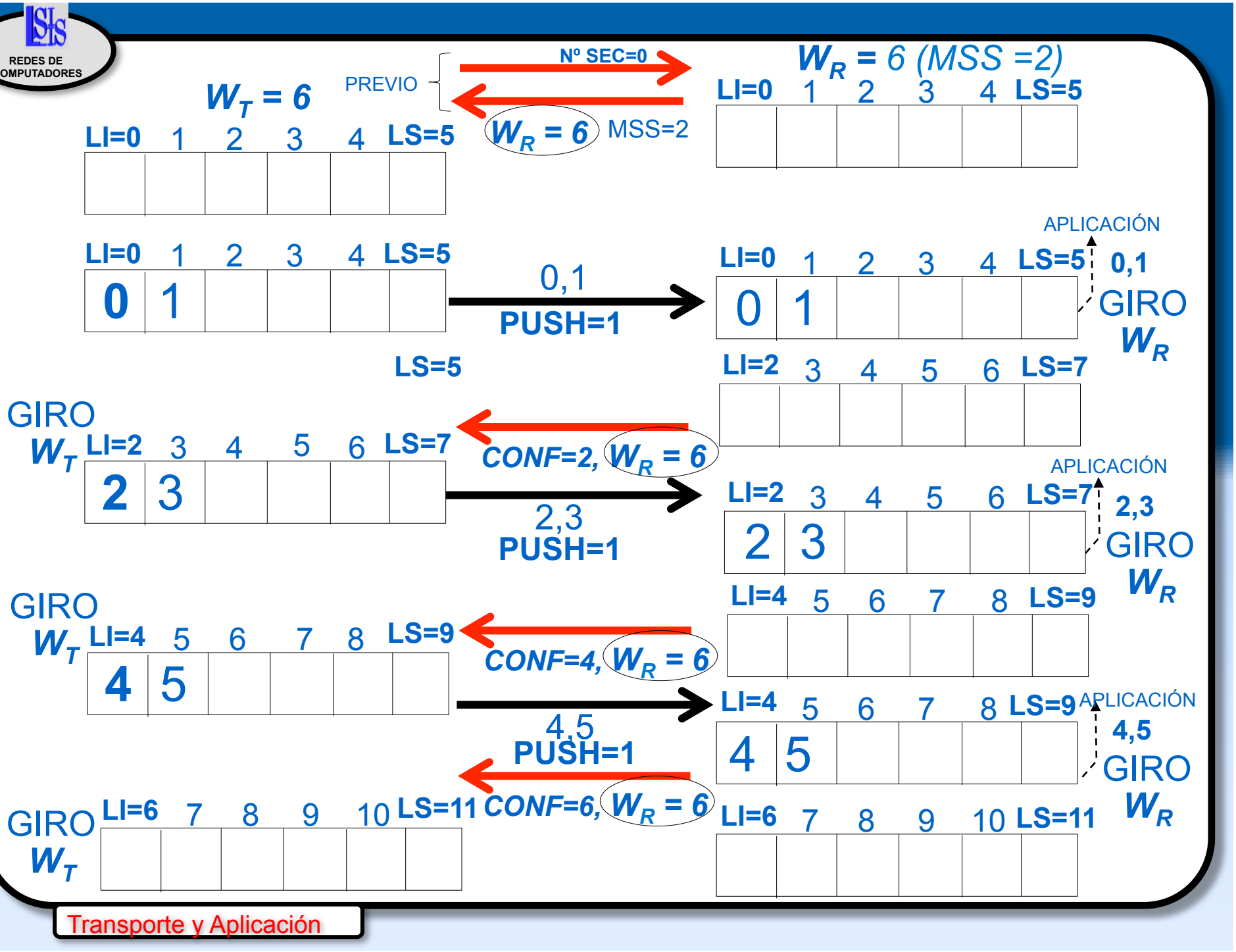
Cantidad de octetos libres que se pueden almacenar en el buffer de recepción

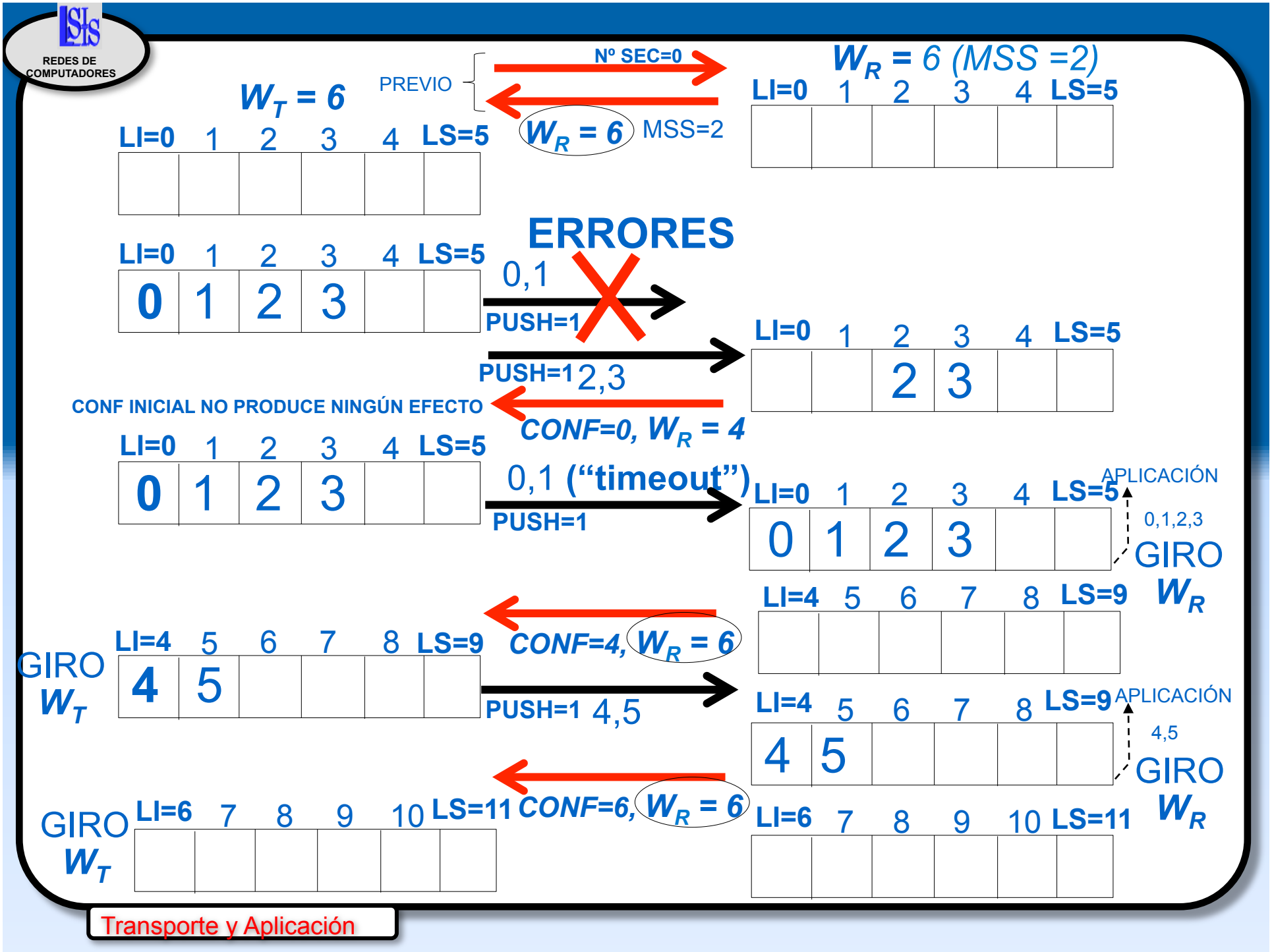
Nivel de transporte
Protocolo TCP

- Campos de la cabecera de control de un segmento TCP:
  - **Puerto de origen**: Proceso de aplicación emisor
  - **Puerto de destino**: Proceso de aplicación receptor
  - **Número de secuencia**: Primer octeto del campo de datos del segmento de información que se va enviar. Se trabaja en módulo 2^{32}
  - **Número de confirmación** (ACK): Primer octeto del campo de datos del siguiente segmento de información que se espera recibir. Se confirma los octetos anteriores
  - **Desplazamiento de datos** (Longitud de la cabecera): Número de bloques de cuatro octetos que ocupa la cabecera
  - **Reservado**: Seis bits a cero reservados para un uso futuro
  - **URG**: Si está activo, indica a la entidad TCP receptora que debe analizar el campo Puntero de Datos Urgentes
  - **ACK**: Si está activo, indica a la entidad TCP receptora que debe analizar el campo Número de Confirmación (ACK)

Nivel de transporte
Protocolo TCP

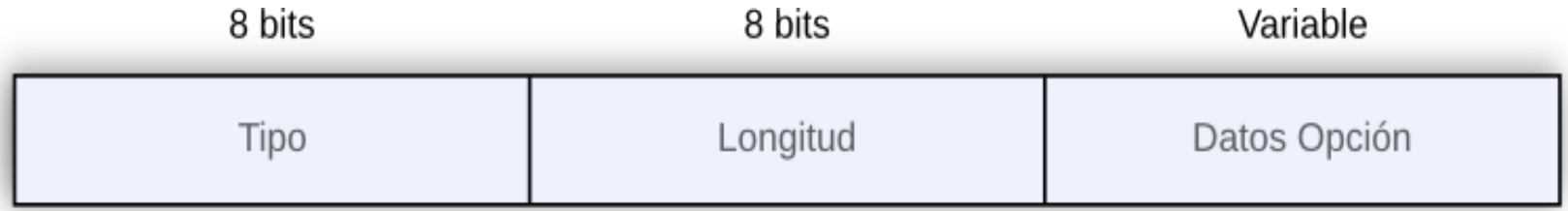
- Campos de la cabecera de control de un segmento TCP (cont.):
  - **PSH**: Si está activo, obliga a la entidad TCP emisora que transmita sin esperar a que se llene su *buffer* de transmisión
  - **RST**: Si está activo, indica a la entidad TCP receptora que abandone la comunicación debido a un error o situación anormal
  - **SYN**: Si está activo, indica que se está estableciendo una conexión TCP
  - **FIN**: Bit de finalización o liberación de la transmisión
  - **Ventana**: Ventana de recepción (W_R): Nº de octetos de datos que la entidad TCP emisora puede almacenar en función del tamaño actual de su *buffer* de recepción
  - **Suma de Comprobación**: Suma aritmética binaria del segmento completo
  - **Puntero de Datos Urgentes**: Si el bit URG está activo, el valor de este campo sumado al valor del campo Número de Secuencia, apunta al último octeto de los datos urgentes de control
  - **Opciones**: Contiene opciones de servicios extras


$W_T = 6$
PREVIO
Nº SEC=0
$W_R = 6$ (MSS =2)
$W_R = 6$ MSS=2
LI=0 1 2 3 4 LS=5
APLICACIÓN
0,1
GIRO W_R
PUSH=1
LS=5
LI=2 3 4 5 6 LS=7
GIRO W_T
CONF=2, $W_R = 6$
2,3
LI=4 5 6 7 8 LS=9
CONF=4, $W_R = 6$
4,5
LI=6 7 8 9 10 LS=11
CONF=6, $W_R = 6$


$W_T = 6$
PREVIO
Nº SEC=0
$W_R = 6$ (MSS =2)
$W_R = 6$ MSS=2
LI=0 1 2 3 4 LS=5
ERRORES
0,1
PUSH=1
PUSH=1 2,3
CONF INICIAL NO PRODUCE NINGÚN EFECTO
CONF=0, $W_R = 4$
0,1 ("timeout")
PUSH=1
APLICACIÓN
0,1,2,3
GIRO W_R
LI=4 5 6 7 8 LS=9
CONF=4, $W_R = 6$
GIRO W_T
PUSH=1 4,5
4,5
LI=6 7 8 9 10 LS=11
CONF=6, $W_R = 6$

Nivel de transporte
Protocolo TCP

- **Campo de opciones:**
  - **Se especifican en la fase de establecimiento de la conexión (segmentos de control con el bit SYN= 1)**



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	-	*SACK PERMITTED* *(Fase de establecimiento)*
5	Variable	-	*SACK* *(Fase de transferencia de datos)*
8	10	Valor actual reloj emisor (4 octetos) + Respuesta Eco (4 octetos)	***Marca de Tiempo***
…	…	…	…

Nivel de transporte
Protocolo TCP

- **Opciones:**
  - **Tamaño Máximo de Segmento** (MSS): Nº máximo de octetos de la carga útil que el receptor (emisor de esta información) desea recibir para un procesado óptimo de dicha carga
  - **Factor de Escala de la Ventana**: Permite ampliar el campo VENTANA
  - **Marca o sello de tiempo** (*Timestamp*): Permite al emisor configurar sus temporizadores mediante el cálculo del valor RTT (*Round Trip Time*): Tiempo de ida y vuelta desde que se envía un segmento de información hasta que se recibe su ACK

Nivel de transporte
Protocolo TCP

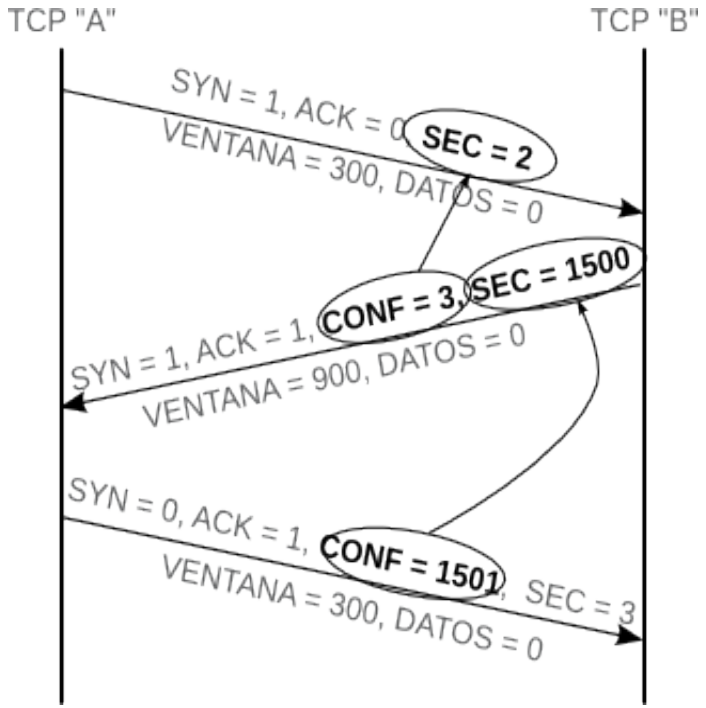
- **Gestión de temporizadores:**
  - La elección de valores adecuados es mucho más compleja en el nivel de transporte que en el nivel de enlace
    - Diferencias en capacidad y retardo de unas conexiones TCP a otras
    - Oscilaciones debidas a la presencia de *routers* y situaciones de congestión
    - Los *routers* pueden tener largas colas de datagramas IP que atender, los enlaces pueden ser de diferentes velocidades y retardos y la ruta puede variar durante la conexión
  - La elección de un valor adecuado tiene una consecuencia directa en el funcionamiento eficiente de TCP
    - Si el temporizador es demasiado alto, el emisor esperará innecesariamente en ciertos casos por ACKs que nunca llegarán
    - Si el temporizador es demasiado bajo se producirán retransmisiones innecesarias de segmentos que habían sido correctamente recibidos

Nivel de transporte
Protocolo TCP

- **Gestión de temporizadores** (cont.)**:**
  - Manejar los temporizadores (*timeouts*) es siempre complejo porque hay que determinar "aproximadamente" el periodo de tiempo existente desde que se envía un segmento de información hasta que se recibe su ACK, es decir, el tiempo de ida y vuelta (RTT: *Round Trip Time*)
  - Los valores del temporizador se establecen mediante algoritmos autoadaptativos que dinámicamente ajustan los valores al estado de la red, percibido por la entidad TCP emisora
  - **Algoritmo autoadaptativo de Karn** para el cálculo del RTT:
    - Se toma una muestra RTT de cada segmento de información transmitido (opción Marca de Tiempo)
    - Se obtiene un promedio de dichas muestras
    - Se le añade un margen de seguridad

Nivel de transporte
Protocolo TCP

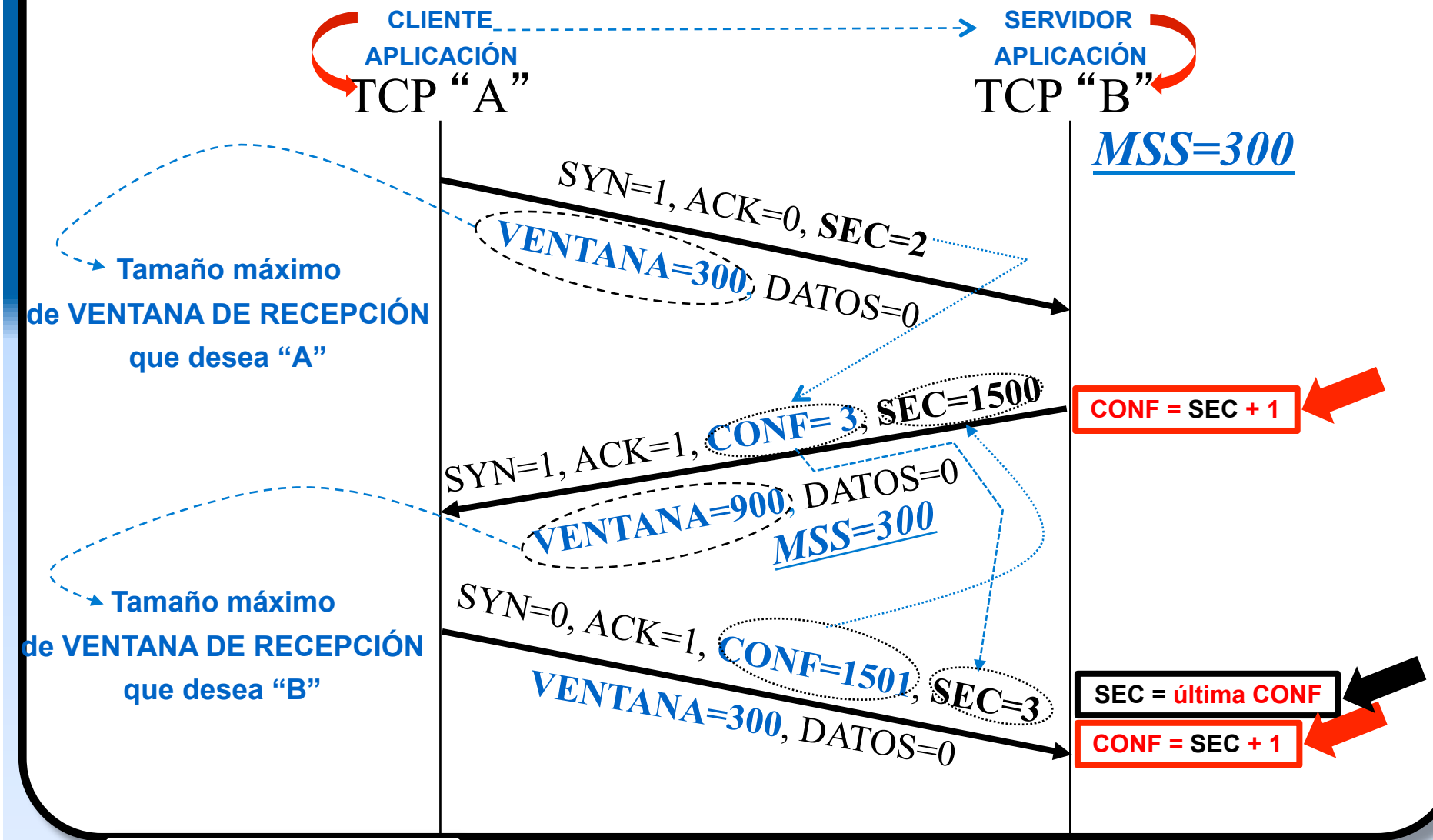
- **Fase de establecimiento de una conexión TCP:**
  - Indicar el tamaño máximo de W_R
  - Negociar opciones y números de secuencia iniciales empleados en ambos extremos de la comunicación



EJEMPLO de Fase de Establecimiento de una Conexión TCP

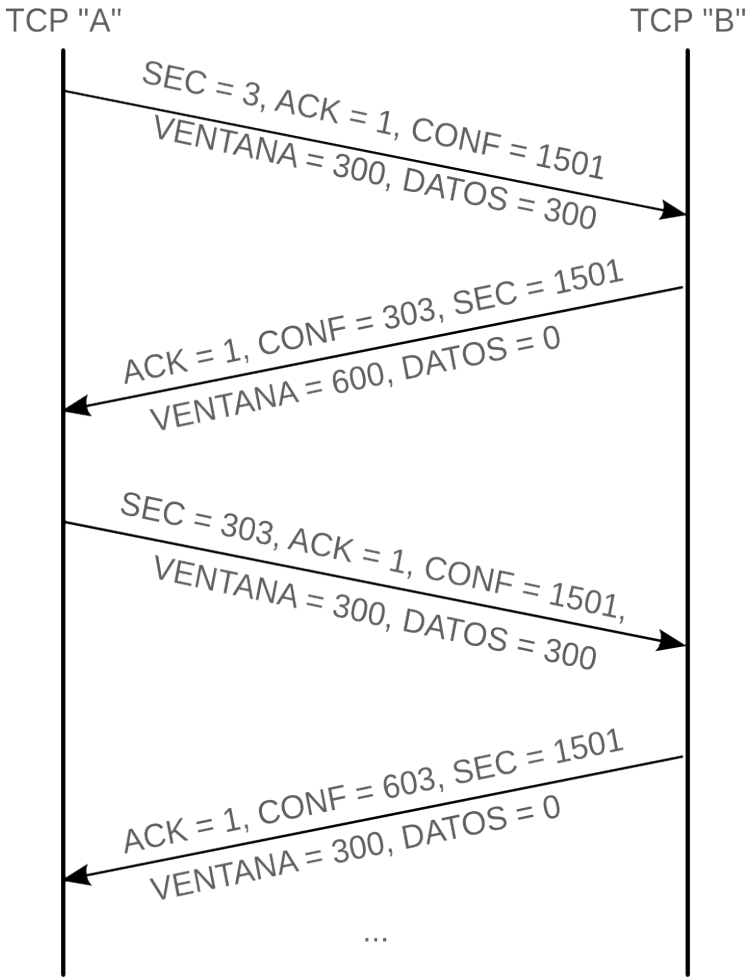
Intercambio de 3 segmentos con la Opción MSS en un extremo ("B")

Con VENTANA en "A" de 300 octetos y en "B" de 900 octetos y MSS de 300 en "B"



Nivel de transporte
Protocolo TCP

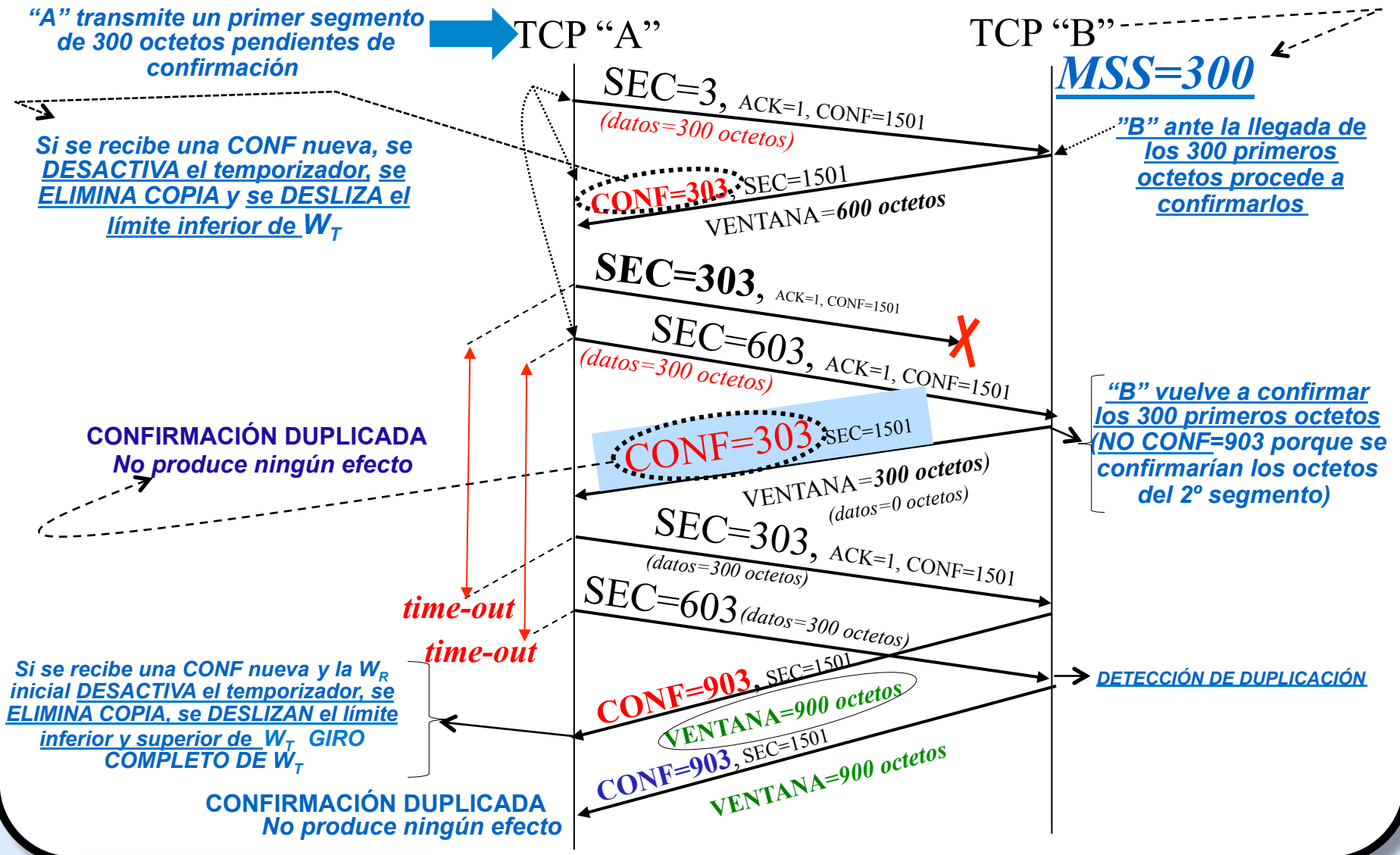
- **Fase de transferencia de datos (sin errores):**



Ejemplo de Transferencia de Datos con Errores

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

(pérdidas de paquetes IP) y con VENTANA en "A" de 300 octetos y en "B" de 900 octetos



Fase de Liberación de la Conexión

- *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*

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	-	***SACK PERMITTED*** ***(Fase de establecimiento)***
5	Variable	-	***SACK*** ***(Fase de transferencia de datos)***
8	10	Valor actual reloj emisor (4 octetos) + Respuesta Eco (4 octetos)	*Marca de Tiempo*
…	…	…	…

