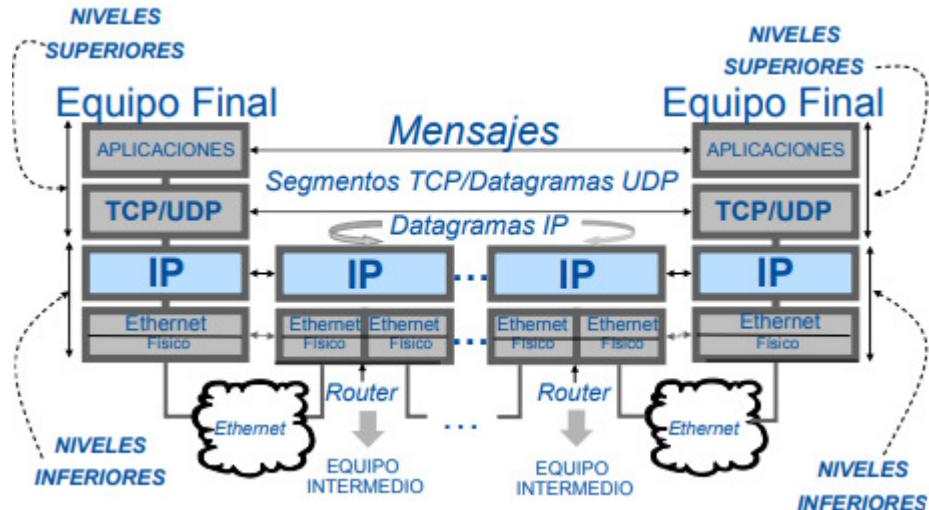


1. Nivel de transporte
2. Nivel de aplicación

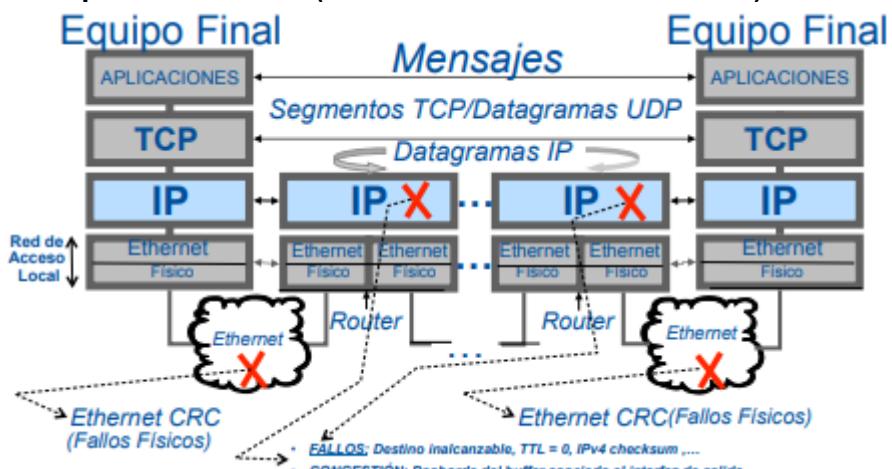


No hay ninguna entidad intermedia de transporte o aplicación en ningún router por el trayecto en Internet.

Las interacciones en el nivel de transporte y aplicación se basan en comunicaciones directas entre dos procesos pares sin intervención de ninguna entidad intermedia.

1. Nivel de transporte:

a. Transporte fiable TCP (*Transmission Control Protocol*):



- Motivos de pérdida de paquete:** CRC (fallos físicos), fallos y congestión en routers (TTL, IPv4 checksum, buffer de salida desbordado, ...), ...
- TCP recupera todos los segmentos TCP perdidos en el nivel de enlace y red y reenvía los segmentos.