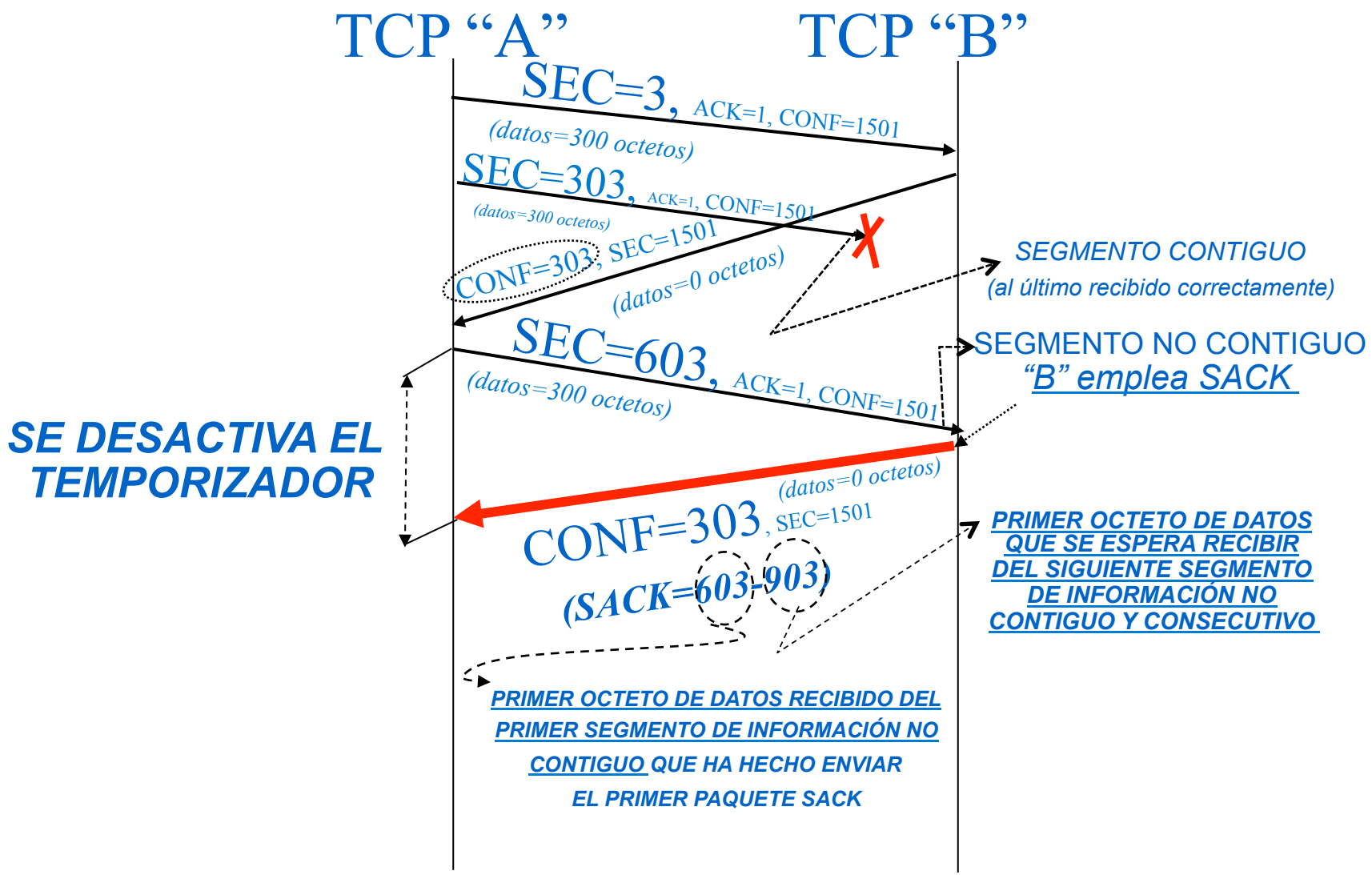
TCP SACK
TCP de Reconocimiento Selectivo *Selective Acknowledgment (SACK)*

- Definido en el *RFC-2018* y, posteriormente, extendido en el *RFC-2883*
- Se recomienda la opción SACK, especialmente, **cuando hay múltiples pérdidas de segmentos TCP**
  - **Para evitar innecesarias retransmisiones** de los octetos de datos de segmentos de información no contiguos que ya fueron entregados al receptor
  - Actualmente, es una opción por omisión en TCP

TCP SACK
TCP de Reconocimiento Selectivo *Selective Acknowledgment (SACK)*

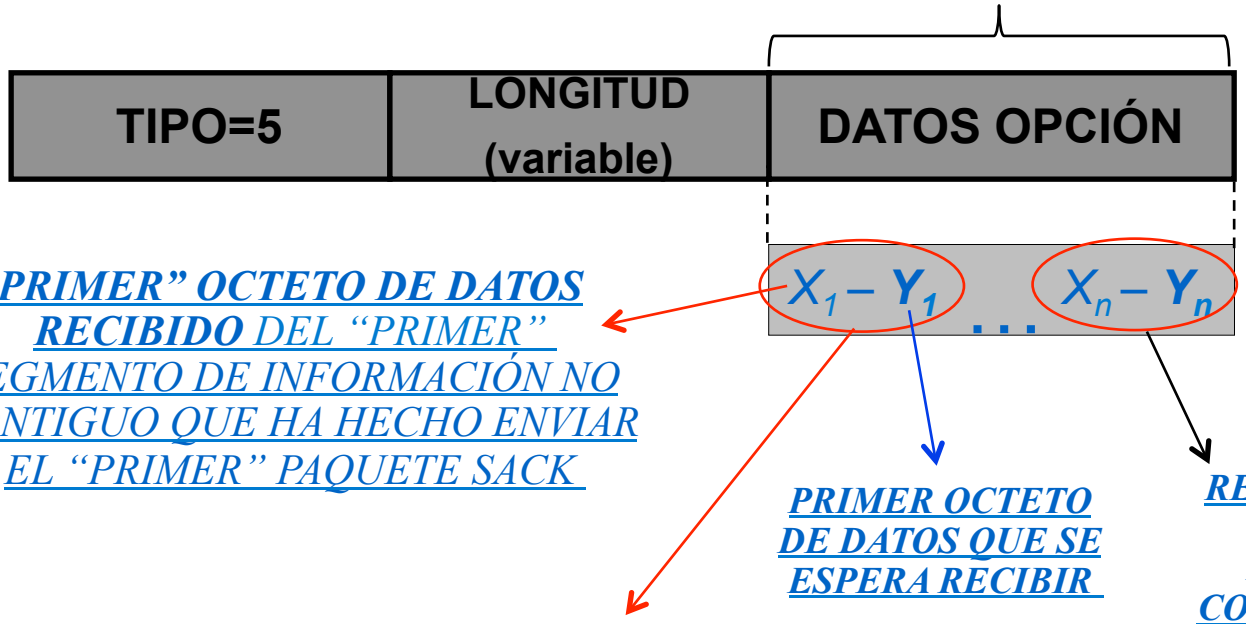
- ***Para poder hacer uso de la opción Tipo 5 SACK en fase de transferencia de datos hay que indicarlo previamente en fase de establecimiento de la conexión TCP***
  - Se indica mediante la **opción *Tipo 4 TCP SACK-PERMITTED*** en la fase de establecimiento de la conexión TCP en un **segmento *SYN = 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 un Paquete SACK

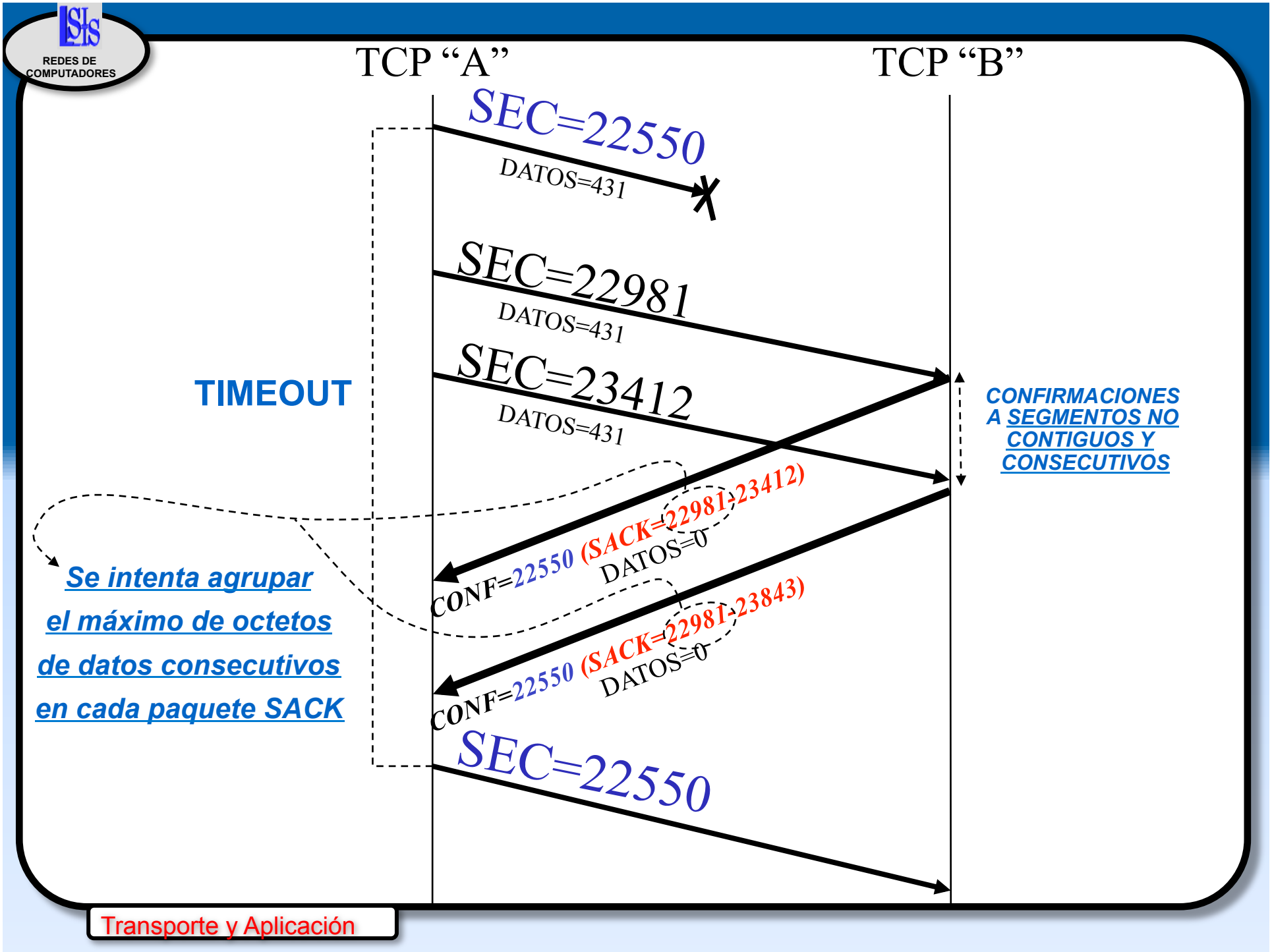


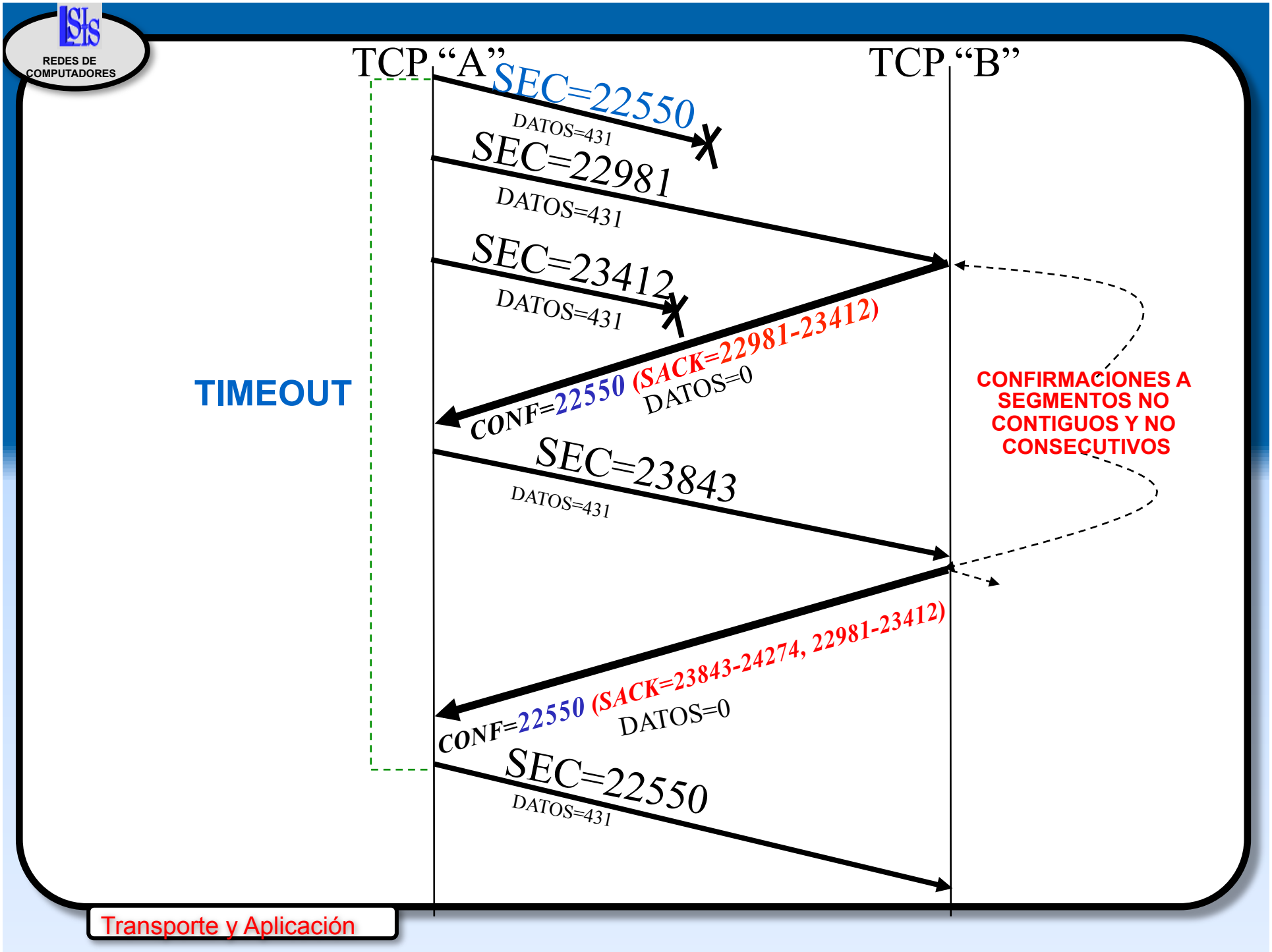
Formato de un Paquete SACK

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

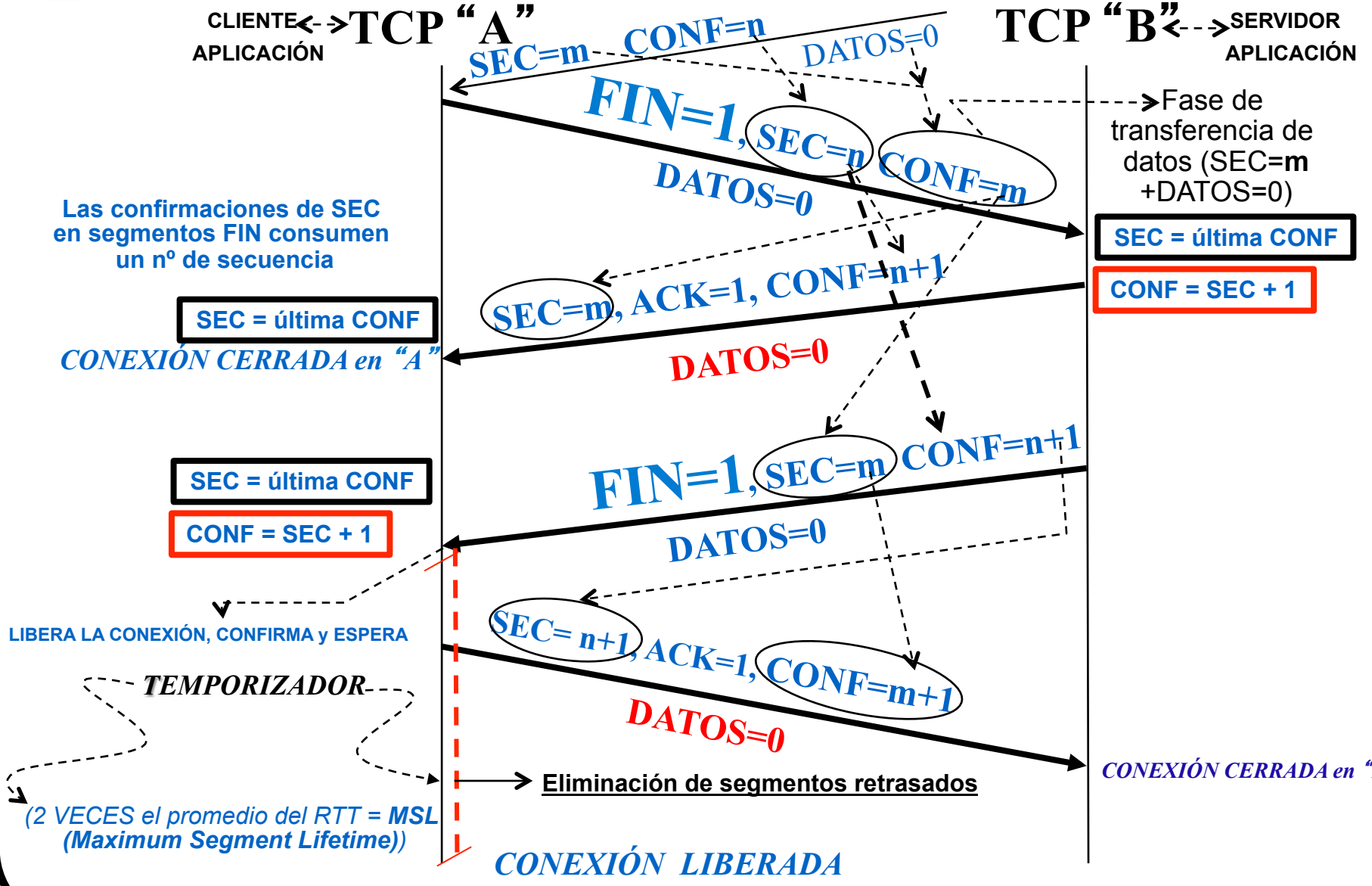


El bloque ($X_1 - Y_1$) debe informar del segmento no contiguo recibido más recientemente


TCP "A"
TCP "B"
SEC=22550
DATOS=431
SEC=22981
DATOS=431
SEC=23412
DATOS=431
TIMEOUT
CONFIRMACIONES A 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
SEC=22550


TCP "A"
TCP "B"
SEC=22550
DATOS=431
SEC=22981
DATOS=431
SEC=23412
DATOS=431
TIMEOUT
CONF=22550 (SACK=22981-23412)
DATOS=0
CONFIRMACIONES A SEGMENTOS NO CONTIGUOS Y NO CONSECUTIVOS
SEC=23843
DATOS=431
CONF=22550 (SACK=23843-24274, 22981-23412)
DATOS=0
SEC=22550
DATOS=431

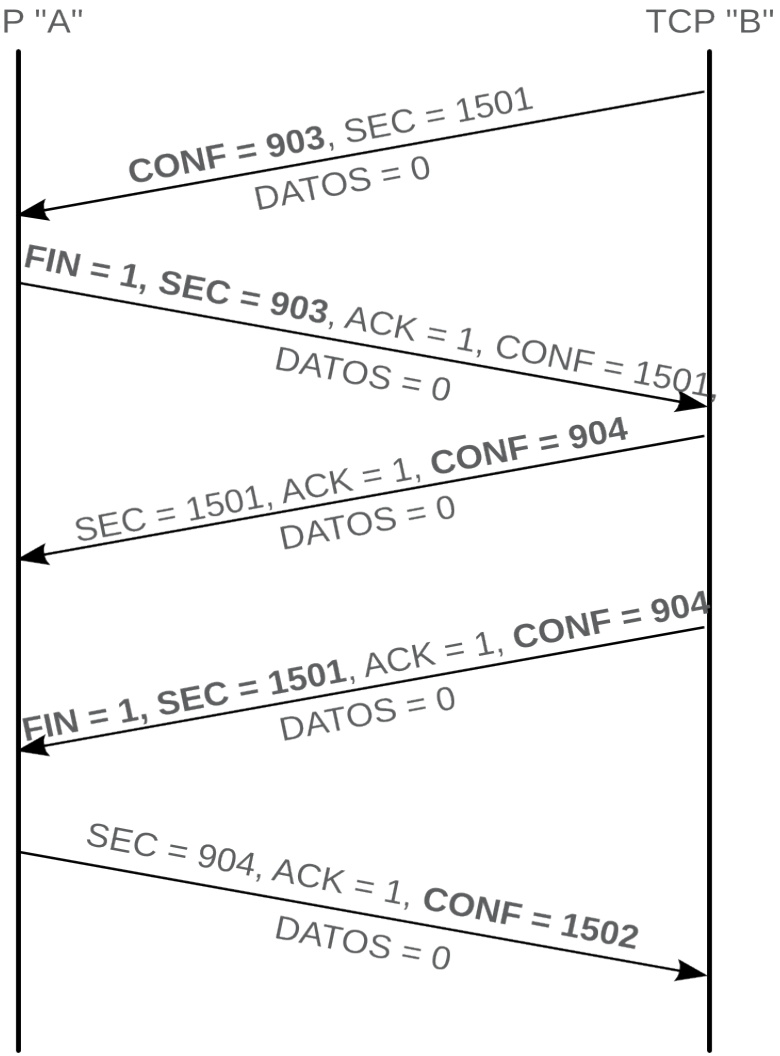
Fase de Liberación de la Conexión



Nivel de transporte

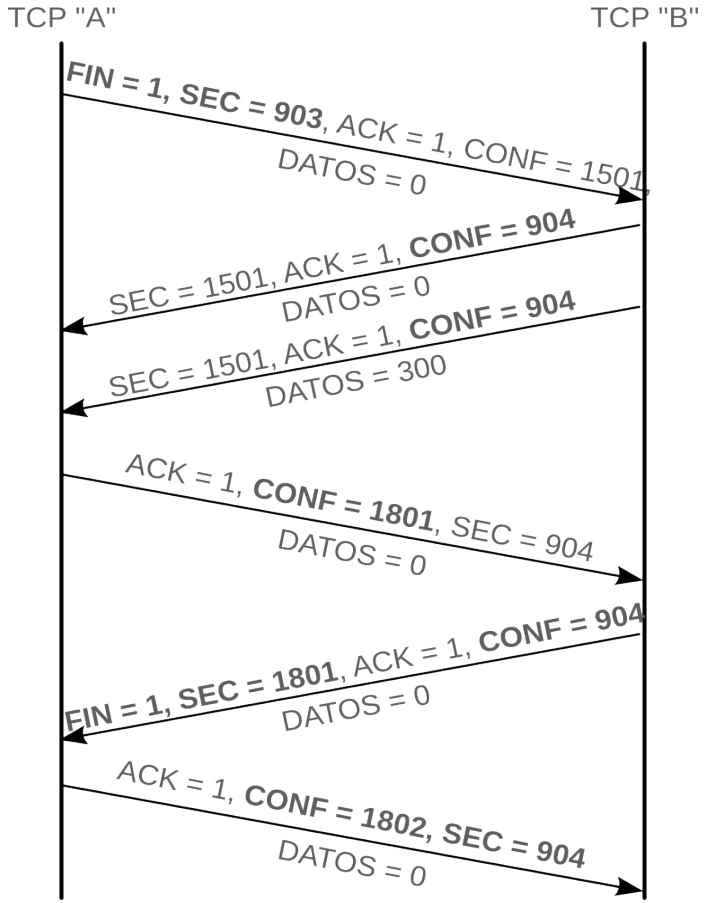
Protocolo TCP

- **Fase de liberación de la conexión TCP:**



Nivel de transporte
Protocolo TCP

- **Fase de liberación de la conexión TCP (con datos):**
  Conexión “cerrada” en “A” pero “abierta” en “B”

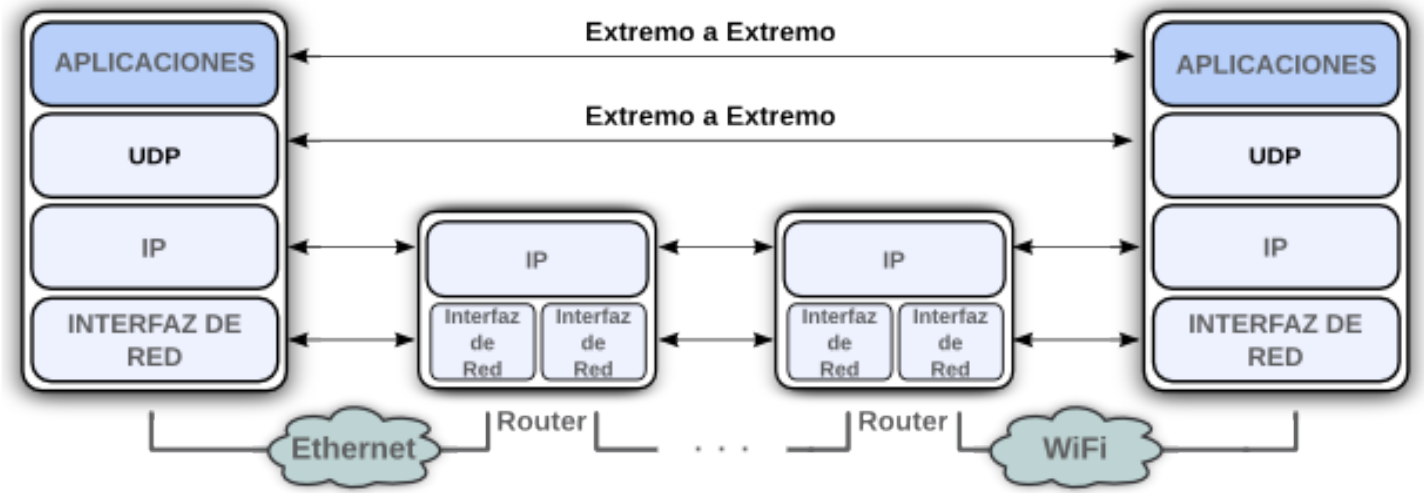


Nivel de transporte
Protocolo TCP

- **Problemática de la fiabilidad de TCP**:
  - Si se desea rapidez en el transporte, TCP retarda dicho transporte porque ofrece un servicio orientado a conexión (tres fases) y fiable (mecanismos de detección y corrección de errores)
  - Hay aplicaciones que no toleran el retardo extremo a extremo producido por los ACKs, los temporizadores, las retransmisiones y controles de TCP
    - Además, el control de congestión TCP reduce la tase de envío (*throughput*) en el emisor
  - 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)
    - Tráfico no interactivo en tiempo real (*streaming* de audio y vídeo)