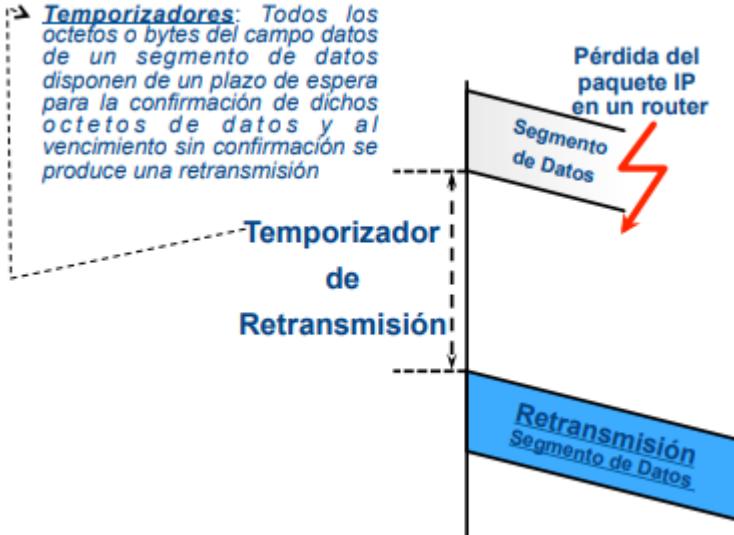
iii. Controles de fiabilidad TCP:

1. Control de fallos o errores:

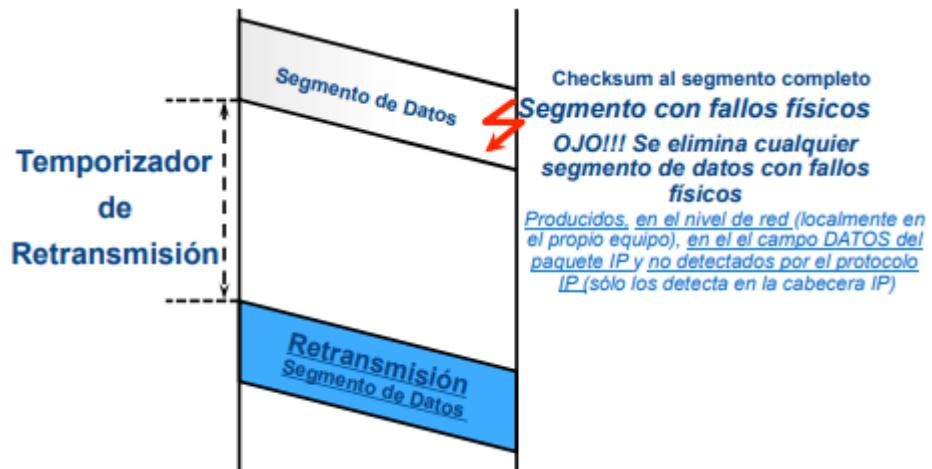
- Lógicos:** bytes perdidos, desordenados o duplicados asociados a los mensajes de aplicación transportados en el campo DATOS de los segmentos TCP.
- Físicos:** bits cambiados en el segmento TCP encapsulado en el campo DATOS de un paquete IP y no detectados por IP (solo detecta fallos físicos en la cabecera IP).
 - Fallo producido en nivel de red en algún router del trayecto.

2. **Control de flujo:** evitar que una entidad o proceso TCP transmita segmentos TCP más rápidamente de lo que otra es capaz de almacenar y procesar.
- iv. **Mecanismos de control de fallos TCP:** todos los bytes de datos del mensaje de aplicación (campo DATOS de segmento TCP) disponen de:
1. **Número de secuencia:** se enumeran todos los bytes del campo datos (no el segmento TCP en conjunto).
 2. **Confirmación:** cuando los bytes se reciben correctamente, se les da un OK (confirmación) (no al segmento TCP, a sus bytes, su contenido).
 3. **Temporizador (plazo de espera de confirmación):** cuando se envía un segmento TCP, se activa el temporizador asociado al campo datos. Si no son confirmados todos los bytes del campo datos, se genera una retransmisión de todos los bytes de datos del segmento.
- v. **Diseño operacional TCP:**
1. El proceso de aplicación montado sobre TCP no delimita sus mensajes, envía un flujo continuo de bytes (**byte-streams**).
 2. La entidad TCP del equipo emisor almacena los **byte-streams** en el **buffer de transmisión**, los enumera y los agrupa en segmentos TCP para su envío a IP.
 3. Por este motivo, TCP no enumera segmentos, sino los bytes en su campo DATOS.
- vi. **Mecanismos de control de fallos:** al enviar un segmento TCP, el equipo emisor activa un temporizador de retransmisión. Si se agota, se retransmite.

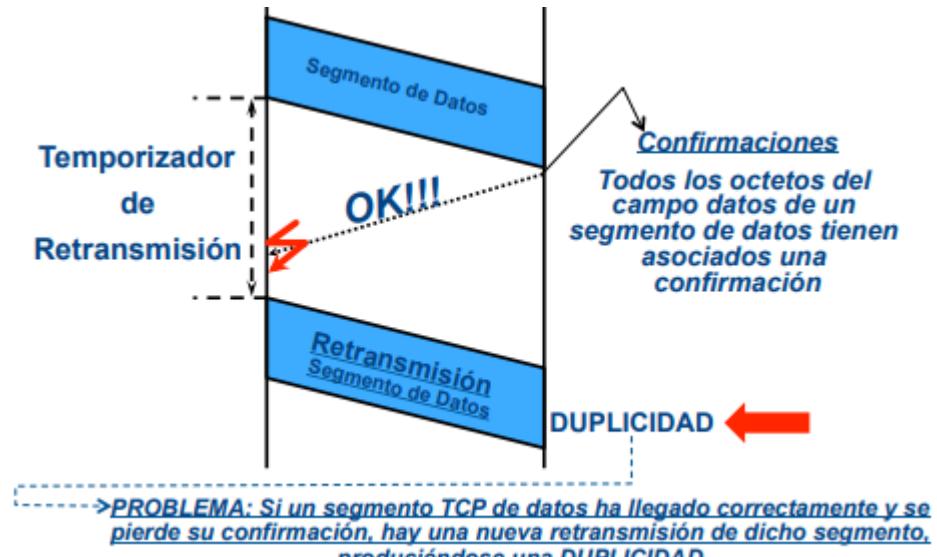
1. **Pérdida de segmento TCP en router (por congestión):**



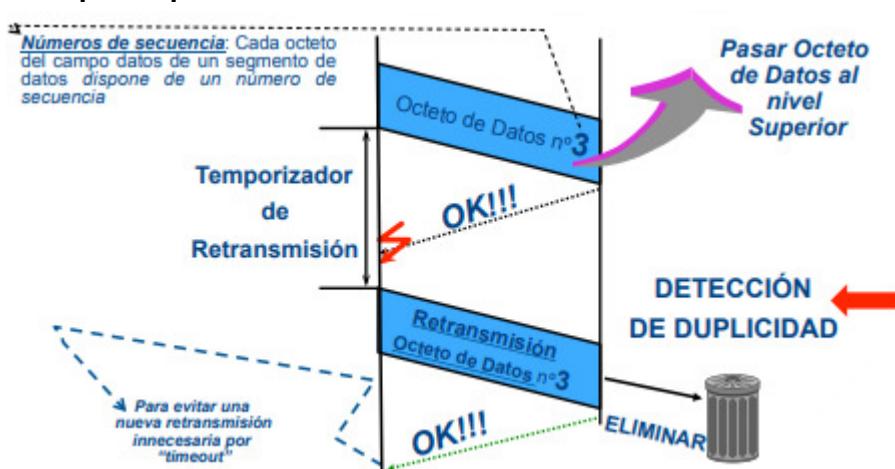
2. Fallo físico (checksum):



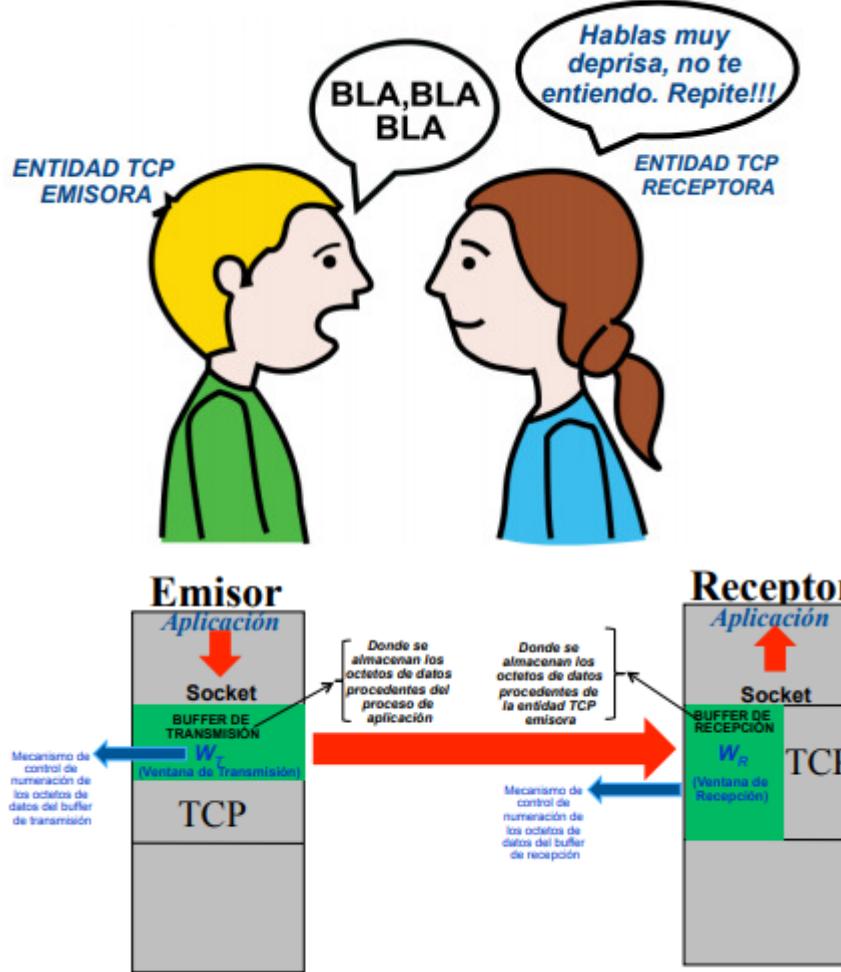
3. Fallo por pérdida de confirmación: congestión en router o fallo físico