Nivel de transporte
Protocolo UDP

- **Protocolo UDP** (*User Datagram Protocol*) RFC-768, STD 0006:
  - Transporte no fiable pero rápido de los mensajes de aplicación encapsulados en datagramas UDP
    - No orientado a conexión y no fiable
    - Sin controles extremo a extremo
  - Las unidades de datos del protocolo UDP se denominan **datagramas UDP**



Nivel de transporte
Protocolo UDP

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

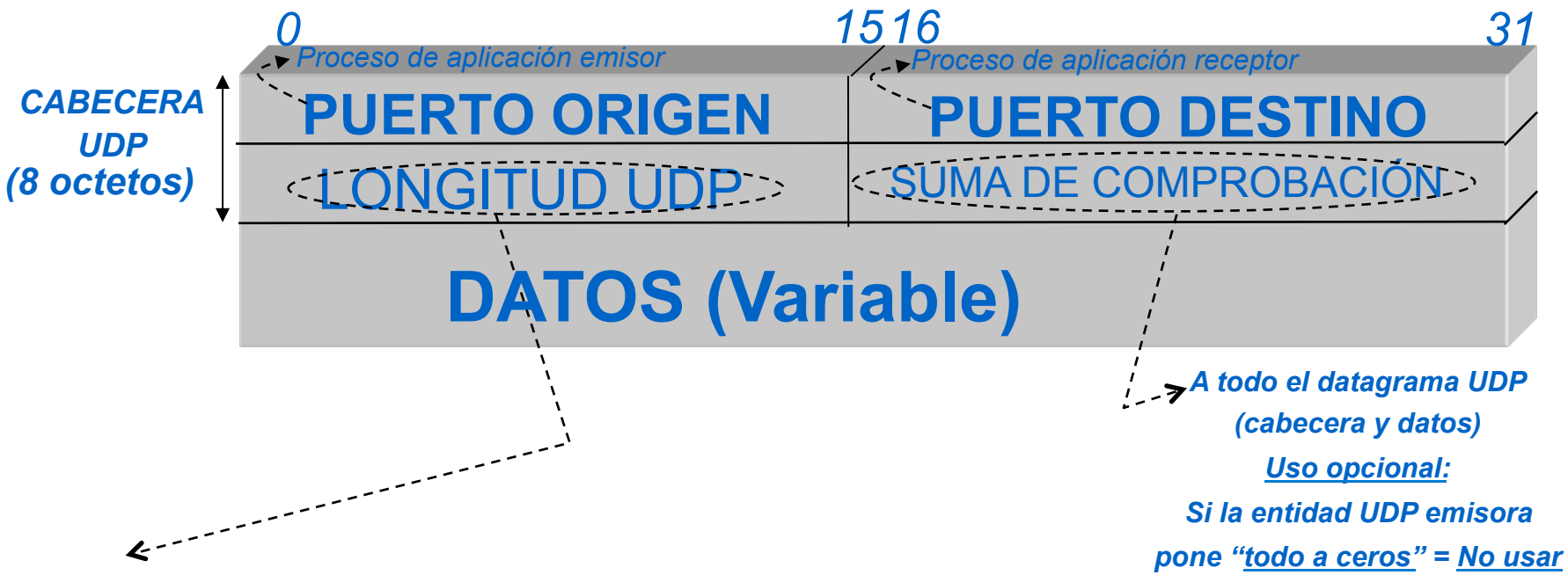
Nivel de transporte
Protocolo UDP

- **Servicios**:
  - No ofrece servicio de flujo de octetos *(byte-stream)*. La entidad del nivel de aplicación debe pasar los datos bien delimitados
  - **No orientado a conexión 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/Demultiplexado**: a través de los números de puerto
  - **Full-duplex**: transferencia bidireccional y simultánea

Nivel de transporte
Protocolo UDP

- **Formato de una datagrama UDP**:

Formato de un Datagrama UDP



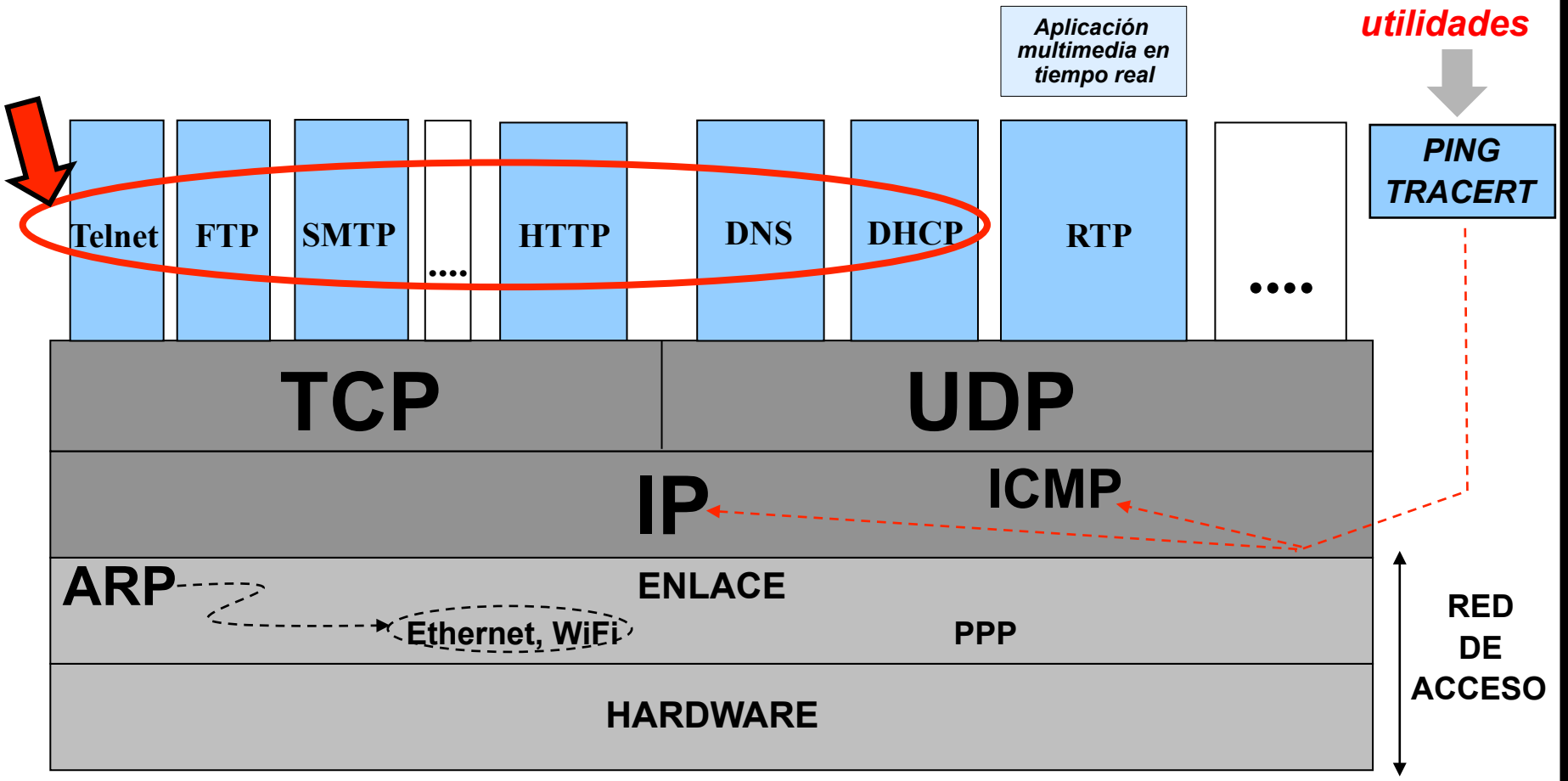
Longitud en octetos del datagrama UDP (cabecera + datos) =

= 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

Un Ejemplo de Arquitectura TCP/IP



Distinción entre Protocolos de Comunicaciones y HERRAMIENTAS

APLICACIONES montadas sobre TCP o UDP

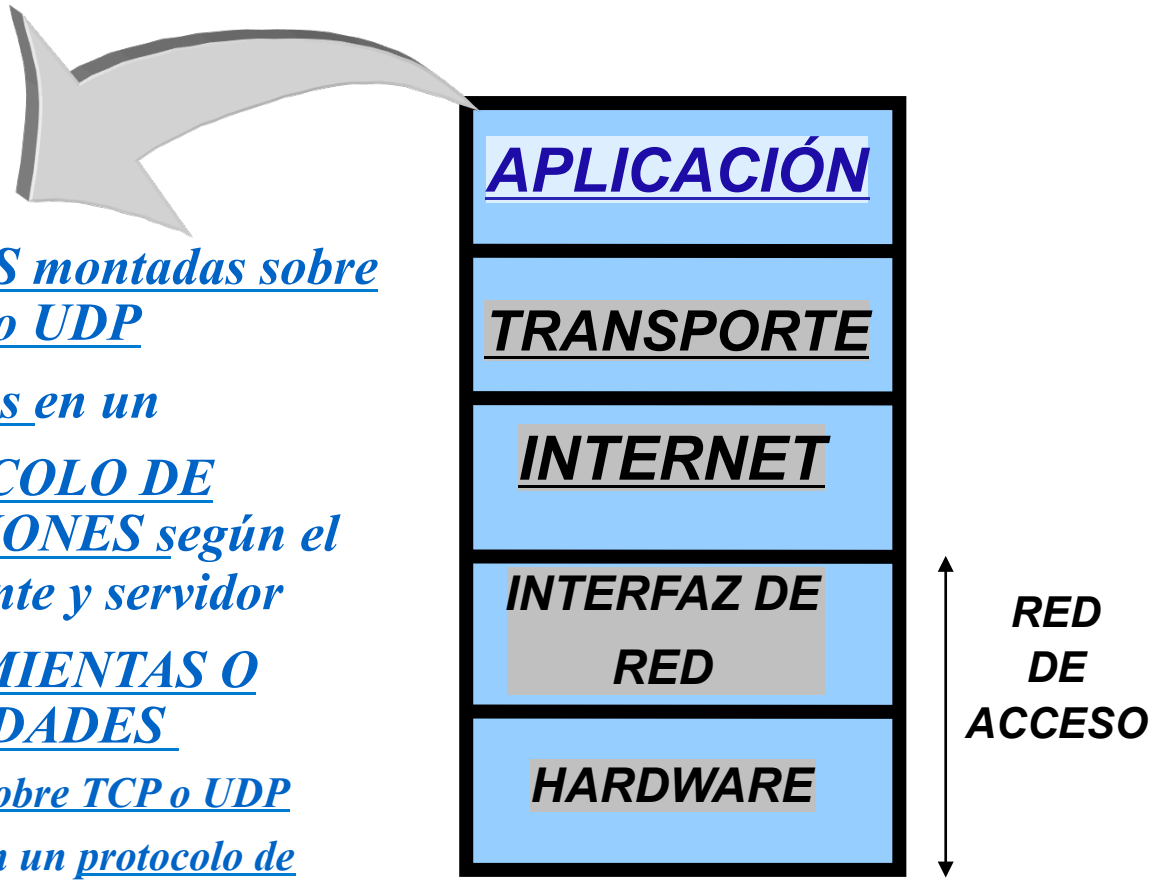
basadas en un

PROTOCOLO DE COMUNICACIONES según el modelo cliente y servidor

y HERRAMIENTAS O UTILIDADES

NO montadas sobre TCP o UDP

y NO basadas en un protocolo de comunicaciones del nivel de aplicación



APLICACIONES TCP/IP

- **Aplicaciones que *siguen el modelo cliente y servidor, es decir, basadas en un Protocolo de Aplicación entre dos Entidades Pares (Entidad Cliente y Entidad Servidora* con *Números de Puerto TCP o UDP***
  - *TELNET, FTP, SMTP, HTTP (Web), DNS, DHCP, ...*
- **Herramientas o utilidades que *NO siguen el modelo cliente y servidor , es decir,* NO basadas en un *Protocolo de Aplicación (No hay entidades pares y NO disponen de números de puerto TCP o UDP)***
  - *PING, IPCONFIG (IFCONFIG en Unix), TRACERT (TRACEROUTE en Linux), NSLOOKUP, ...*
    - *ARP es una utilidad de aplicación que se basa en la información obtenida a través del protocolo ARP del nivel de enlace*

TCP ofrece un SERVICIO ORIENTADO A CONEXIÓN y FIABLE a los Procesos del Nivel de Aplicación

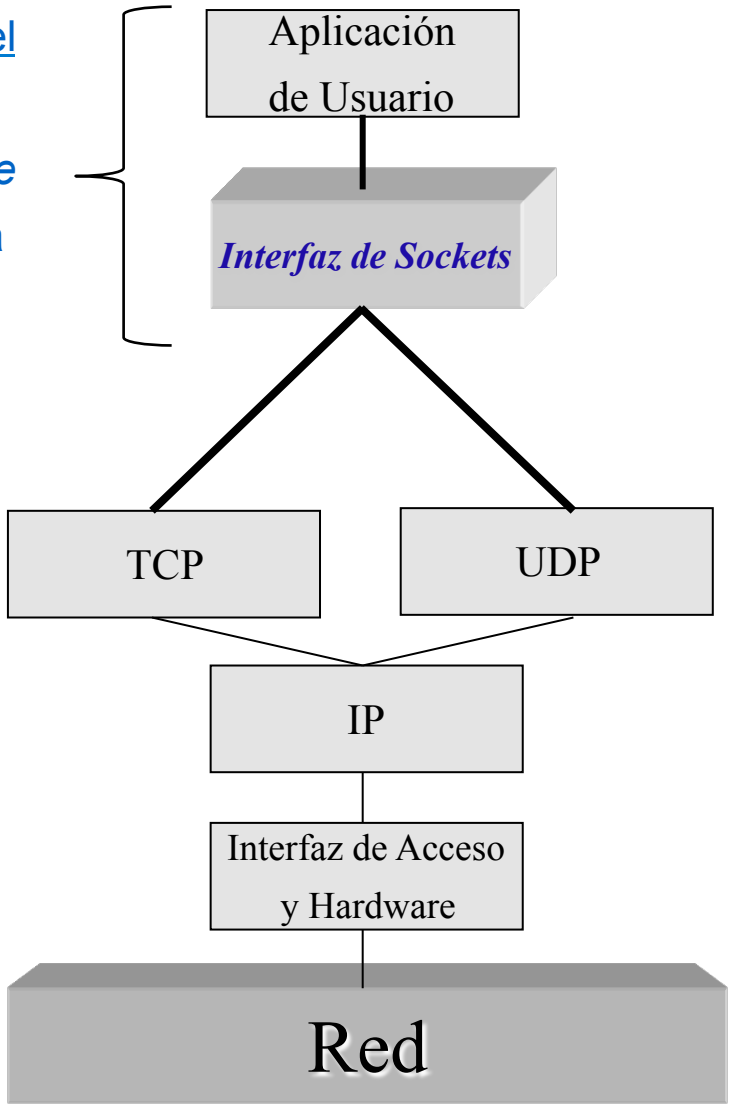
- Un servicio orientado a conexión dispone de TRES FASES
  1. **ESTABLECIMIENTO DE LA CONEXIÓN**
  2. **TRASFERENCIA DE DATOS**
  3. **LIBERACIÓN DE LA CONEXIÓN**
- *¿Cómo accede la entidad de aplicación al servicio TCP?*
  - Haciendo uso tanto en el *código del cliente* como en el *código del servidor* de unas *funciones de comunicaciones estándares* pertenencientes a un *API de programación de aplicaciones en red* (API de Red o API de Sockets) denominado *INTERFAZ DE SOCKETS*

INTERFAZ DE SOCKETS

Universidad de Berkeley

El programador tiene que trabajar a bajo nivel y llamar directamente a las funciones de comunicaciones del *API de programación* proporcionado por un sistema basado en un interfaz de sockets

- ***API** basado en un conjunto de directorios o carpetas o librerías o bibliotecas de funciones de comunicaciones, ficheros cabecera y estructuras en lenguaje C que hay que llamar en el código del lado cliente y servidor para:*
  - ***Obtener PREVIAMENTE un socket cliente para TCP que identifique al proceso cliente y un socket servidor para TCP que identifique al proceso servidor***
  - ***FASE DE ESTABLECIMIENTO DE LA CONEXIÓN:** Establecer la conexión a petición del cliente entre los dos sockets a través de la función "connect"*
  - ***FASE DE TRANSFERENCIA DE DATOS:** Enviar y recibir datos entre los dos sockets a través de las funciones "write" y "read"*
  - ***FASE DE LIBERACIÓN DE LA CONEXIÓN:** "Cerrar la conexión entre los dos sockets a través de la función "close"*



CÓDIGO CONCEPTUAL DE UN CLIENTE TCP

socket (*familia de protocolos*, *tipo de socket*, *tipo del protocolo de comunicaciones*)

SERVICIO DESEADO DE TRANSPORTE o IP

AF_INET	**SOCK_STREAM**	**IP_PROTO_TCP**
AF_INET6	SOCK_DGRAM	IP_PROTO_UDP
	SOCK_RAW	IP_PROTO_IP

La función socket PERMITE ENGANCHAR EL CÓDIGO DE LA APLICACIÓN CON TCP y devolver un descriptor tipo entero del socket que, luego, se usará para representar al punto extremo de comunicación en posteriores llamadas al sistema

```
. . .
int descriptor
descriptor = socket(AF_INET, SOCK_STREAM, IP_PROTO_TCP)
connect(descriptor, dir_IP_remota, nº_puerto_remoto)
```

= ESTABLECIMIENTO DE LA CONEXIÓN

```
write(descriptor, …)
```

= TRANSFERENCIA DE DATOS: ENVIAR

```
read(descriptor,..)
```

= TRANSFERENCIA DE DATOS: RECIBIR

```
close(descriptor)
```

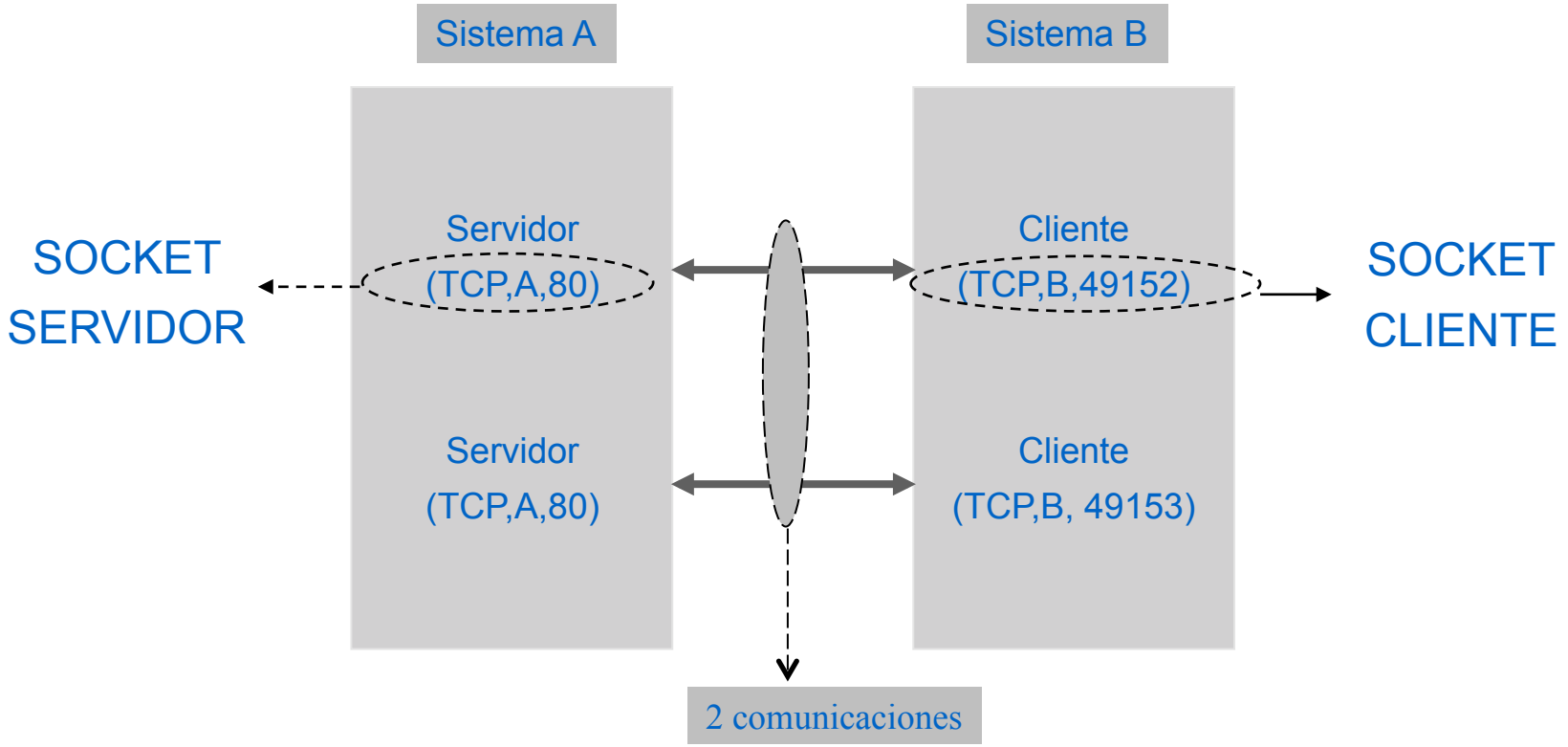
= LIBERACIÓN DE LA CONEXIÓN

CÓDIGO CLIENTE

RECORDATORIO

UNA COMUNICACIÓN ENTRE UN PROCESO CLIENTE Y OTRO SERVIDOR QUEDA PLENAMENTE DEFINIDA CUANDO SE CONOCEN SUS EXTREMOS

SOCKET: *PUNTO DE ACCESO o EXTREMO DE UNA COMUNICACIÓN*



UNA COMUNICACIÓN ENTRE UN PROCESO CLIENTE Y OTRO SERVIDOR QUEDA PLENAMENTE DEFINIDA POR UNA PAREJA DE SOCKETS

RECORDATORIO

Para TCP o UDP, la combinación de una dirección IP y un número de puerto identifica un punto de conexión del nivel de transporte o "socket"

Ejemplo de socket: **138.10.1.16** : **80**

Dirección IP

Número de Puerto

Socket
(Para TCP o UDP)

UDP ofrece un SERVICIO NO ORIENTADO A CONEXIÓN y NO FIABLE a los Procesos del Nivel de Aplicación

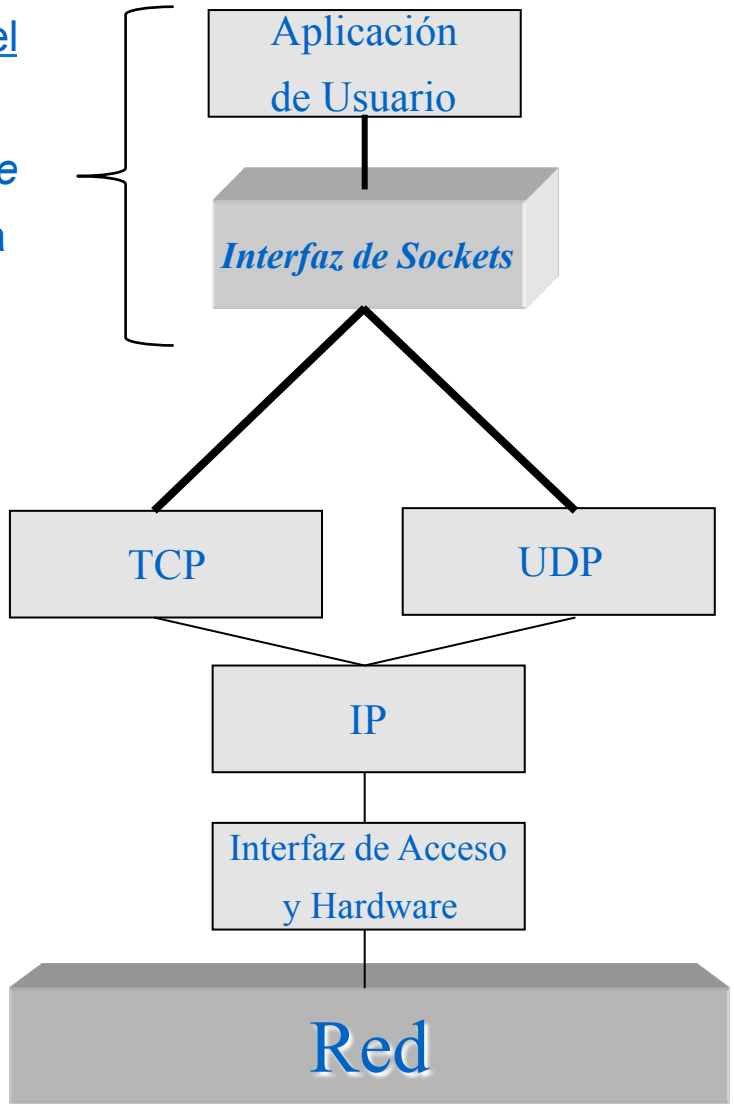
- Un servicio no orientado a conexión dispone de UNA FASE
  1. **TRASFERENCIA DE DATOS**

- *¿Cómo accede la entidad de aplicación al servicio UDP?*
  - Haciendo uso tanto en el *código del cliente* como en el *código del servidor* de unas *funciones de comunicaciones estándares* pertenencientes a un *API de programación de aplicaciones en red* (API de Red o API de Sockets) denominado *INTERFAZ DE SOCKETS*

INTERFAZ DE SOCKETS
Universidad de Berkeley

El programador tiene que trabajar a bajo nivel y llamar directamente a las funciones de comunicaciones del *API de programación* proporcionado por un sistema basado en un interfaz de sockets

- *API basado en un conjunto de directorios o carpetas o librerías o bibliotecas de funciones de comunicaciones, ficheros cabecera y estructuras que hay que llamar en el código del lado cliente y servidor para:*
  - *Obtener PREVIAMENTE un socket cliente para UDP que identifique al proceso cliente y un socket servidor para UDP que identifique al proceso servidor*
  - *FASE DE TRANSFERENCIA DE DATOS: Enviar y recibir datos entre los dos sockets a través de las funciones "sendto" y "recvfrom"*



CÓDIGO CONCEPTUAL DE UN CLIENTE UDP

socket (*familia de protocolos*, *tipo de socket*, *tipo del protocolo de comunicaciones*)