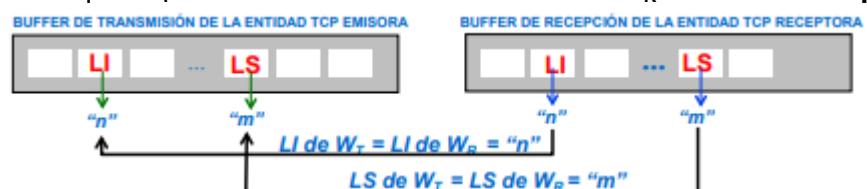
4. Fallo por duplicidad de datos:



vii. Mecanismos de control de flujo:



- Buffer de transmisión/recepción:** almacena/recibe bytes de datos.
- Ventana de transmisión/recepción:** enumera los bytes de datos.
- Si todos los bytes de un segmento TCP ya han sido enumerados, no se espera a que el buffer de transmisión se llene para enviar el segmento.
- Si todos los bytes de un segmento TCP se han recibido correctamente, no se espera a que el buffer de recepción se llene para enviar bytes de datos del segmento al proceso de aplicación.
- Control de flujo TCP:** lo ejerce la entidad TCP receptora mediante W_R , por lo que **W_T en el lado emisor es esclava de W_R en el lado receptor.**



$$W_T = \text{Límite Superior (LS)} - \text{Límite Inferior (LI)} + 1$$

$$W_R = \text{Límite superior (LS)} - \text{Límite Inferior (LI)} + 1$$

Límite Inferior (LI) de W_T = Primer n° de secuencia del primer octeto de datos enviado pendiente de confirmación.

Límite Superior (LS) de W_T = Último n° de secuencia del último octeto de datos enviado pendiente de confirmación

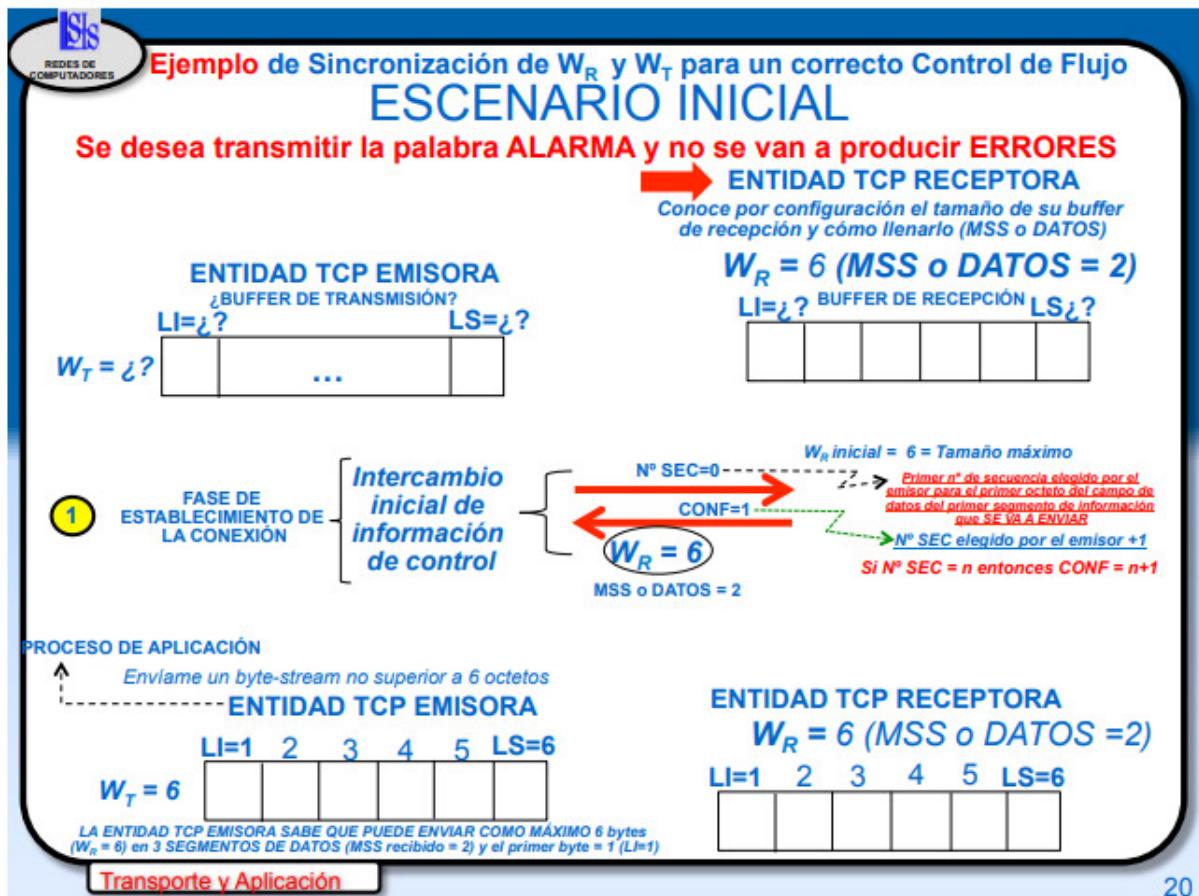
Límite Inferior (LI) de W_R = Primer n° de secuencia del primer octeto de datos esperado.

Límite Superior (LS) de W_R = Último n° de secuencia del último octeto de datos esperado.

- W_R y W_T** se denominan **ventanas deslizantes** porque a medida que van llegando correctamente los bytes de datos a recepción,

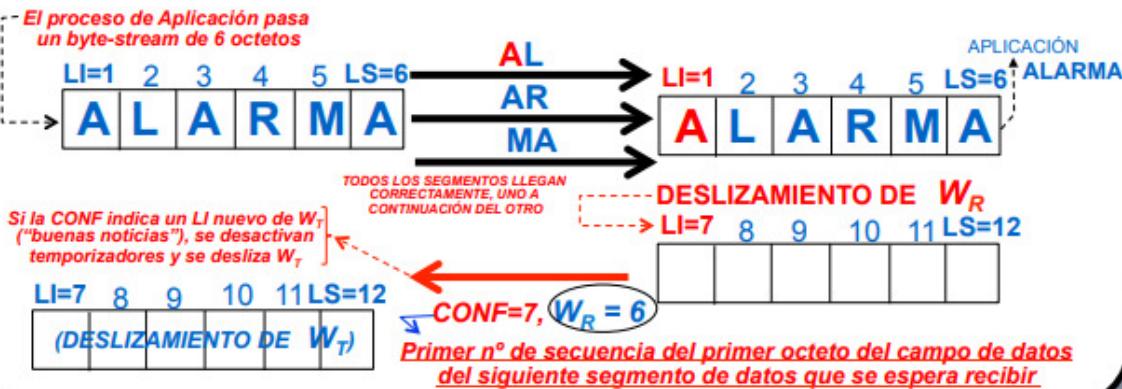
éstos se van **confirmando** y los límites se deslizan o asocian a nuevos números de secuencia.

6. (1) **Fase de establecimiento de la conexión:** intercambio de información de control antes del intercambio de datos.