SERVICIO DESEADO DE TRANSPORTE o IP

AF_INET	SOCK_STREAM	IP_PROTO_TCP
AF_INET6	**SOCK_DGRAM**	**IP_PROTO_UDP**
	SOCK_RAW	IP_PROTO_IP

La función socket PERMITE ENGANCHAR EL CÓDIGO DE LA APLICACIÓN CON TCP *y devolver un descriptor tipo entero del socket que, luego, se usará para representar al punto extremo de comunicación en posteriores llamadas al sistema*

. . .

int descriptor

descriptor = ***socket(AF_INET, SOCK_DGRAM, IP_PROTO_UDP)***

sendto*(descriptor, …)*

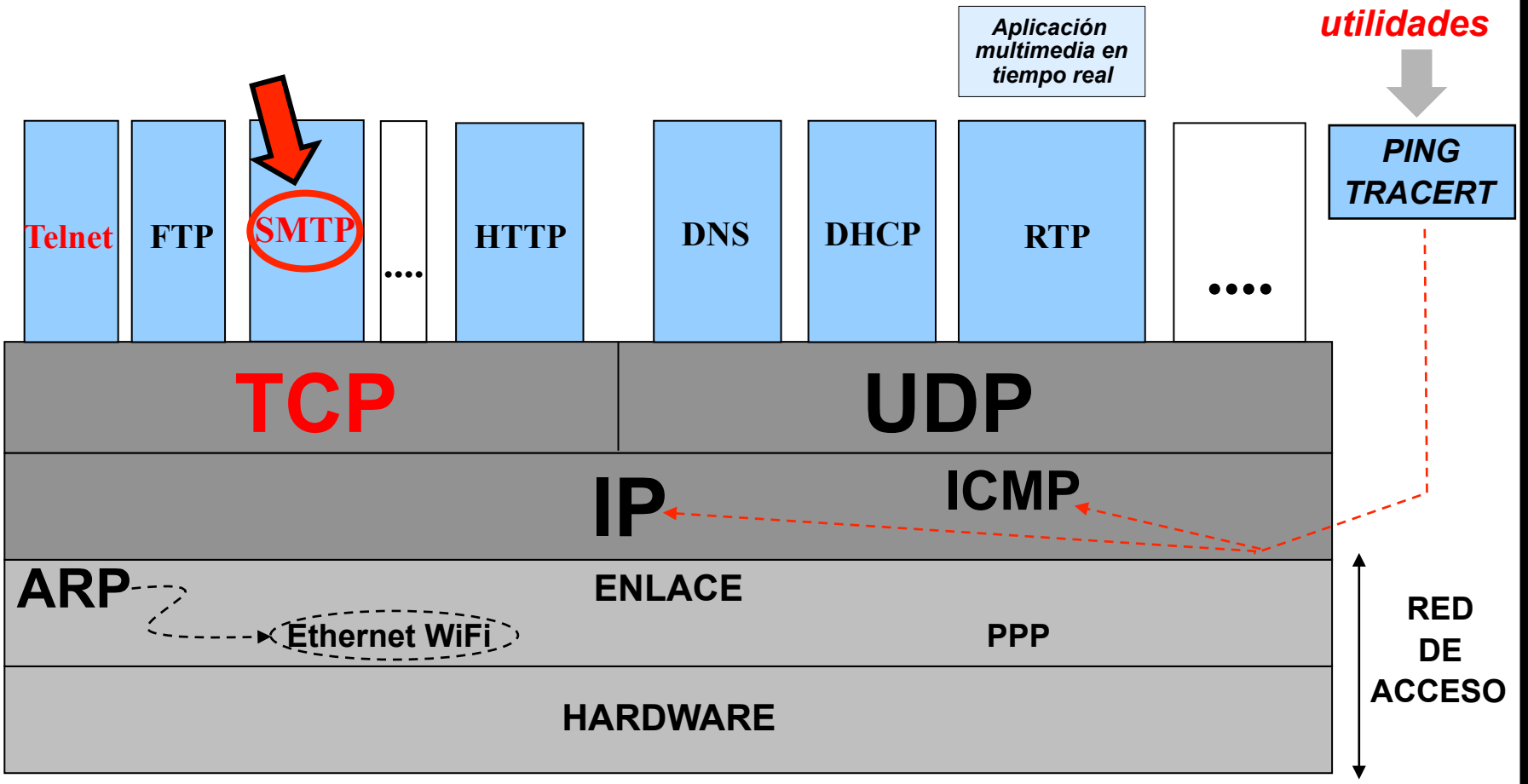
= TRANSFERENCIA DE DATOS: ENVIAR

recvfrom*(descriptor,..)*

= TRANSFERENCIA DE DATOS: RECIBIR

CÓDIGO CLIENTE

Protocolos de Aplicación sobre TCP



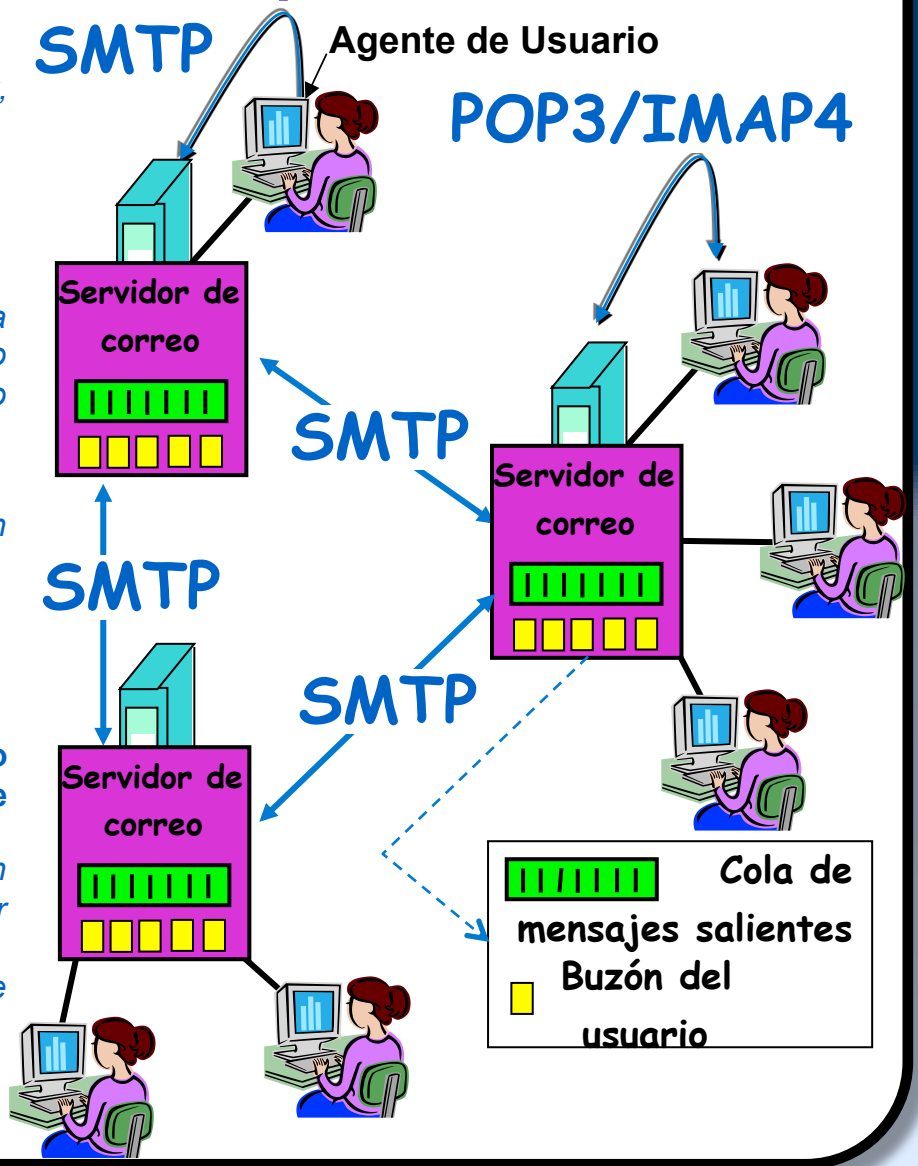
APLICACIÓN DE CORREO ELECTRÓNICO TCP/IP

Tres Componentes Principales

1. **AGENTE DE USUARIO** *(p.ej., Microsoft Office Outlook, Kmail y Evolution de linux, etc.)*
   - *Sistema de correo en la máquina del usuario*
   - *CLIENTE DE CORREO SMTP*
   - *Editor de texto*
   - *Codificador/decodificador o CODEC MIME*
   - *Protocolo 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** del usuario
   - *Sistema de correo en la máquina de la organización del usuario o en la red IP de su ISP*
   - *SERVIDOR DE CORREO SMTP*
   - *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)**
   1. *Enviar correo desde el cliente de correo SMTP de un agente de usario a su servidor de correo SMTP por omisión*
   2. *Enviar correo desde el servidor origen o del remitente al servidor destino (buzón del destinatario)*

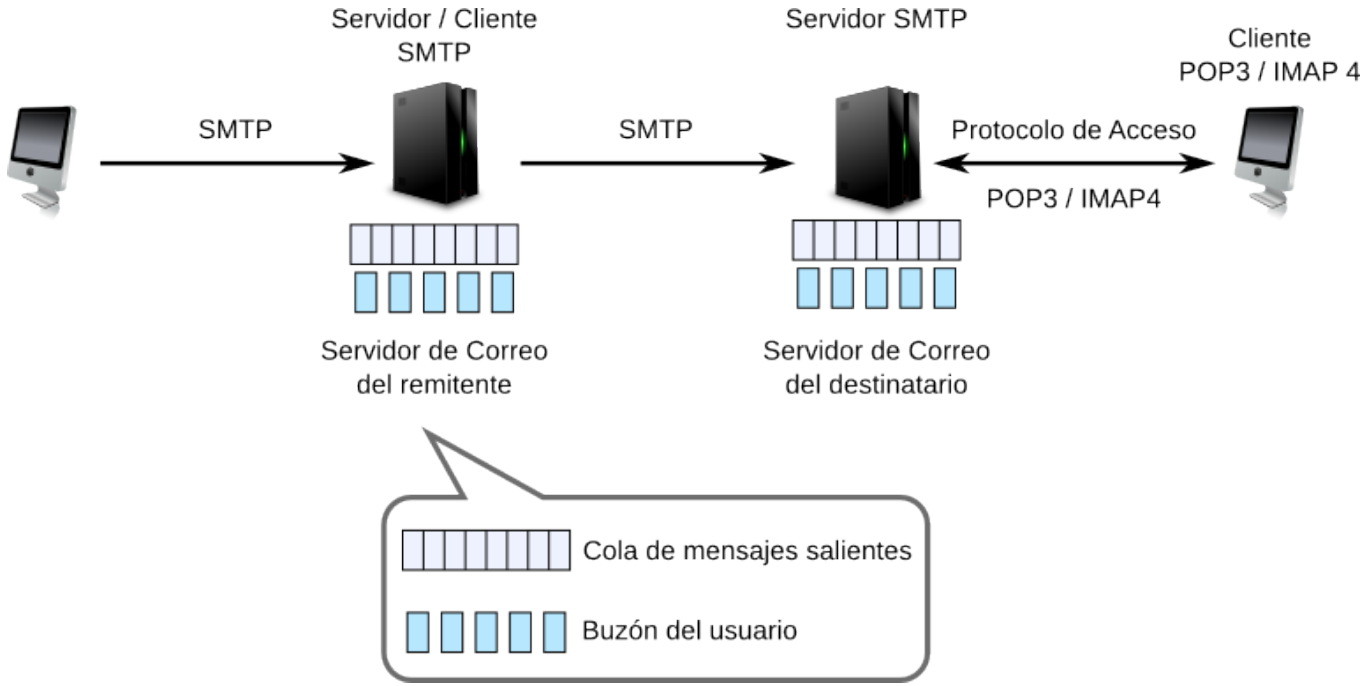


Nivel de aplicación
Servicio de correo electrónico

- Componentes principales:
  - **Agente de usuario**: Sistema de correo en la máquina del usuario
    - Cliente de correo SMTP
    - Editor de texto
    - Codificador/decodificador (CODEC MIME)
    - Protocolo 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
  - **Servidor de correo**: Sistema de correo en la máquina de la organización del usuario
    - Servidor de correo SMTP
    - Buzones de los usuarios
    - *Buffers* de los mensajes salientes
  - **Protocolo de envío SMTP**:
    - Enviar correo desde el cliente de correo SMTP de un agente de usuario a su servidor de correo SMTP por omisión
    - Enviar correo desde el servidor origen o del remitente al servidor destino (buzón del destinatario)

Nivel de aplicación
Servicio de correo electrónico

- **Protocolo SMTP** (*Simple Mail Transfer Protocol*) RFC-5321:
  - Uno de los primeros protocolos TCP/IP
  - Usado entre el cliente SMTP del agente de usuario y su servidor SMTP
  - Usado entre dos servidores de correo para enviar mensajes (servidor de correo del remitente y servidor de correo del destinatario)



REDES DE COMPUTADORES

4 Pasos Básicos en el Envió del Correo Electrónico

usuario@dominio

De *javier*@***smtp.origen.com***

Para *miguel*@***smtp.destino.com***

javier

Cliente SMTP*(remitente)*

Destino: ***miguel@smtp.destino.com***

Nombre de dominio del servidor destinatario de correo

miguel

Cliente SMTP

1

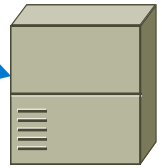
AUTENTICACIÓN

Nombre usuario

contraseña

(CONEXIÓN AUTENTICADA)

2 **Servidor DNS**



smtp.destino.com---220.10.1.2

3 ***smtp***

miguel@***smtp.destino.com***

220.10.1.2

Servidor SMTP

Buzón de javier

Cliente SMTP

smtp.origen.com

Servidor de correo origen

4

Servidor SMTP

smtp.destino.com

Servidor de correo destino

Buzón de miguel

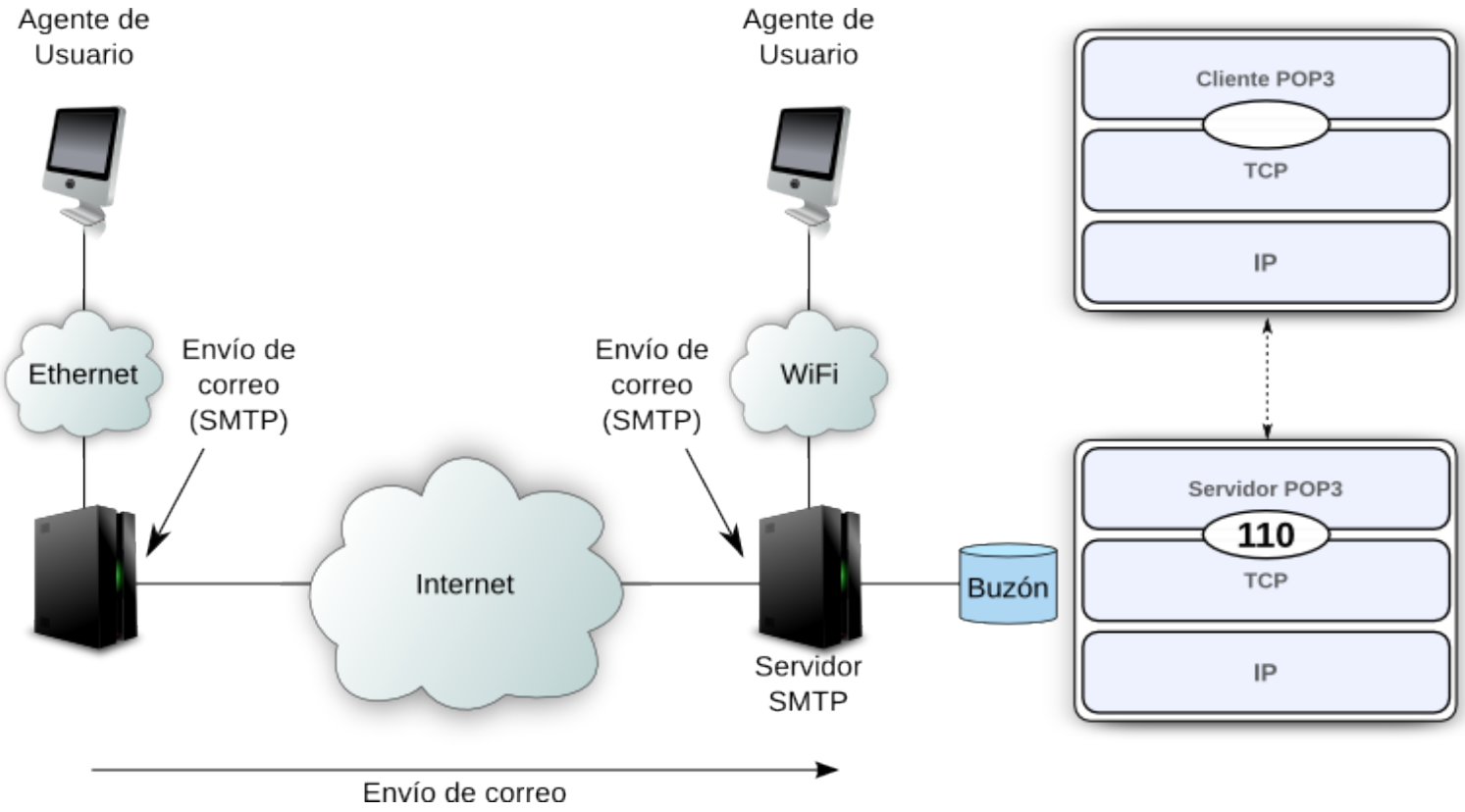
smtp

Transporte y Aplicación

Nivel de aplicación

Servicio de correo electrónico

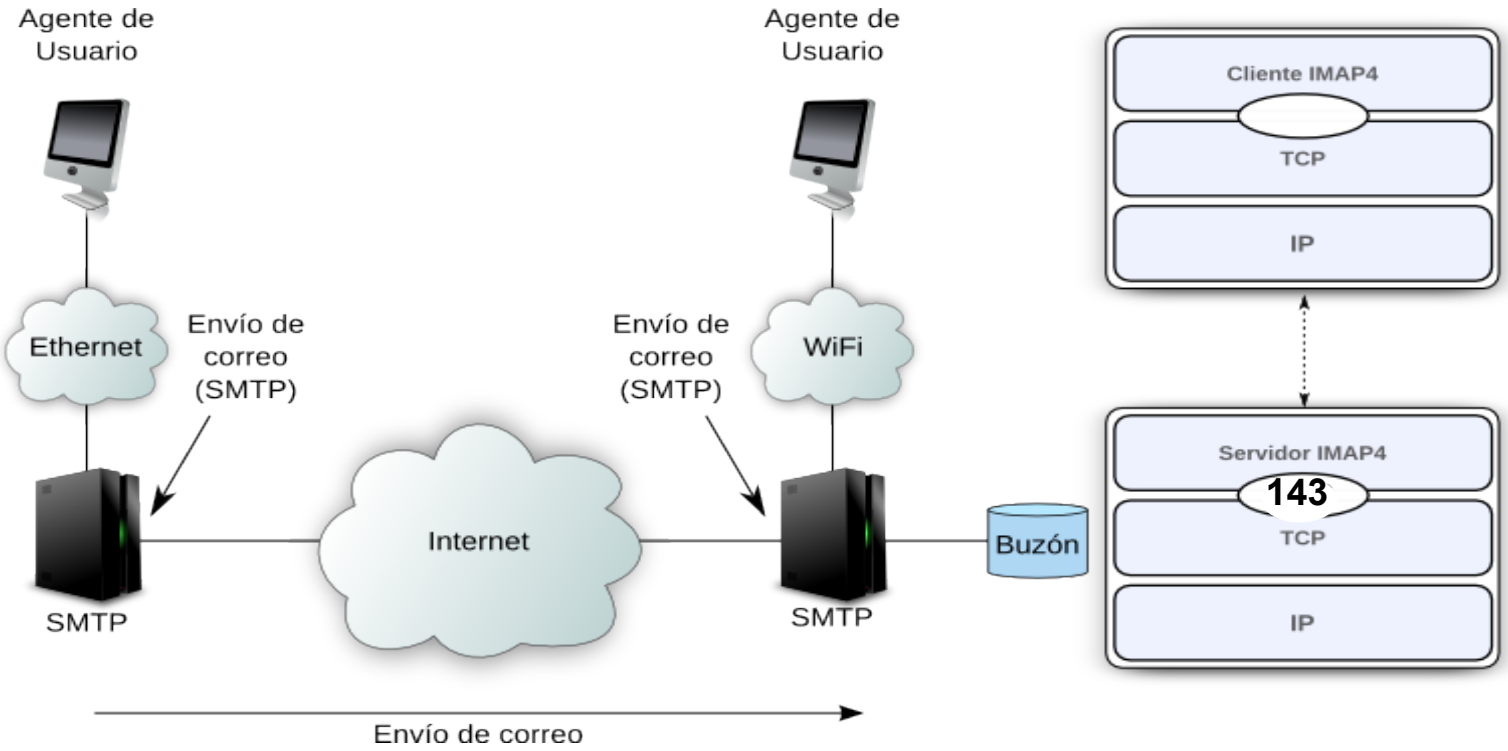
- **Protocolo POP3** (*Post Office Protocol - Version 3*) RFC-1939:
  - Proporciona un servicio de recogida de todos los mensajes existentes en el buzón de correo del usuario en su servidor de correo a un directorio de un disco duro de su máquina local



Nivel de aplicación

Servicio de correo electrónico

- **Protocolo IMAPv4**(*Internet Message Access Protocol-Version 4)*RFC-3501
  - 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
    - Eliminar y clasificar correo en carpetas en el disco duro del servidor de correo
    - Copiar o mover mensajes desde su buzón hasta el disco duro de su ordenador



Nivel de aplicación
Servicio de correo electrónico

- **Protocolo IMAPv4**:
  - Permite al usuario clasificar, eliminar y distribuir su correo en carpetas en el disco duro de la máquina servidora de correo
  - Permite al usuario copiar o mover mensajes, previamente seleccionados, desde su buzón hasta el disco duro de su ordenador, distribuyéndolos en carpetas locales
  - El correo no se lee a través de la red.
    - IMAP4 se trae las cabeceras de los mensajes de forma temporal a un determinado directorio local en la máquina del usuario
    - Posteriormente, se trae el texto del mensaje al seleccionar previamente la cabecera del mensaje en cuestión

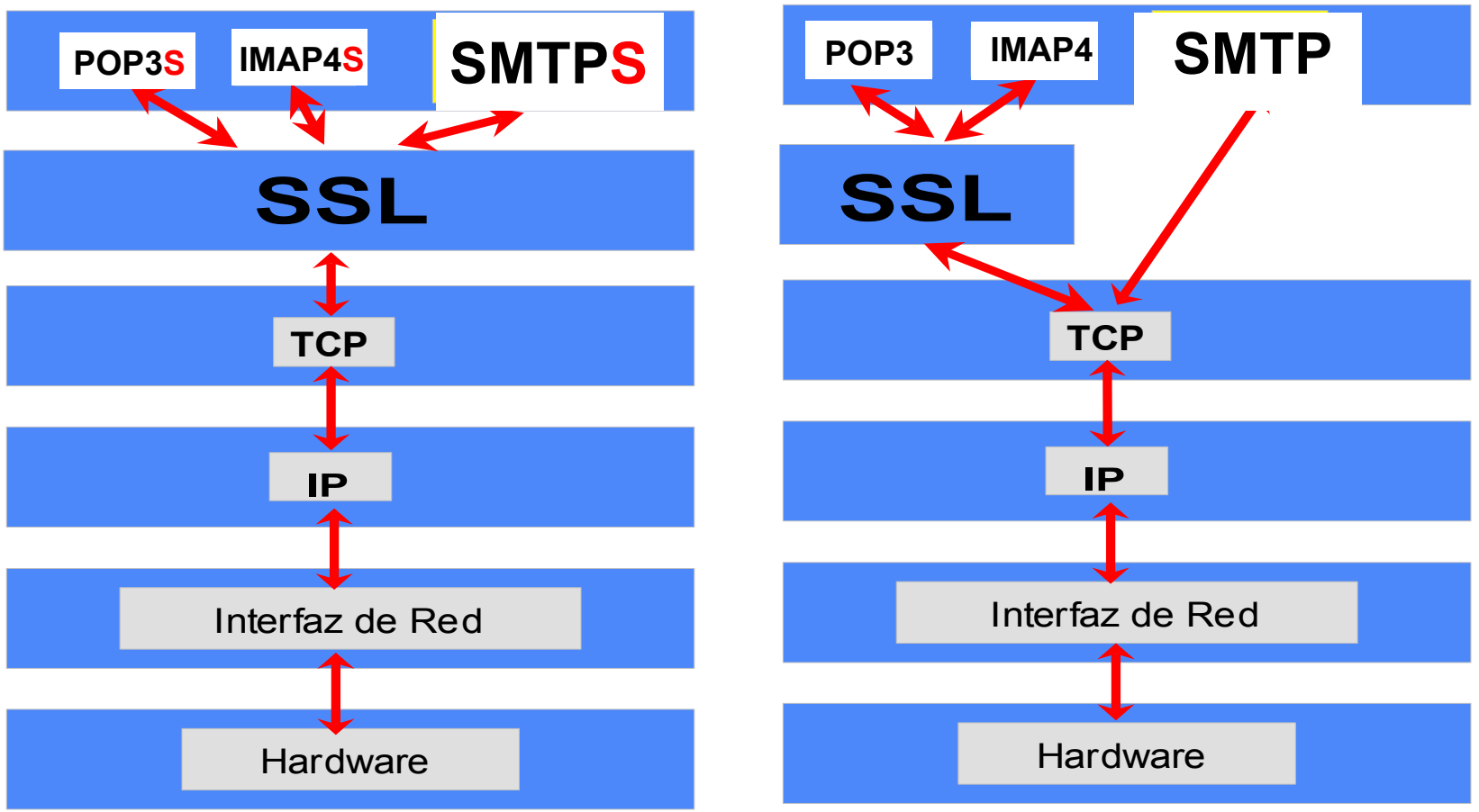
Nivel de aplicación

Servicio de correo electrónico: Puertos Servidores

- 25
  - *Permisivo*
  - *Nombre de usuario y contraseña visible (autenticación no segura)*
- **465**
  - *Seguro*
    - *Cifrado del nombre de usuario y contraseña (autenticación segura)*
    - *Cifrado de todo el correo enviado del cliente SMTP al servidor SMTP*
    - *Cifrado de todo el correo recibido por el servidor SMTP hacia el cliente*

Nivel de aplicación

Servicio de correo electrónico



REDES DE COMPUTADORES

Un Correo Electrónico Seguro vía SSL

(junto a Filtros de direcciones, Filtros antispam y Listas negras)

AUTENTICACIÓN SEGURA, POR SSL, PARA EL SERVIDOR DE SALIDA SMTP

Nombre usuario

contraseña

(CONEXIÓN AUTENTICADA SEGURA)

465

SSL

CIFRADOS

AUTENTICACIÓN SEGURA, POR SSL, PARA EL SERVIDOR DE ENTRADA IMAP4/POP3

Nombre usuario

contraseña

SSL

CON FIREWALL

Se pueden filtrar direcciones IP de clientes o no para el nº de puerto 465

(eliminar el 143) ***imap4: 993***

(eliminar el 110) ***pop3: 995***

•FILTROS ANTISPAM

•LISTAS NEGRAS

smtp

25

smtp

25

Buzón

smtp.origen.com

•FILTROS ANTISPAM

•LISTAS NEGRAS

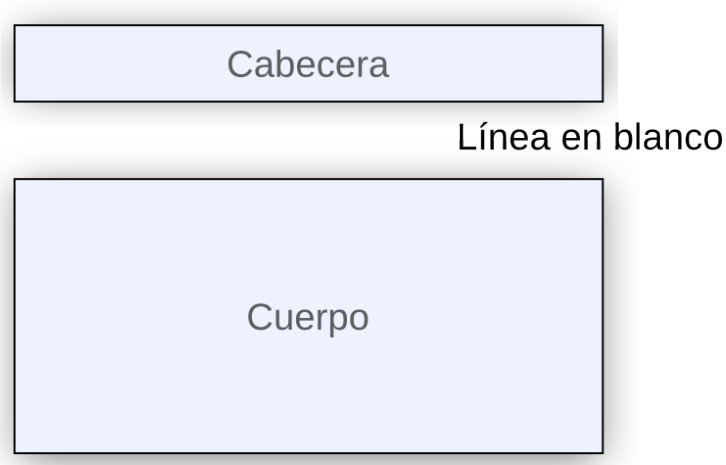
mx.destino.com

SERVIDOR DE INTERCAMBIO DE CORREO ELECTRÓNICO

smtp.destino.com

Nivel de aplicación
Servicio de correo electrónico

- **Formato de los mensajes SMTP** RFC-822 STD-0011:
  - Estándar para el formato de mensajes de texto. Define la **cabecera** (*To*:, *Cc*:, *Bcc*:, *From*:, *Subject*:, ...), una línea en blanco y el **cuerpo** del mensaje en formato ASCII de 7 bits (español, francés, ..., no soportados)



  - Para poder enviar mensajes NO ASCII (acentos, diéresis, ficheros JPG, MP3, MPEG-4, ...) en el cuerpo de un mensaje de correo se emplea el formato MIME

Nivel de aplicación
Servicio de correo electrónico

- **MIME** (*Multipurpose Internet Mail Extensions*) RFC-2045:
  - Permite el uso del formato RFC-822 pero agregando una cabecera al cuerpo del mensaje (declarando el contenido tipo MIME) y definiendo reglas de codificación para los mensajes no ASCII
  - Transforma los datos no ASCII en formato ASCII (codificador MIME en el agente de usuario emisor) y viceversa (decodificador MIME en el agente de usuario receptor)
  - MIME usa, entre otros, el sistema de codificación base64 o radix 64 (3 bloques de 8 bits = 4 caracteres de 6 bits) utilizando los caracteres ASCII de 7 bits o US-ASCII o (no acentuados = caracteres “A-Z, a-z,0-9, +/”)
  - Los ficheros incluidos en el mensaje (*attachments*) se traducen a MIME en el cuerpo del mensaje
  - Toda aplicación de correo electrónico y navegador de Internet acepta el base64 de MIME

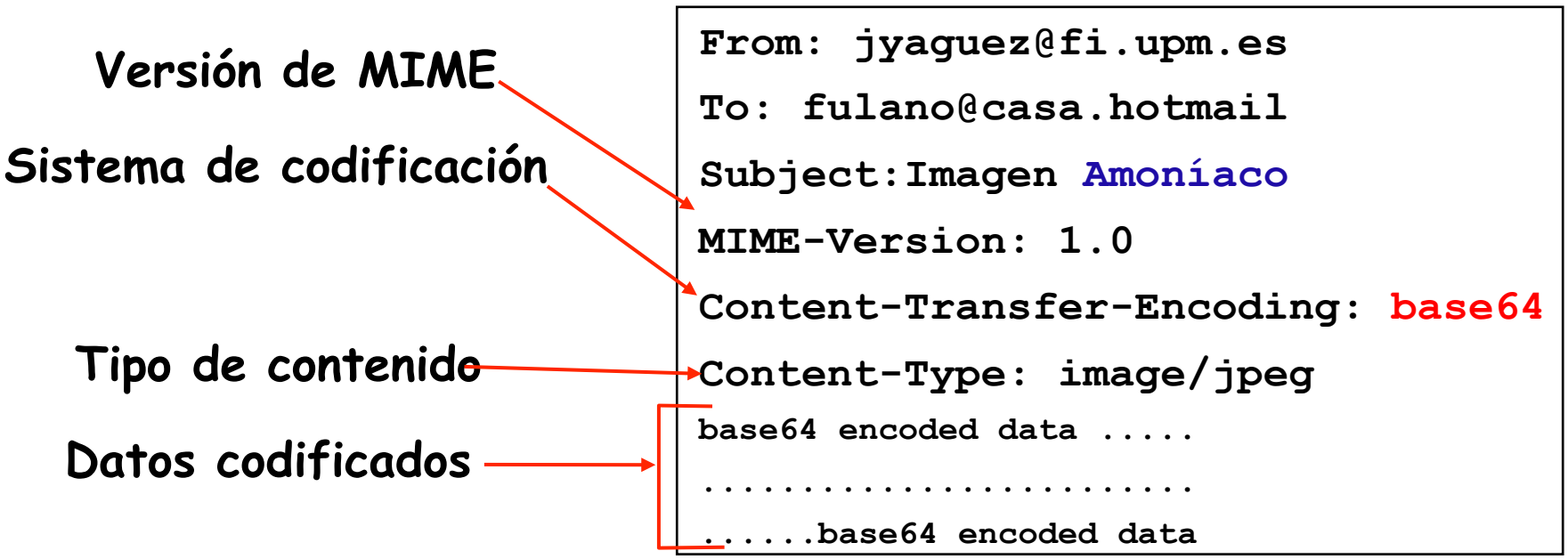
Nivel de aplicación
Servicio de correo electrónico

- **MIME – Encabezados de mensaje:**

Encabezado	Significado
MIME - Version	Identifica la versión MIME
Content - Description	Cadena de texto que describe el contenido
Content - Id	Identificador único
Content - Transfer - Encoding	Cómo se codifica el mensaje para su transmión
Content - Type	Naturaleza del mensaje

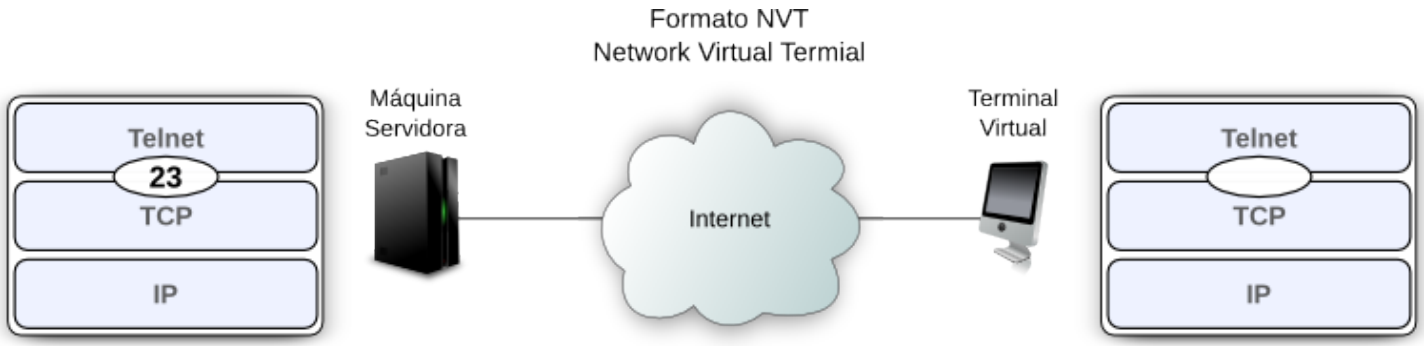
Nivel de aplicación
Servicio de correo electrónico

- **MIME:**



Nivel de aplicación
Protocolo Telnet (SSH)

- **Protocolo Telnet** RFC-854 STD-0008:
  - Uno de los primeros protocolos TCP/IP
  - Servicio de terminal remoto que permite que una máquina de usuario se convierta en un terminal de una máquina servidora en donde se esté ejecutando el proceso servidor Telnet y poder ejecutar comandos en dicha máquina servidora



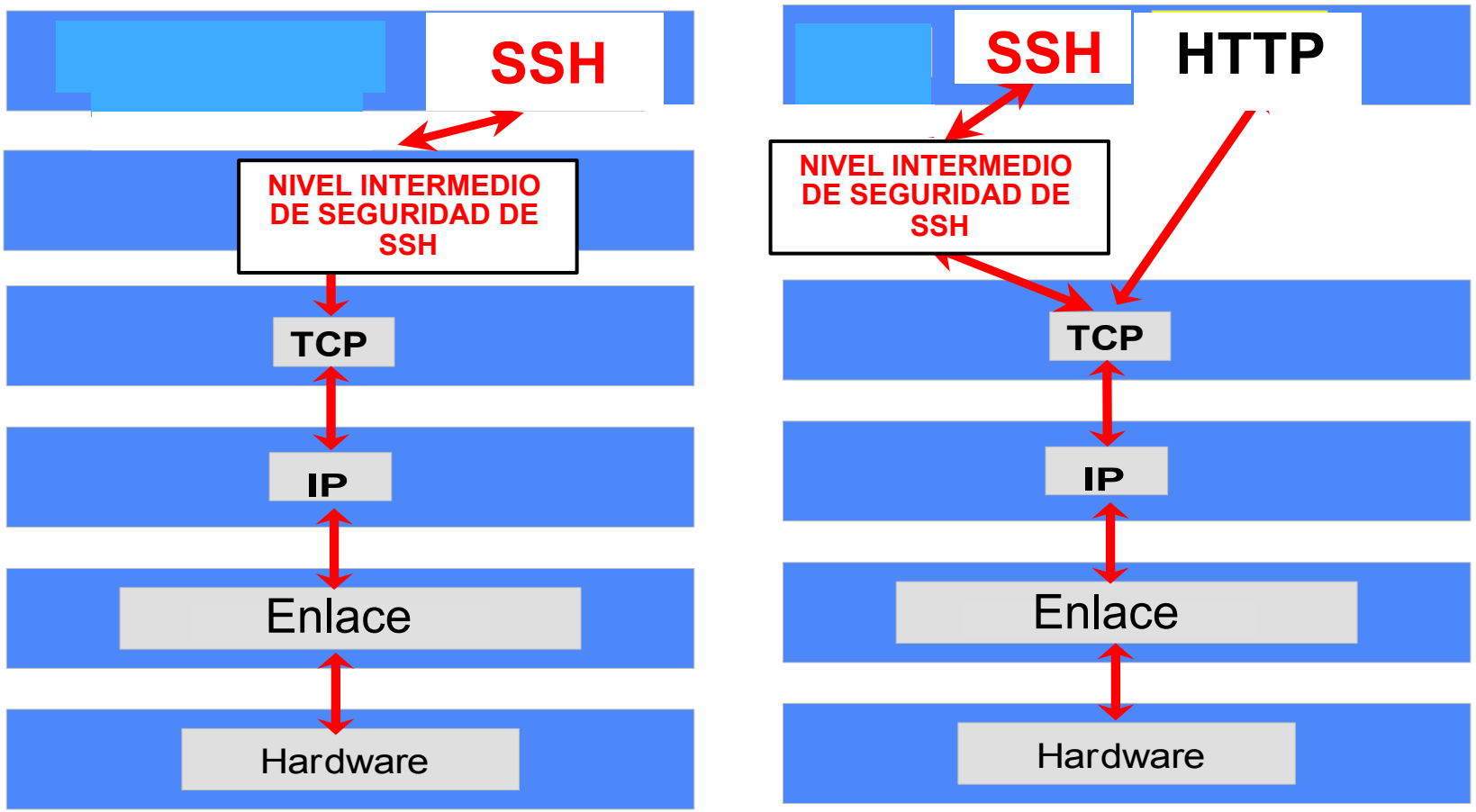
Nivel de aplicación
Protocolo Telnet (SSH)

- **Protocolo Telnet**:
  - **Formato NVT** (*Network Virtual Terminal*)
    - Caracteres de control ASCII pertenecientes a un Terminal Virtual de Red que permite una vez establecida la conexión TCP, negociar el modo de operación (opciones de comunicación) entre el proceso cliente y servidor de Telnet y poder teclear comandos en la máquina servidora
    - Esta negociación selecciona el tipo de terminal ANSI (Windows) o VT100 (Linux) para interpretar las secuencias de escape específicas, la definición del número de filas y columnas en la pantalla del cliente, el movimiento del cursor, tabulados, retornos de carro, etc.

Nivel de aplicación
Protocolo Telnet (SSH)

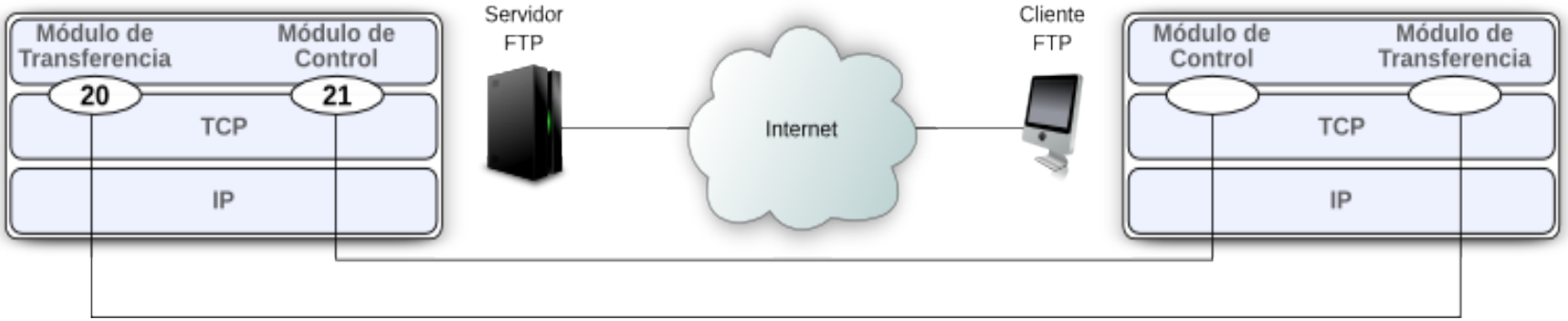
- **SSH** (*Secure Shell*): Intérprete seguro de comandos
  - SSH ha dejado obsoleto a Telnet ya que hace lo mismo que Telnet pero añadiendo más funcionalidad y de una forma segura
  - SSH es un protocolo de acceso remoto cliente/servidor TCP/IP para acceder y controlar por consola o shell de comandos y de forma segura una máquina remota y ejecutar comandos en dicha máquina
    - Copiar, mover y borrar ficheros entre dos máquinas
    - SSH server en el puerto 22 sobre TCP
    - Proporcionar servicios de seguridad equivalentes a SSL/TLS mediante autenticación y cifrado
    - OpenSSH (*Open Secure Shell*): Código abierto

SSH utiliza un Nivel Intermedio de Seguridad Propio equivalente a SSL

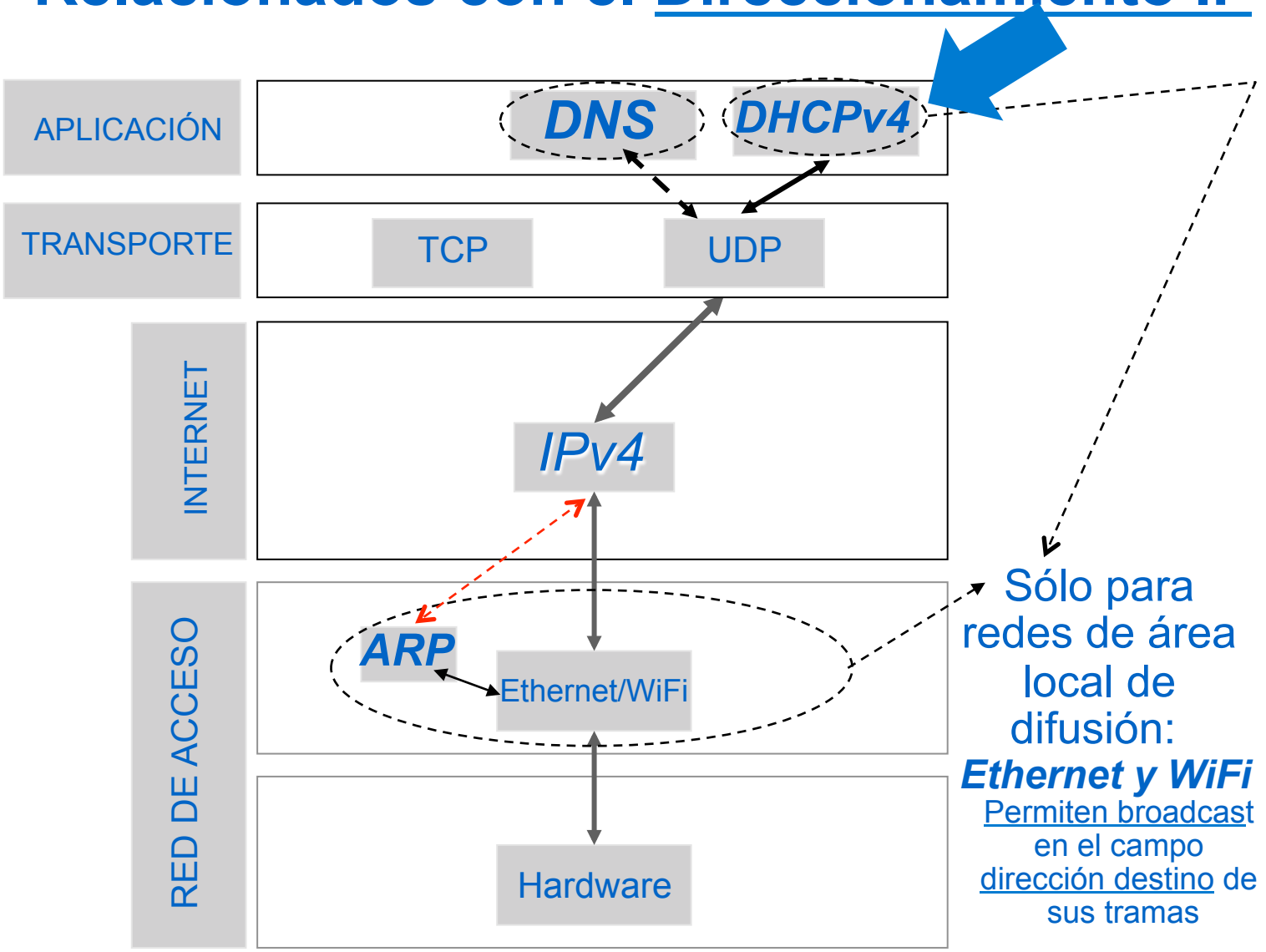


Nivel de aplicación
Protocolo FTP

- **Protocolo FTP** (*File Transfer Protocol*) RFC-959 STD-0009:
  - Uno de los primeros protocolos TCP/IP
  - HTTP lo ha dejado obsoleto. Actualmente, gran parte de los ficheros en Internet se descargan de Servidores HTTP
  - Utiliza dos módulos y dos números de puerto:
    - Puerto 21: establecer la conexión TCP entre dos entidades FTP
    - Puerto 20: transferir ficheros (*get/put*) entre dichas máquinas sin salir de la conexión previamente establecida



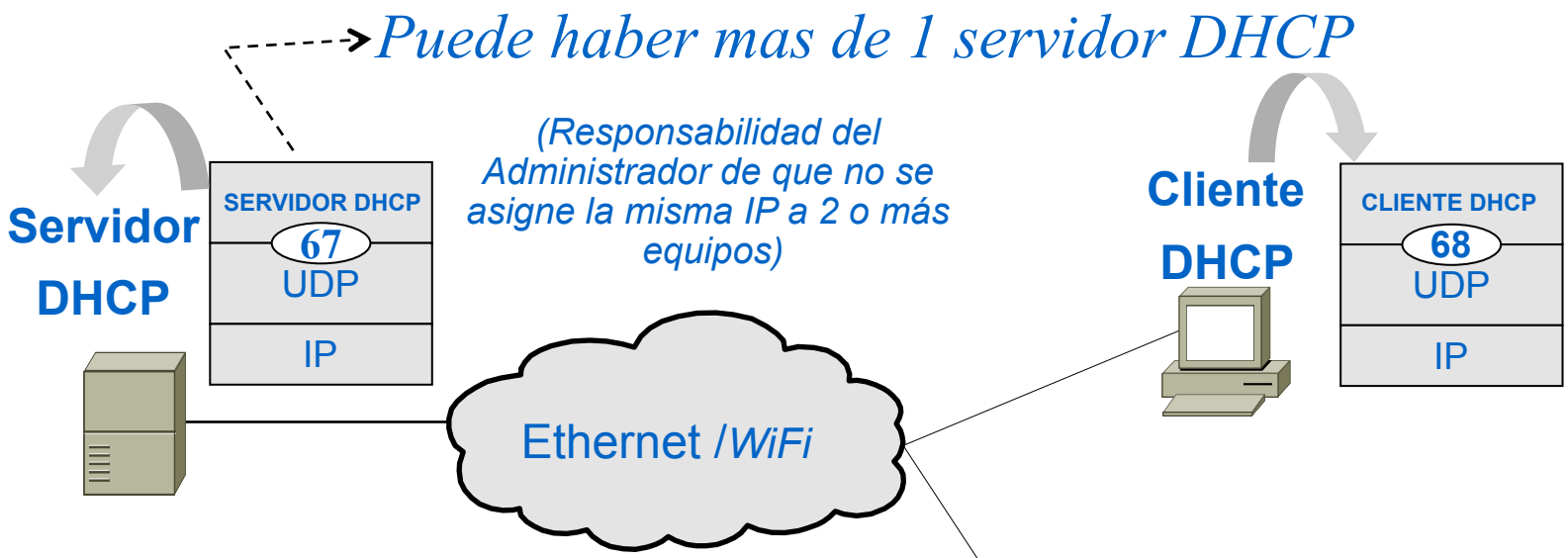
Protocolos y Niveles TCP/IP Relacionados con el Direccionamiento IP



DHCP (Dynamic Host Configuration Protocol)
RFC-2131

- ***PROTOCOLO DE CONFIGURACIÓN TCP/IP para las direcciones del nivel de enlace, o dirección MAC Ethernet/WiFi, de máquinas vecinas a un servidor DHCP en redes de difusión Ethernet/WiFi***
- ***Broadcast a nivel de paquete IP y trama***
- ***Protocolo del nivel de aplicación sobre UDP***
  - ***Configuración DINÁMICA Y ESTÁTICA de direcciones IP junto con el resto de información TCP/IP***
    - ***Asignación dinámica*: Modo habitual para *"n" máquinas (portátiles, tabletas, smartphones, etc.)* que no disponen de direcciones fijas ni del resto de información TCP/IP en la red de acceso**
    - ***Asignación estática*:**
      - ***Los equipos conectados de forma fija a la red Ethernet de la organización se configuran estáticamente***

Escenario Habitual del Protocolo DHCP
Configuración Dinámica TCP/IP



Protocolo dinámico de configuración TCP/IP:

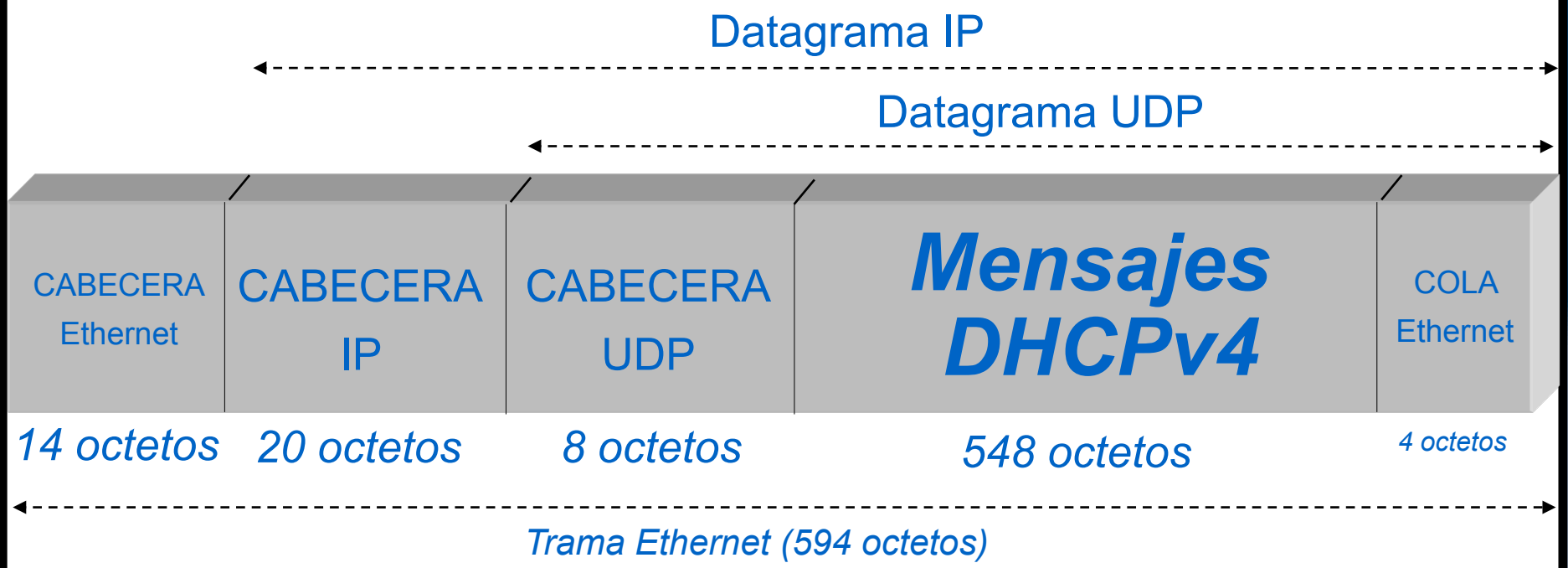
Asignación dinámica de direcciones IP temporales

(en función de las direcciones IP libres)

a las direcciones MAC solicitantes

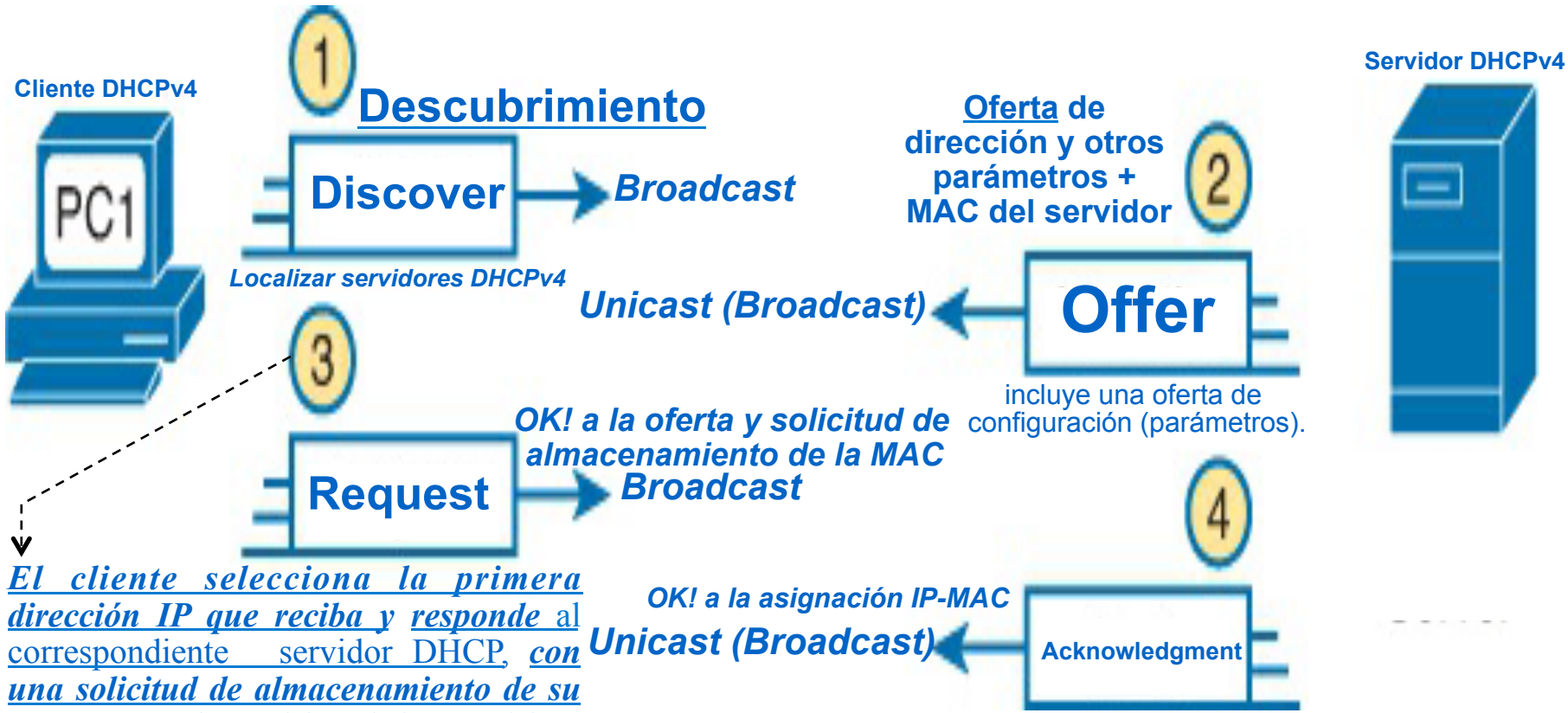
junto con la dirección IP de la red, dirección IP del siguiente router, máscaras asociadas, direcciones IP de los servidores DNS, etc.

Encapsulación de *Mensajes DHCPv4*



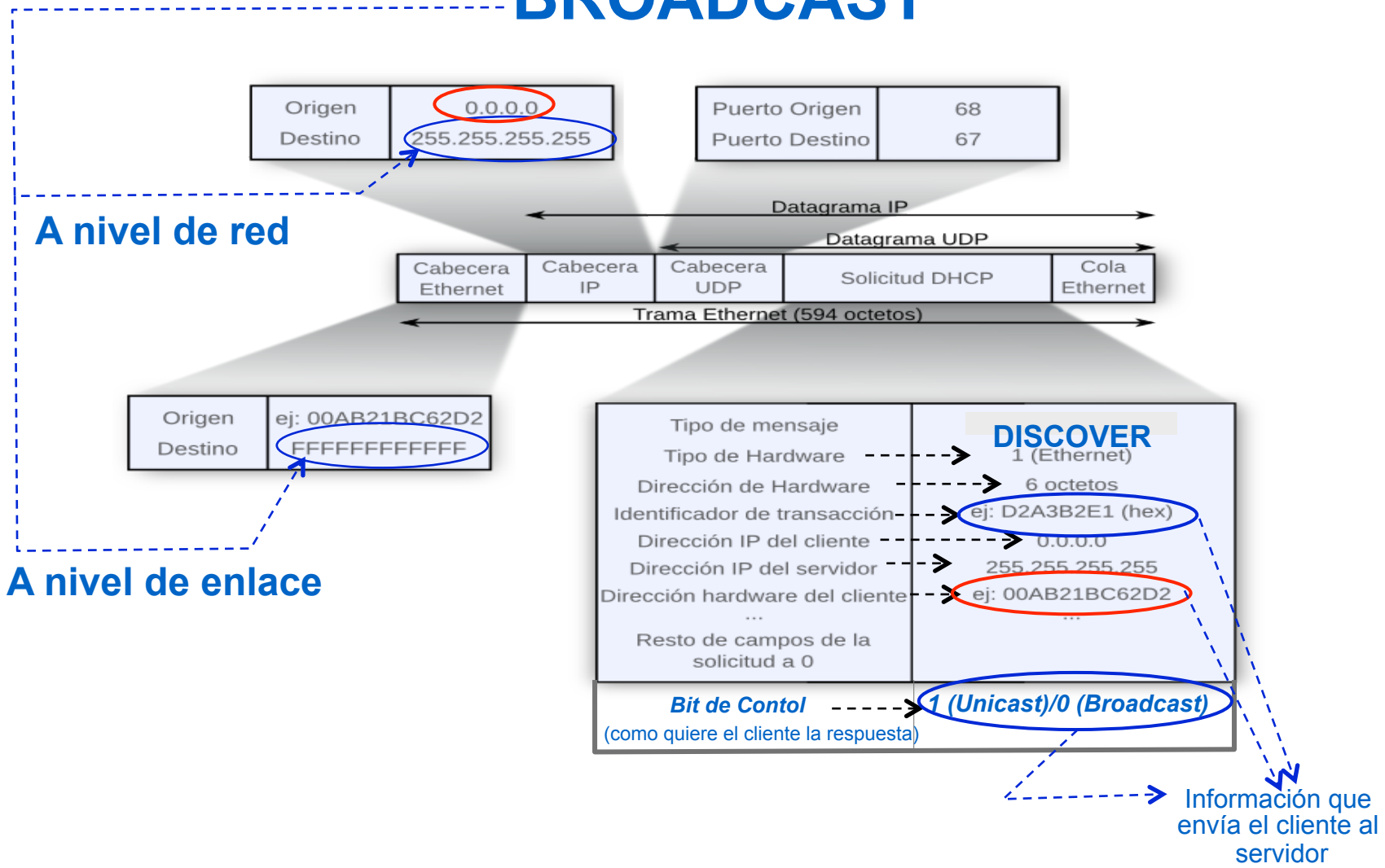
Mensajes Básicos DHCPv4 (Típica Sesión) entre Cliente y Servidor

Intercambio **DORA** en DHCPv4 (**D**iscover, **O**ffer, **R**equest y **A**cknowledgment)

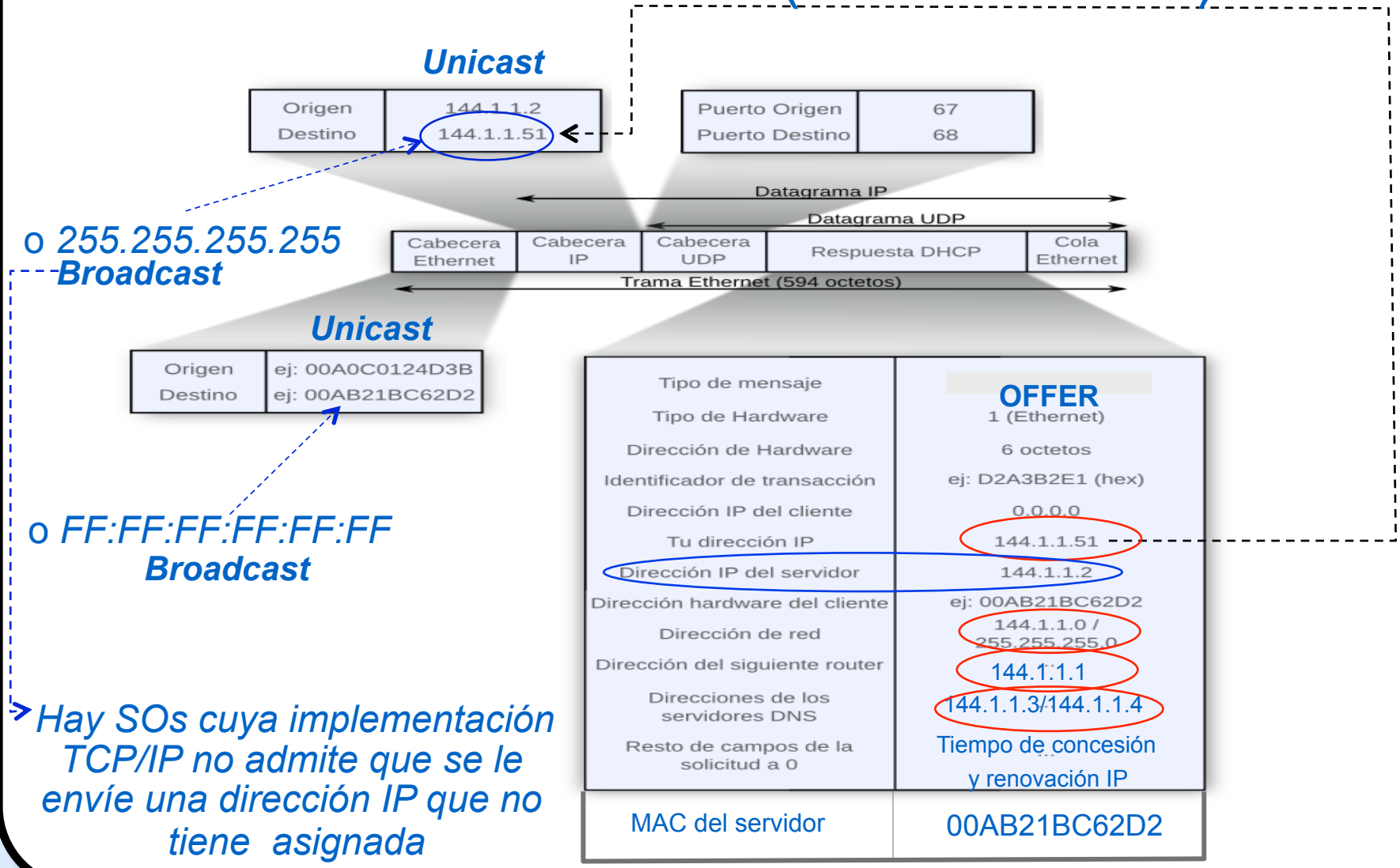


Ejemplo del Primer Mensaje Discover

BROADCAST

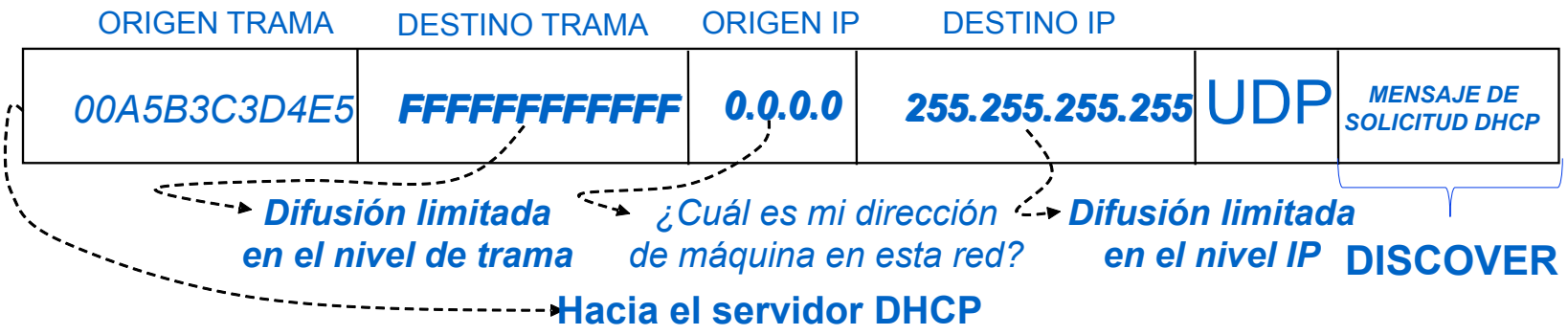


Ejemplo del Segundo Mensaje Offer UNICAST (BROADCAST)

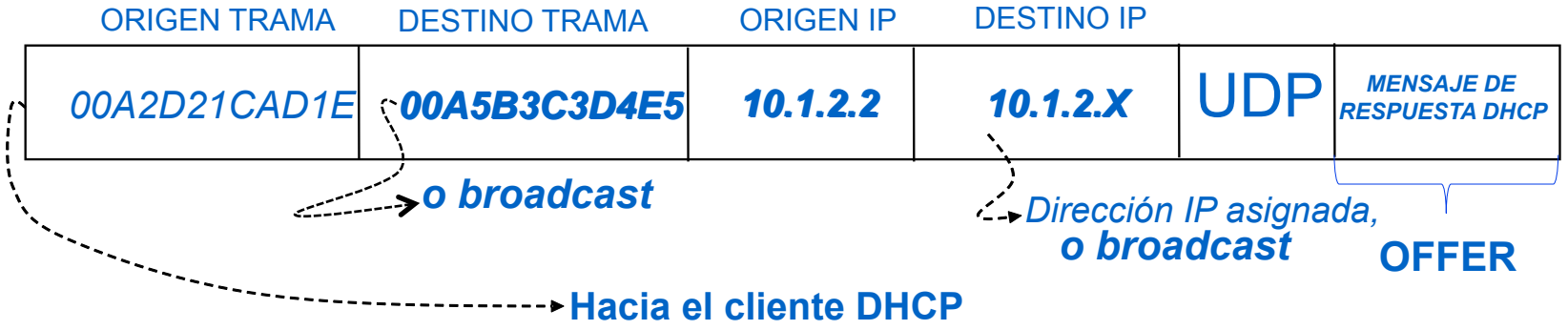


Resumen del Broadcast en el Nivel de Enlace y Red en Discover y Offer

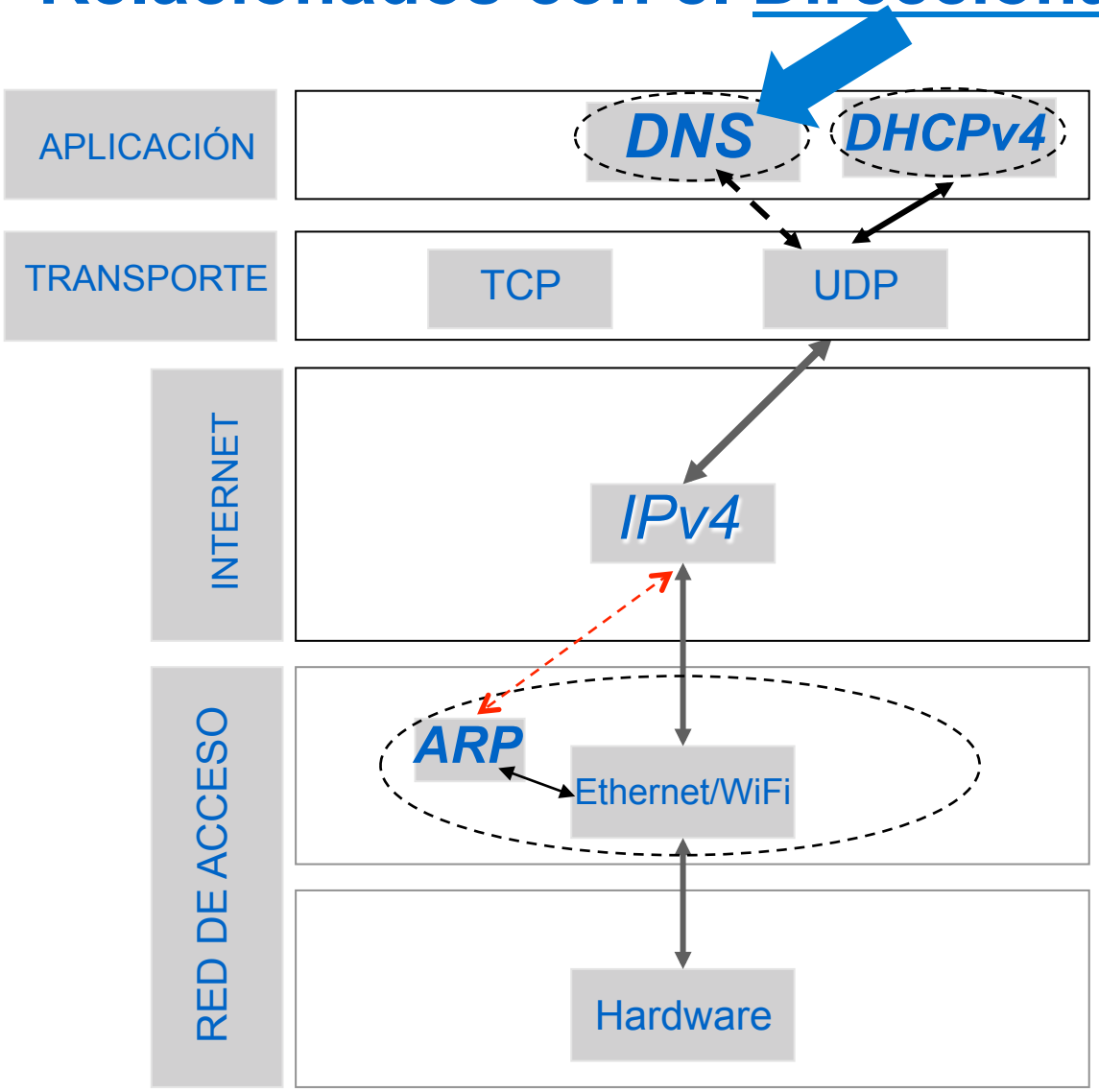
GENERADO POR EL CLIENTE DHCP



GENERADO POR EL SERVIDOR DHCP



Protocolos y Niveles TCP/IP Relacionados con el Direccionamiento IP



El Sistema DNS

RFC-1034 y RFC-1035

Sistema de Nombres de Dominio (DNS)

2 Componentes Principales

1. **BASE DE DATOS DNS**: *Una **BD Distribuida**, a través de los **servidores DNS locales de las diferentes organizaciones conectadas a Internet** y que mantienen registros locales con **las asociaciones locales entre los nombres simbólicos y las direcciones** numéricas qu conocen*

   - Ningún servidor DNS contiene la BD completa
   - Cada organización suele disponer de su propio servidor DNS
   - La BD se representa mediante una *ESTRUCTURA JERÁRQUICA, en forma de ÁRBOL, **DE NIVELES DE GESTIÓN DE BDs*** que permite garantizar la unicidad o singularidad de un NOMBRE DE DOMINIO
   - Cada BD DNS se gestiona a través de un servidor DNS
   - Para acceder a un servidor DNS, se necesita un cliente DNS y un protocolo DNS

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

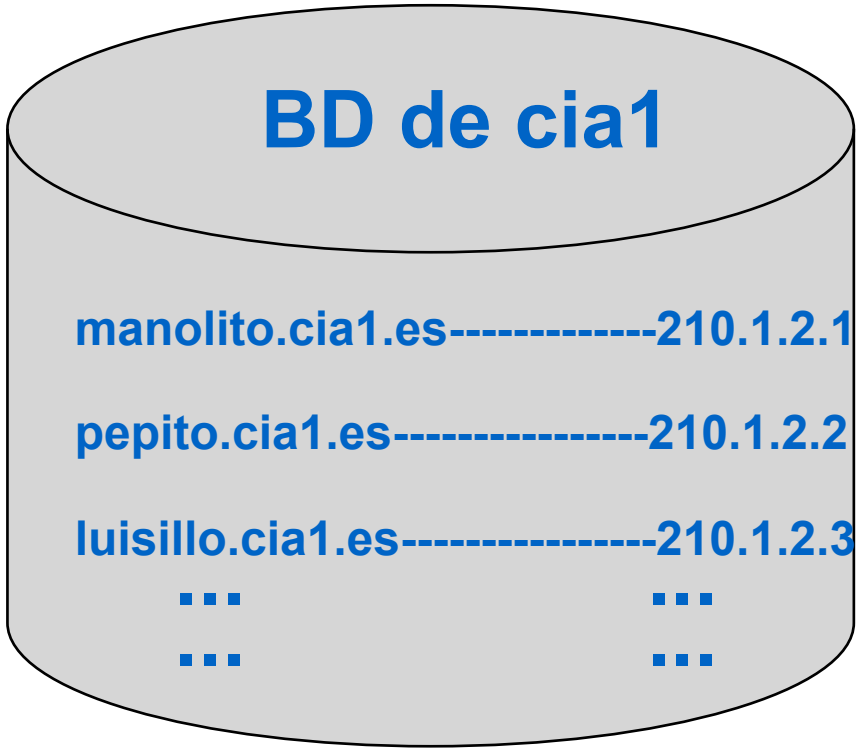
   - **Un cliente DNS comienza resolviendo un Nombre de Dominio**, interrogando a su servidor DNS
   - *Si un servidor DNS no tiene la RESOLUCIÓN SIMBÓLICA-NUMÉRICA solicitada, se convierte en cliente de otro servidor DNS* en la JERARQUÍA DNS establecida previamente en Internet

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

Servidor DNS de cia1

Ningún servidor DNS contiene la BD completa

Cada organización suele disponer de su propio servidor DNS que gestiona su BD DNS local

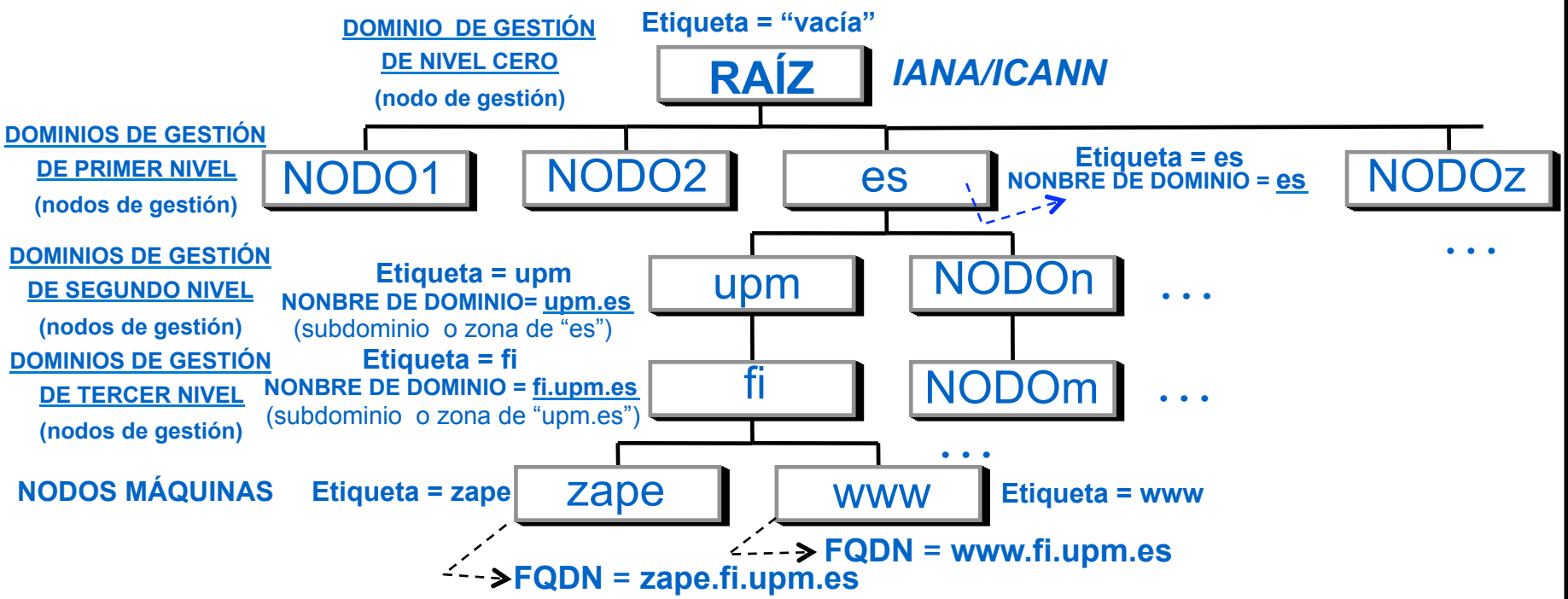


BD DNS de Internet = Internet: BD de **cia1** + BD de cia2+ …

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

ÁRBOL = Nodos de Gestión (organizaciones) + Nodos Máquinas

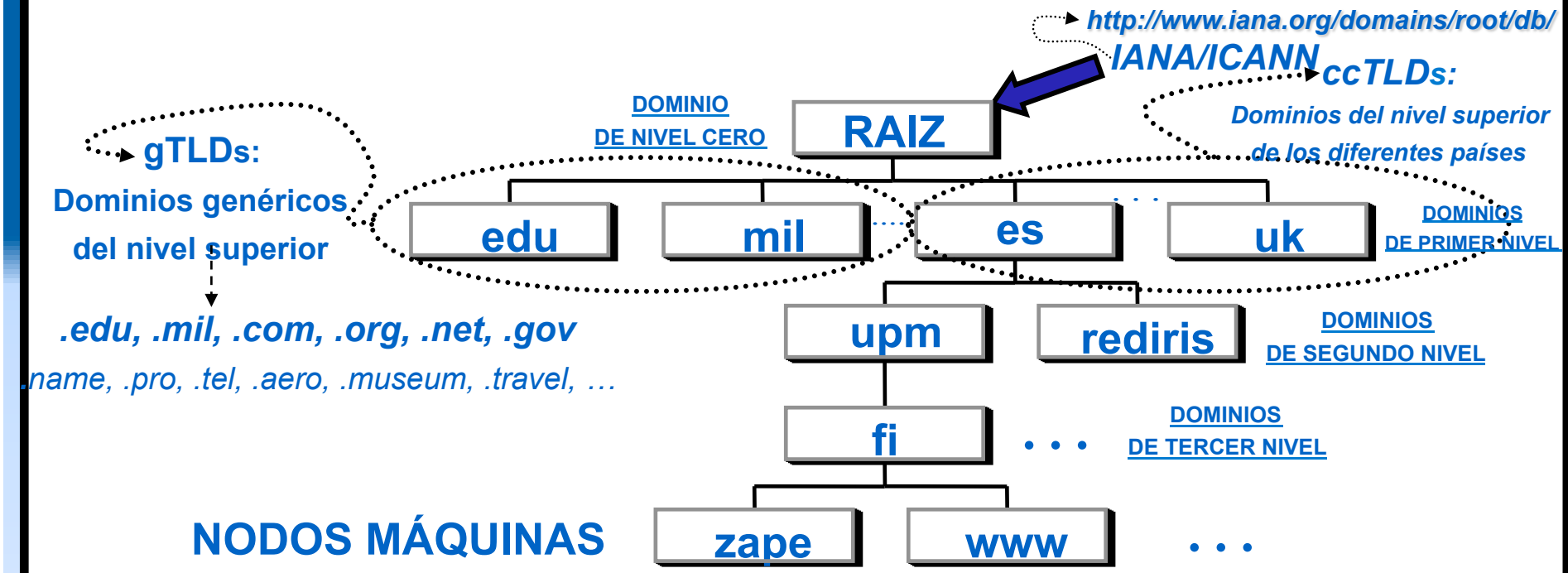


- Cada NODO DE GESTIÓN dispone de una **ETIQUETA** y un **NOMBRE DE DOMINIO**
- Cada NODO MÁQUINA disponde de una **ETIQUETA** y un **FQDN** (*Fully Qualified Domain Name*)
  - **FQDN = ETIQUETA (máquina) + NOMBRE DE DOMINIO (organización de la máquina)**

EJEMPLO DE DOMINIOS DE GESTIÓN DE PRIMER NIVEL o TOP LEVEL

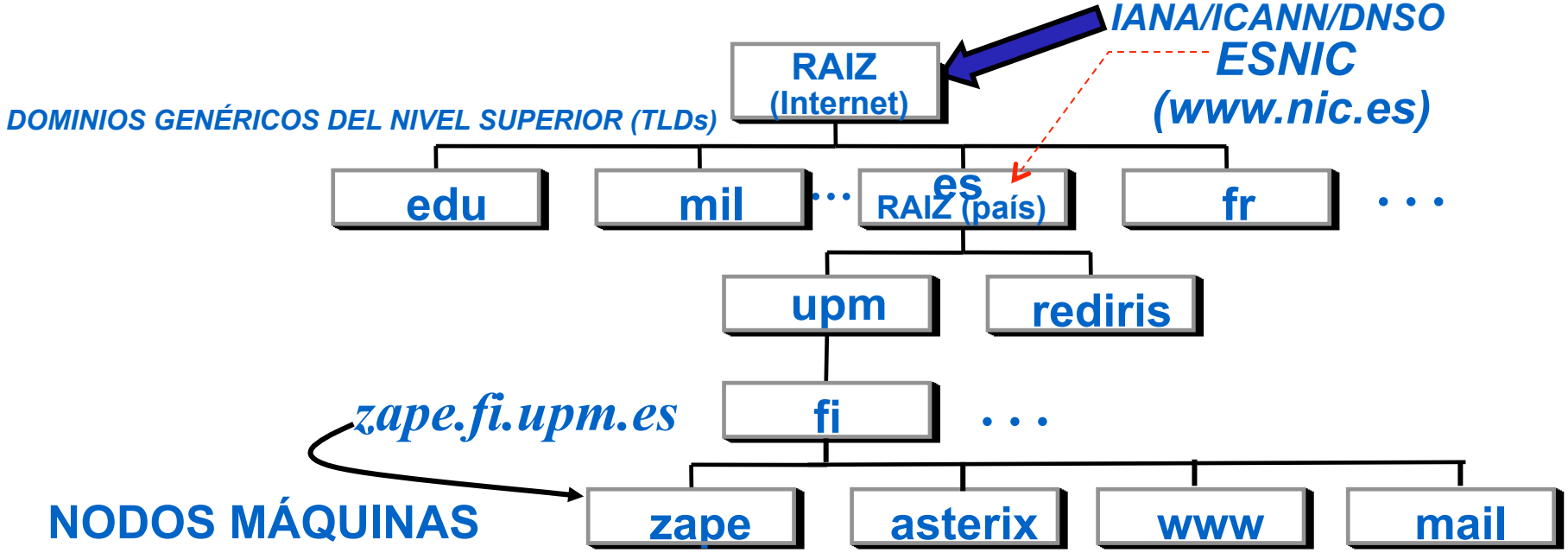
TLDs Genéricos + TLDs países

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



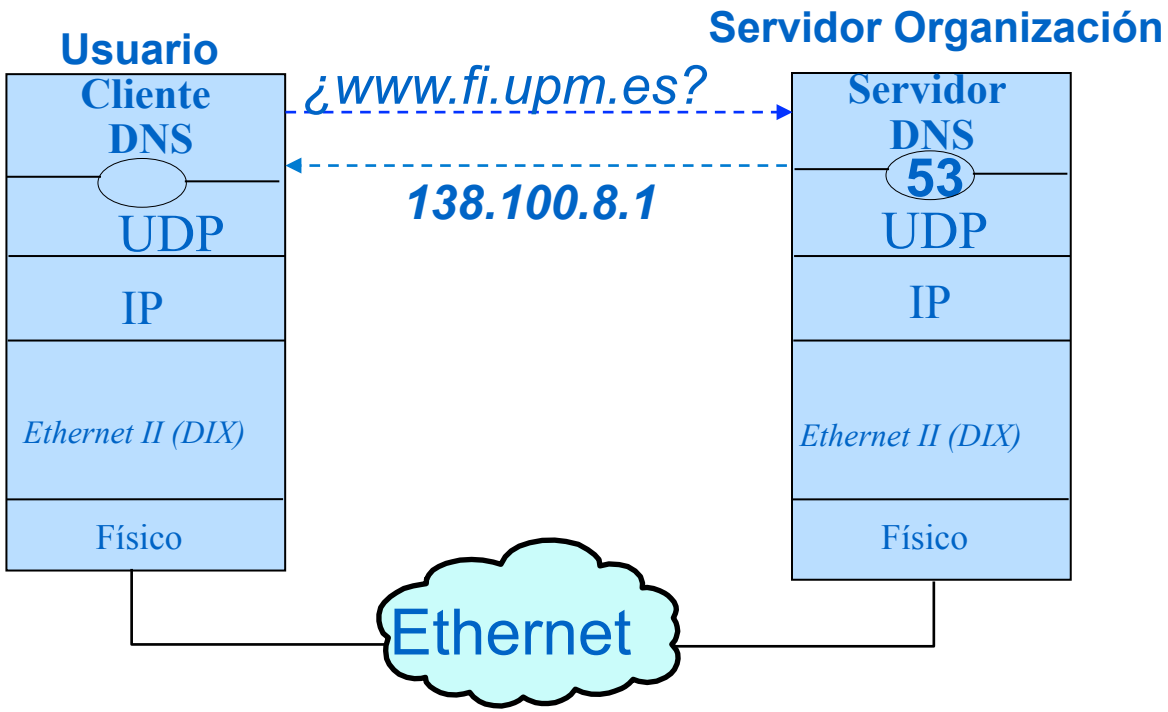
REGISTROS DE DIRECCIONES SIMBÓLICAS

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



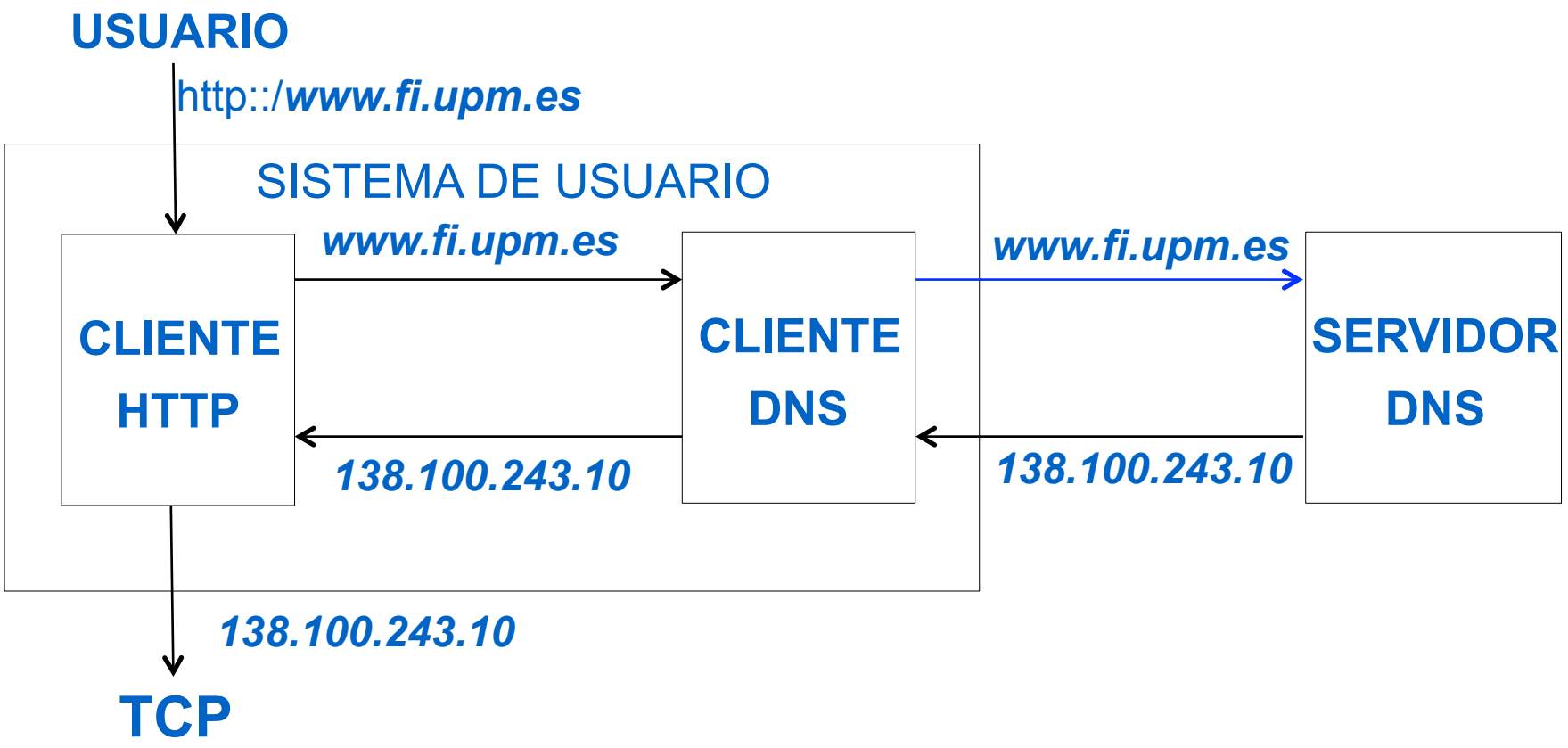
- **Agentes Registradores acreditados por ESNIC/IANA-ICANN:**
  - *Registros de nombres simbólicos de primer nivel (x.es, x.com, x.org,…) y segundo nivel (x.com.es, x.org.es, …)*
- Asimismo, existen dominios particulares TLD (aprobados por el IANA) con un coste superior:
  - *p. ej.: .madrid, .empresa,...,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

El Protocolo DNS (DOMAIN NAME SYSTEM)

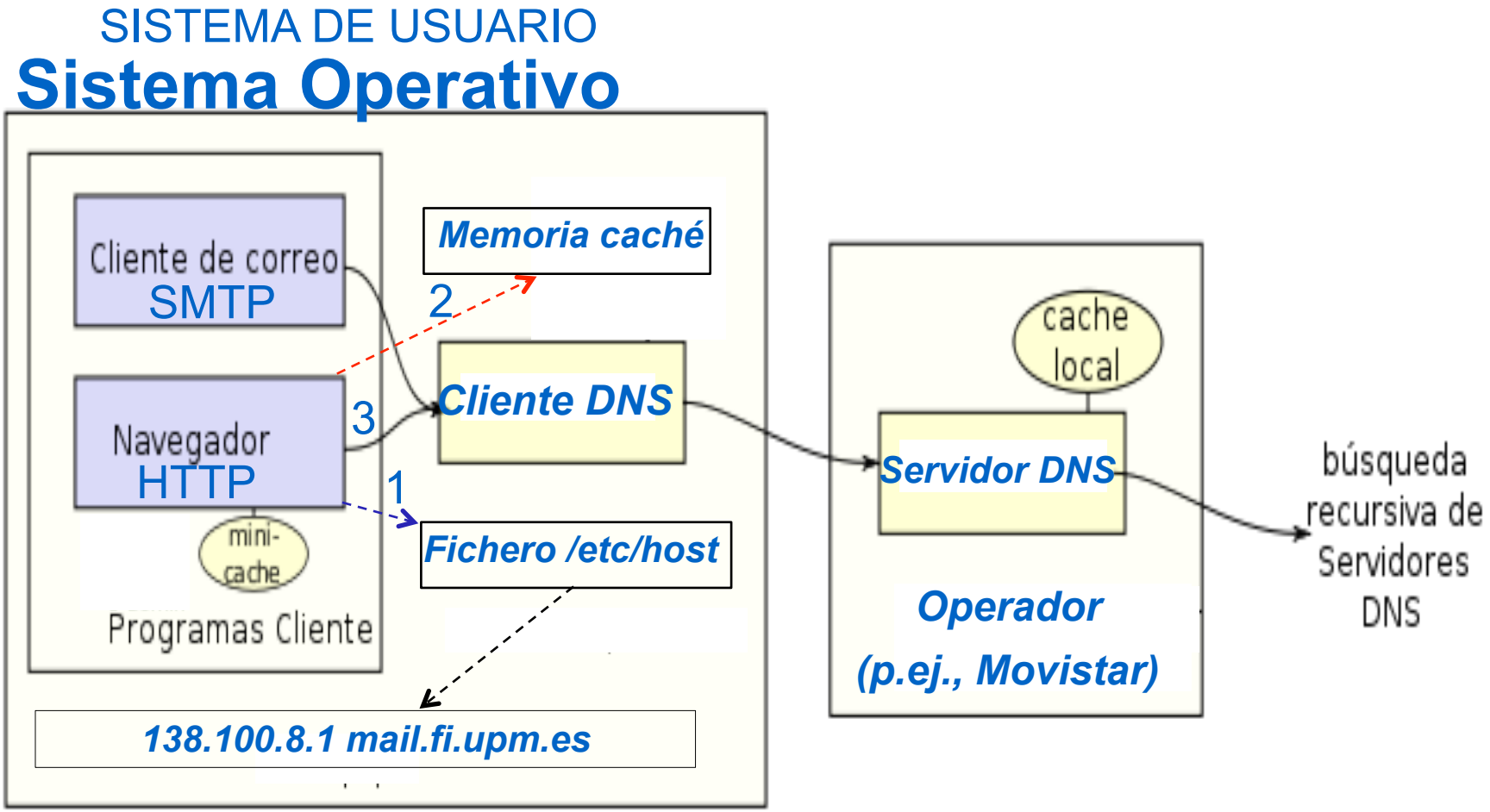


- ***Protocolo del nivel de aplicación que sigue el modelo cliente y servidor para la RESOLUCIÓN DE NOMBRES DE DOMINIO en direcciones numéricas***
  - ✓ Los **programas cliente y servidor** se pueden dividir en **dos categorías**
    - Protocolos que pueden ser usados directamente por el usuario: *HTTP, SMTP, etc.*
    - **Protocolos como DNS que, a su vez,** *da soporte, a otros protocolos* **o aplicaciones como *HTTP y SMTP* entre otros**

Ejemplo de Uso del
Servicio DNS Solicitado Previamente por HTTP

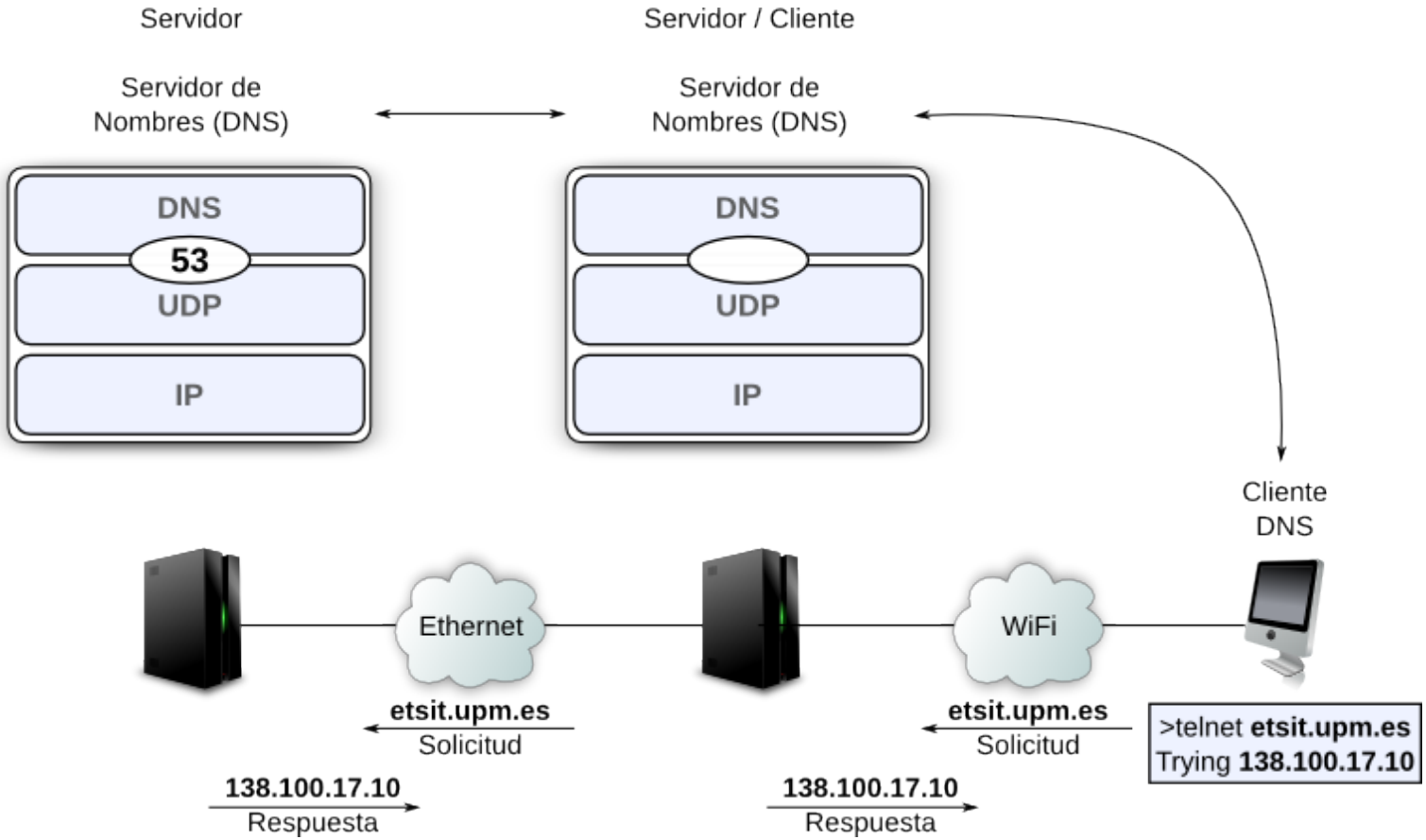


Interacciones Previas al Servicio DNS



Nivel de aplicación
Protocolo DNS

- **Protocolo DNS** (*Domain Name System*) RFC-1035 STD-00013:
  - Servicio de direcciones simbólicas en direcciones IP numéricas



El Sistema DNS

Tipos de Traducciones

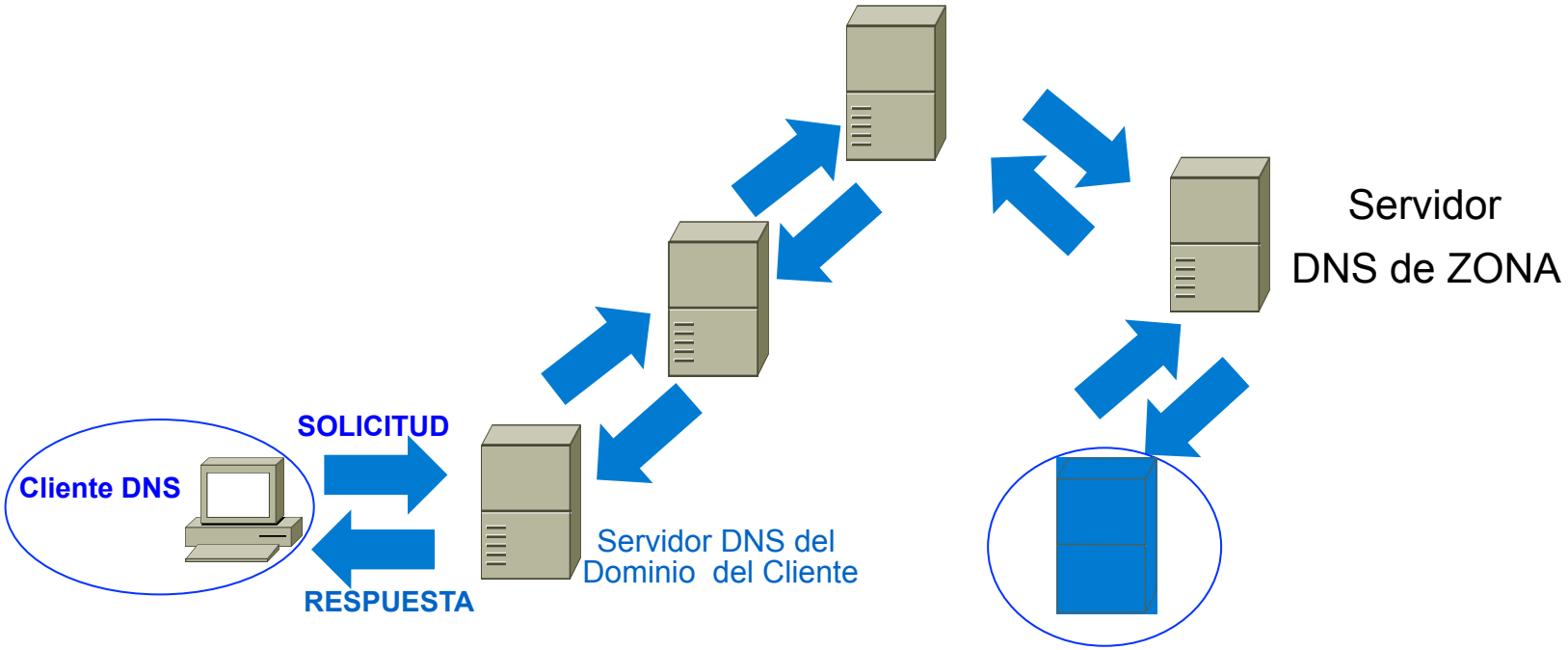
- Resoluciones

Dos Tipos de Traducciones o Resoluciones DNS

1. Recursiva
2. Iterativa

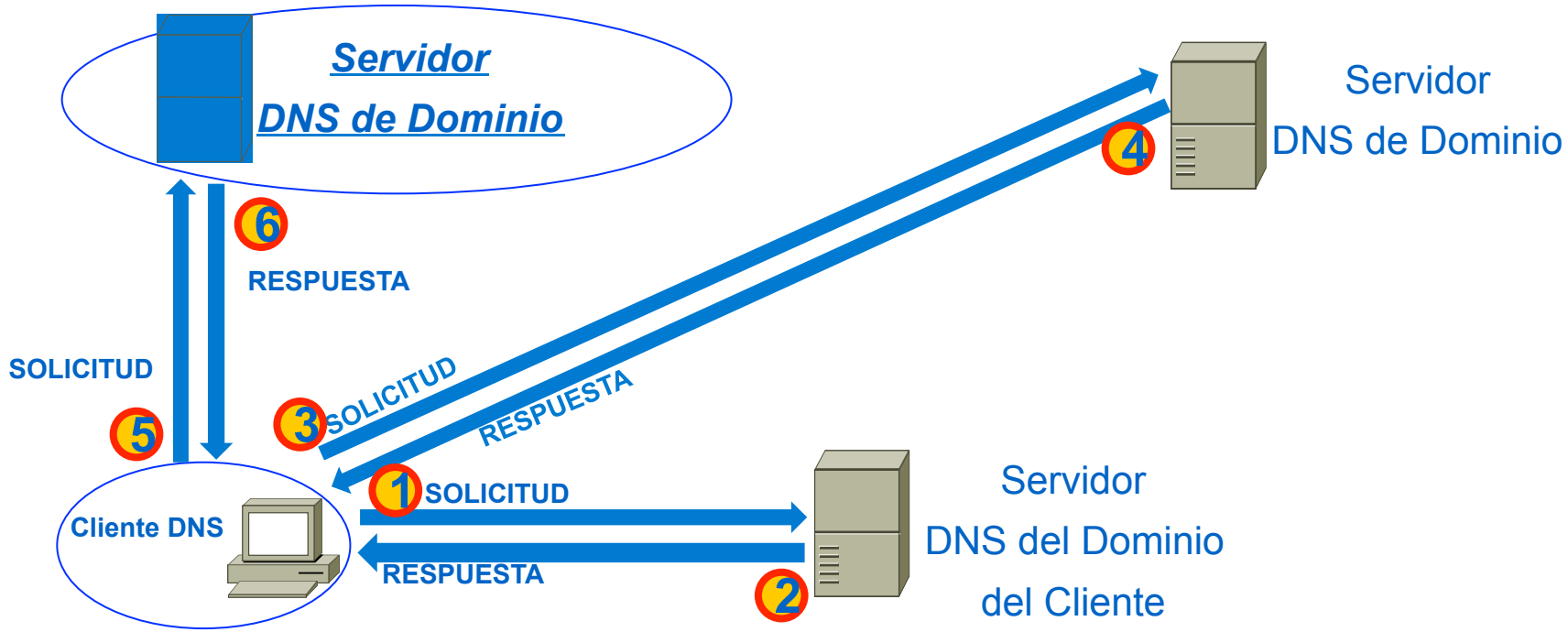
Nivel de aplicación

Traducciones DNS Recursivas



Nivel de aplicación

Traducciones DNS Iterativas



El Sistema DNS

Registros DNS

Contenido de un Registro DNS: 5 Tuplas en ASCII

NOMBRE DE DOMINIO (FQDN)	TTL	CLASE	TIPO	DATOS
mail.fi.upm.es	3600	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	IN	A	138.100.8.1
mail.fi.upm.es	3600	IN	A	138.100.243.11
www.fi.upm.es	86400	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 (NSLOOKUP), es decir, **obtener un Nombre de Dominio a partir de una dirección IP**
    - *P.ej., nslookup 138.100.8.1*

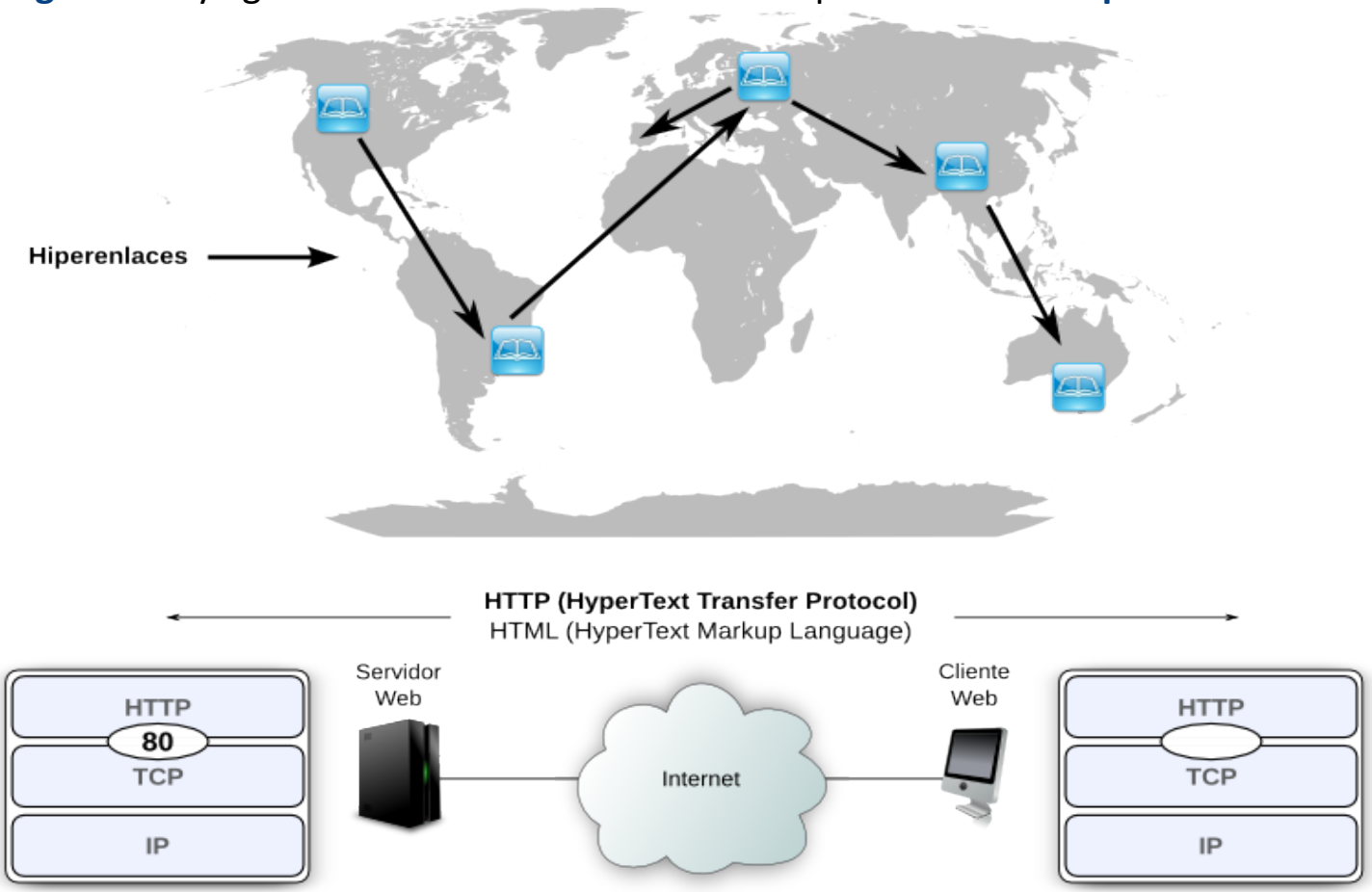
Contenido del Campo Tipo: MX

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*

Nivel de aplicación
Protocolo HTTP

- **Servicio Web** (WWW): Colección mundial de ficheros de texto y multimedia en **código HTML** y ligados a través de un sistema de hiperenlaces vía el **protocolo HTTP**



Nivel de aplicación
Protocolo HTTP

- **Terminología Web**:
  - Cada **sitio Web** almacena uno o más documentos, denominados páginas Web
  - Una **página Web** consta de un **fichero HTML** (*Hypertext Markup Language*) base que contiene el código HTML de la página Web inicial y que incluye **hiperenlaces** o referencias a otros objetos en la misma máquina o en diferentes máquinas
  - Los **objetos** pueden ser ficheros HTML (conteniendo, a su vez, sus propios objetos), ficheros pdf, ficheros doc, imágenes JPEG, ficheros de audio, ficheros de vídeo, ...
  - Cada objeto es direccionable por su **URL**
  - Las páginas se pueden recuperar y visualizar utilizando **navegadores Web**:
    - Cliente HTTP
    - Intérprete HTML, Java o JavaScript, dependiendo del tipo de documento Web que se desea visualizar

Nivel de aplicación
Protocolo HTTP

- **Localizador URL** (*Uniform Resource Locator*):

  Formato utilizado por el protocolo HTTP para localizar un objeto (HTML, texto, audio, vídeo, ...) por Internet, mantenido por un servidor Web en un sitio Web

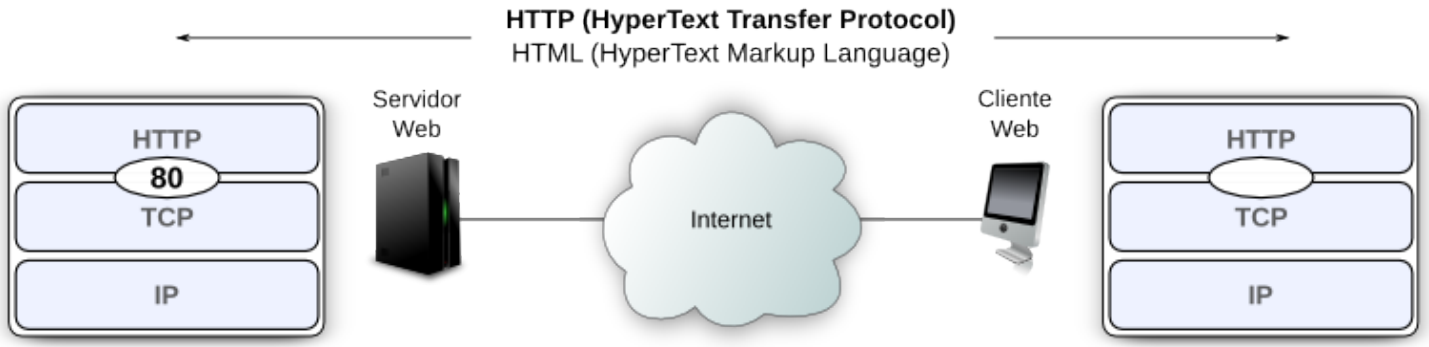
  protocolo://máquina:puerto/ruta

  - **protocolo**: http (también telnet, ftp, ...)
  - **máquina**: Alias del sitio Web (suele comenzar por www) o dirección IP
  - **puerto**: Número entero que identifica al proceso servidor (por defecto 80)
  - **ruta**: Nombre del fichero o camino (*path*) de directorios (separados por "/") para acceder al fichero

http	://	pegaso.ls.fi.upm.es	/	redes_computadores	/	normas2012-13.pdf
protocolo		máquina		ruta		objeto (fichero pdf)

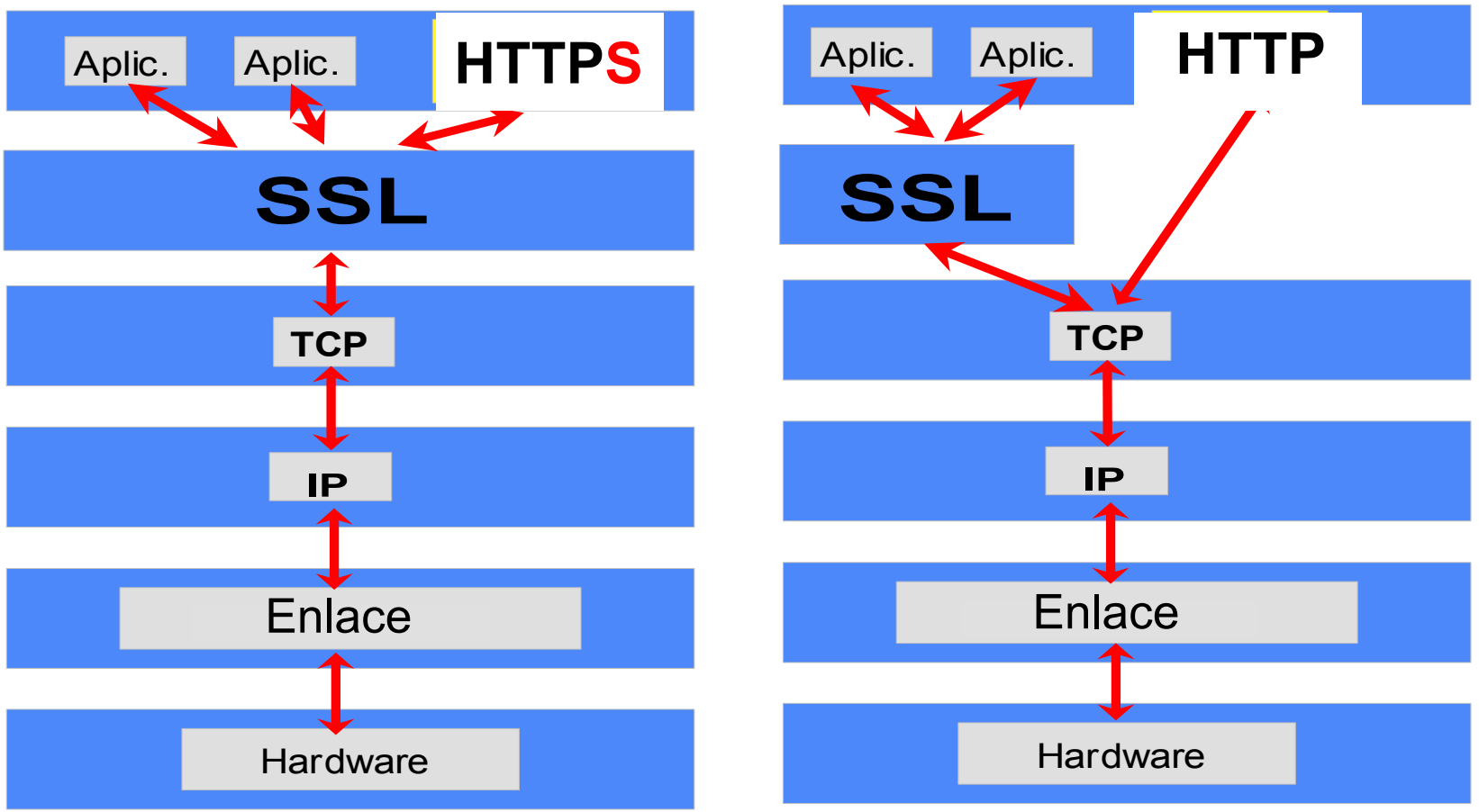
Nivel de aplicación
Protocolo HTTP

- **Protocolo HTTP** (*Hypertext Transfer Protocol*) RFC-2616:
  - Junto con SMTP, es uno de los protocolos TCP/IP del nivel de aplicación más utilizados por usuarios finales en Internet
  - Protocolo de presentación y distribución de información en Internet
  - Se emplea para descargar desde un cliente HTTP (navegador) cualquier tipo de fichero mantenido por un servidor HTTP
    - Fichero HTML de una página Web
    - Ficheros de texto (pdf, txt, doc, …), audio (AAC, mp3, WAV…), vídeo (MPEG-4, H. 264, …), ...



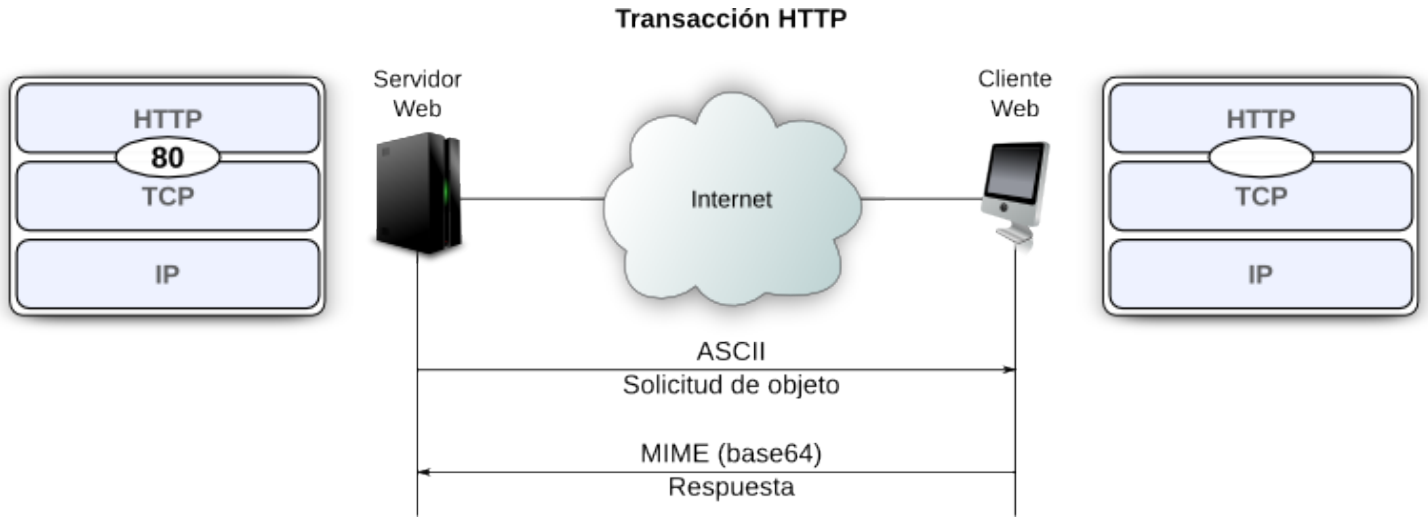
PRESENTACIÓN y DISTRIBUCIÓN DE LA INFORMACIÓN SEGURA y NO SEGURA

Convivencia HTTP Seguro y No Seguro



Nivel de aplicación
Protocolo HTTP

- **Protocolo HTTP**:
  - HTTP 1.1 es un protocolo cliente/servidor orientado a transacciones
  - Para proporcionar fiabilidad, HTTP hace uso de TCP

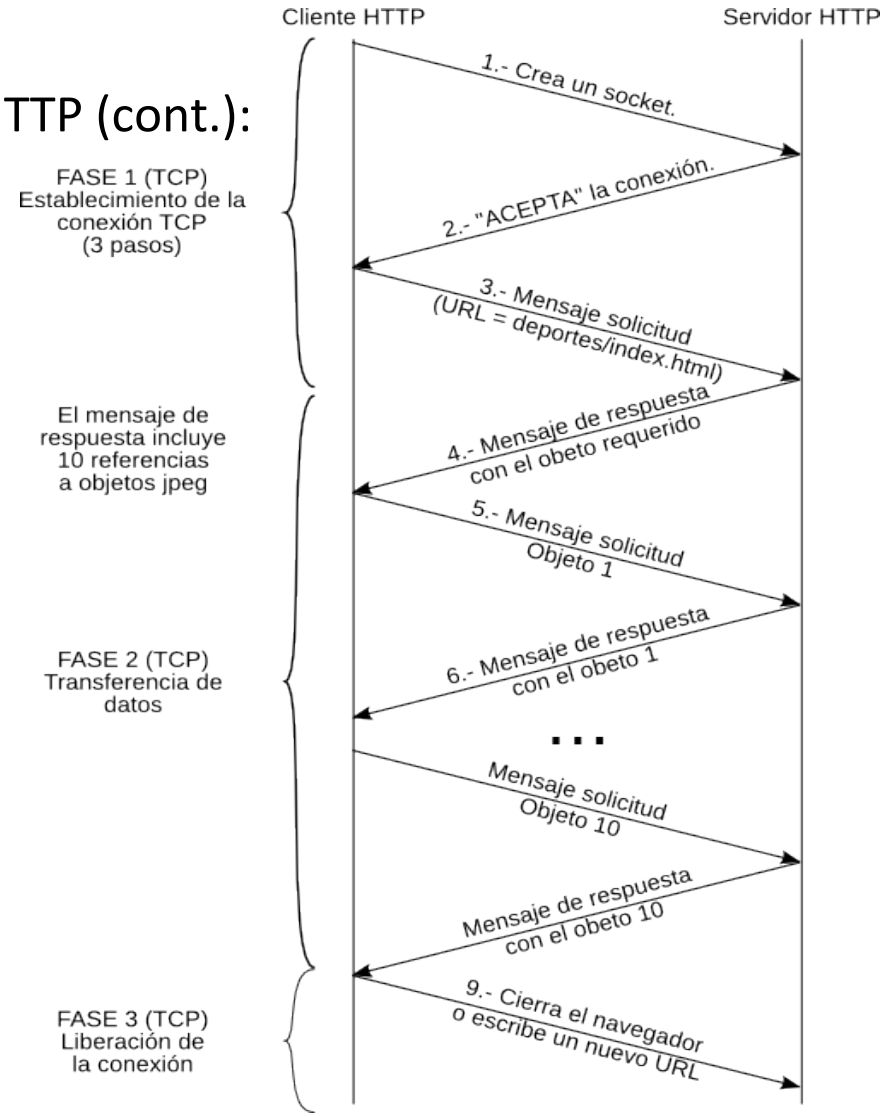


Nivel de aplicación
Protocolo HTTP

- Características del protocolo HTTP:
  - **Persistente**: Soporta conexiones persistentes entre el cliente y el servidor. Se pueden enviar múltiples ficheros por una única conexión TCP
    - Sin *pipelining*:
      - El cliente envía una nueva solicitud sólo cuando ha recibido el objeto de la anterior solicitud
      - Un RTT por cada objeto referenciado
    - Con *pipelining*:
      - El cliente puede enviar más de una solicitud tan pronto encuentre más de un fichero referenciado en el código HTML
      - Un solo RTT para todas las referencias

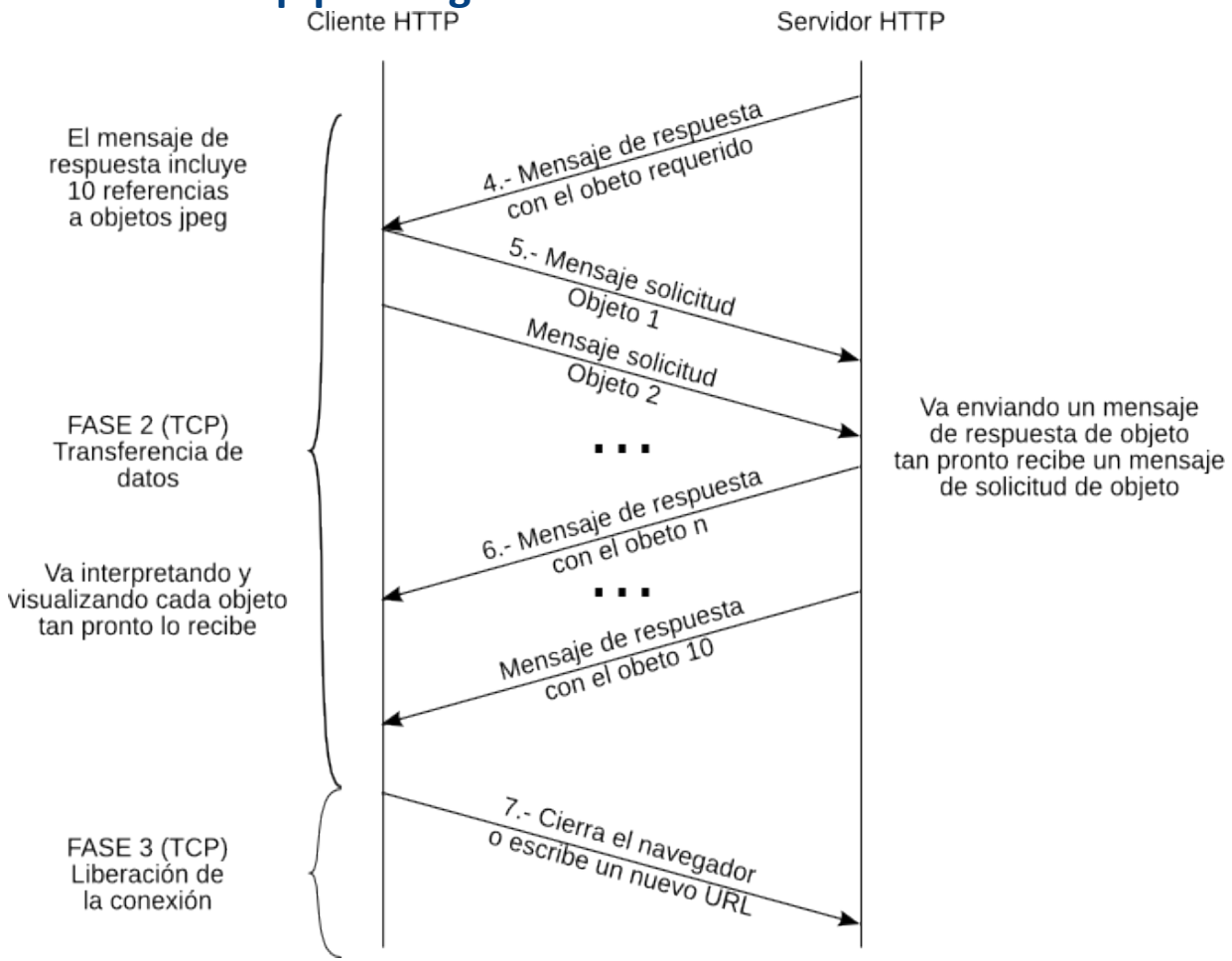
Nivel de aplicación
Protocolo HTTP

- Características del protocolo HTTP (cont.):
  - **Persistente sin pipelining**:



Nivel de aplicación
Protocolo HTTP

- Características del protocolo HTTP (cont.):
  - **Persistente con pipelining**:



Nivel de aplicación
Protocolo HTTP

- Características del protocolo HTTP (cont.):
  - **Sin estado**: El servidor HTTP no mantiene el estado sobre las solicitudes o acciones (historial) de los clientes HTTP al cerrarse la conexión TCP. Para mantener el estado, el programador de la aplicación Web tendrá que gestionar el estado por encima de HTTP:
    - Reescritura de los URL
    - Campos ocultos en la página HTML
    - *Cookies*: Ficheros de texto que contienen las acciones del usuario para cada servidor Web visitado y que se almacenan en el disco duro del cliente, a través de su navegador, a petición del servidor Web. Esta información puede ser recuperada por el servidor en posteriores visitas para:
      - Control de usuarios: Cuando un usuario introduce su usuario y contraseña, se almacena una *cookie* para que no tenga que introducirlos por cada página del servidor
      - Seguimientos de usuarios: Estadísticas de usos, aficiones, cestas de compras, ...
      - Personalización del sitio Web en función de las preferencias del usuario: Particularizar el aspecto en cuanto a presentación y funcionalidad
      - …

Nivel de aplicación
Protocolo HTTP

- **Modelos de seguridad en Internet:**
  - Seguridad en el nivel de transporte (Nivel Intermedio SSL/TLS)
    - Un nivel adicional de seguridad intermedio sobre TCP y transparente a los niveles superiores
    - Se implementa extremo a extremo ofreciendo un nivel de transporte seguro común, estándar y homogéneo para todas las aplicaciones montadas sobre TCP
      - Autenticación de un servidor
      - Autenticación de un cliente
      - Conexión cifrada segura cliente/servidor (confidencialidad)

Formato de los Mensajes HTTP de Solicitud y Respuesta

Mensaje de Solicitud

Línea de Solicitud
Información del navegador y SO

Cabeceras
(presente sólo en algunos mensajes)
- *Contenido del objeto enviado al servidor*
- *Longitud del objeto enviado*

Una línea en blanco

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

Mensaje de Respuesta

Línea de Estado

Cabeceras
- *Contenido del objeto solicitado*
- *Longitud del objeto solicitado*
- *etc.*

Una línea en blanco

Cuerpo
(objeto solicitado)

Líneas de Solicitud y Estado y Formato de la Cabecera

LÍNEA DE SOLICITUD (Mensaje de solicitud)

Espacio Espacio

Tipo de Solicitud | URL | Versión de HTTP

GET, POST, PUT, …

…

HTTP/1.1

User-Agent: nombre-cliente (nombre del navegador, la versión, el SO, y el idioma)

LÍNEA DE ESTADO (Mensaje de Respuesta)

Espacio Espacio

Versión de HTTP | Código de Estado | Frase de Estado

HTTP/1.1

200 OK, 400 SOLICITUD INCORRECTA, …

Código de 3 dígitos: Rango 100 (informativo), rango 200 (solicitud con éxito), rango 300 (redirección a otro URL), rango 400 (error al cliente) y rango 500 (error en el sitio servidor)

FORMATO DE LAS CABECERAS (Mensajes de Solicitud y Respuesta)

PARES NOMBRE:VALOR

Nombre de la Cabecera | : | Valor de la Cabecera

Espacio

RESPUESTA HTTP
Formato MIME
(base 64)

Date: Thu, 06 Aug 2009 12:00:15 UT
Server: Apache/2.0.3
Content-Length: 6821
Content-Type: text/html
…

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

  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

- 5xx Error del servidor

  500 Error interno

  501 No implementado

  503 Servicio no disponible

  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/1.1 ----------> **LÍNEA DE SOLICITUD**

Host: ***www.miapache.com***
User-Agent: ***Mozilla 5.0, Windows NT 6.0, Firefox 42.0***
Accept: ***text/html, application/xml***
Accept Language: ***en-US, en***
Accept Encoding: ***gzip***
...

----------> **CABECERAS**
PARES NOMBRE:VALOR

2. LA RESPUESTA DEL SERVIDOR

HTTP/1.1 200 OK ----------> **LÍNEA DE ESTADO**

Server: ***Apache/2.0.3***
Date: ***Fri, 31 Oct 2015 23:59:59 GMT***
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 (Data)
(index.html)

Nivel de aplicación
Protocolo HTTP

- **Tipo de solicitud**: en función de los métodos HTTP utilizados
  - **GET**: Método HTTP para solicitar un objeto, a veces, con parámetros que se encapsulan en el localizador URL e indican lo que busca el usuario
    - La mayoría de las solicitudes HTTP son mediante GET
    - Los parámetros pasados se codifican y son visibles en el URL
    - Pares de valores: nombre-valor
  - **POST**: Método HTTP para solicitar un objeto siempre con parámetros o datos del usuario que no son visibles en el localizador URL
    - Típica solicitud HTTP cuando se pulsa el botón de un formulario
    - Los parámetros se introducen en el cuerpo del mensaje y no son visibles en el localizador URL
    - Pares de valores: nombre-valor
    - El tamaño de los datos no está limitado como en GET
  - **PUT**: Método HTTP para enviar un objeto al servidor