20

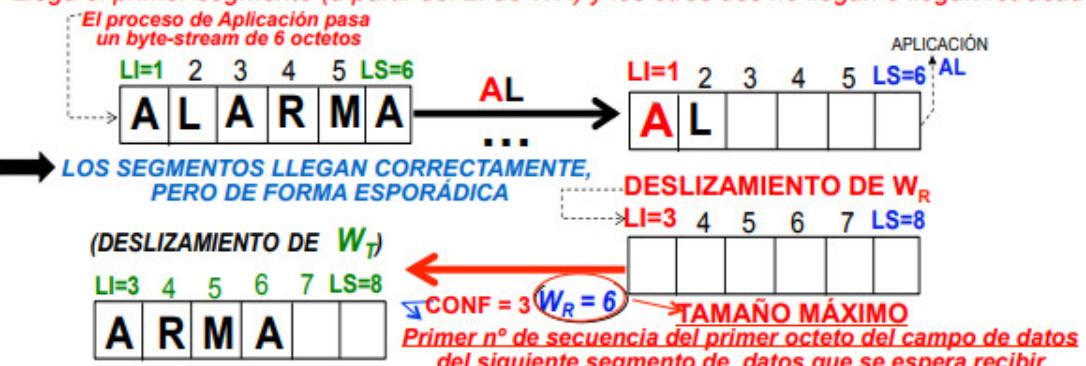
- Emisor:** envía el primer nº de secuencia elegido por el emisor para el primer byte del campo datos del primer segmento de información que se va a enviar ($N^{\circ} SEC = 0$).
 - Receptor:** devuelve el límite inferior de W_R ($CONF = N^{\circ} SEC + 1$) en el campo confirmación, el tamaño del buffer de recepción para que el buffer de transmisión se ajuste a él y el MSS (longitud máxima del campo datos de un segmento TCP).
 - Si $CONF = 100$, se han confirmado hasta el espacio 99 del buffer de recepción.
7. (2) **Fase de transferencia de datos:**
- Caso mejor sin fallos: llegan todos los segmentos seguidos:**



- i. Los segmentos TCP se transmiten de 2 en 2 bytes (MSS=2)
- ii. Cada segmento transporta el nº de secuencia de su primer byte de datos. Si el buffer de recepción detecta un nº de secuencia **repetido** o **fuera de rango** [LI, LS], lo **desecha**.
- iii. Si CONF indica un nuevo LI de W_T (deslizamiento de W_R), confirma que se han aceptado los bytes enviados [1, 6], se desactivan temporizadores y se desliza W_T .

b. Caso peor sin fallos: llegan parte de los segmentos seguidos

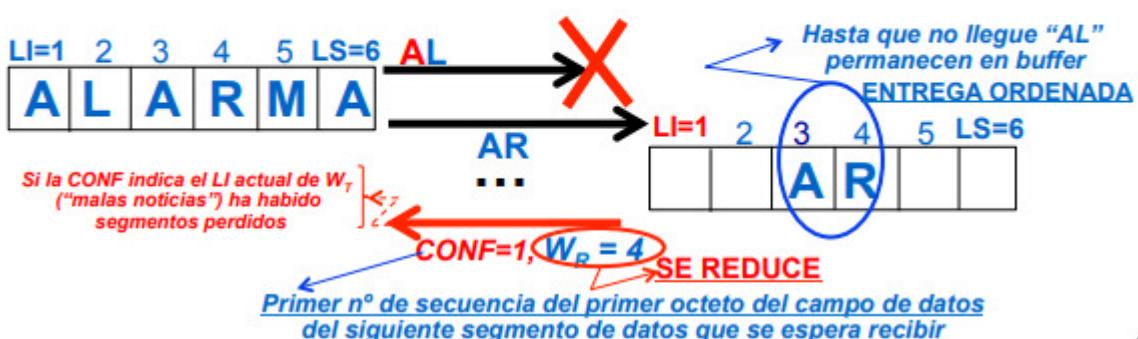
Llega el primer segmento (a partir del LI de WR) y los otros dos no llegan o llegan retrasados



- i. Si se recibe el segmento cuyo nº de secuencia coincide con el límite inferior de W_R , se desliza hasta el siguiente espacio vacío del buffer de recepción (en este caso CONF $\Rightarrow 1 \rightarrow 3$) y W_T sincroniza el deslizamiento.
- ii. Mientras se reciben los segmentos siguientes, podrá subir los recibidos al nivel de aplicación.

c. Caso con fallos: no llegan los segmentos seguidos

LOS SEGMENTOS LLEGAN FUERA DE SECUENCIA POR PÉRDIDAS y el primer octeto de datos recibido no coincide con el LI de WR



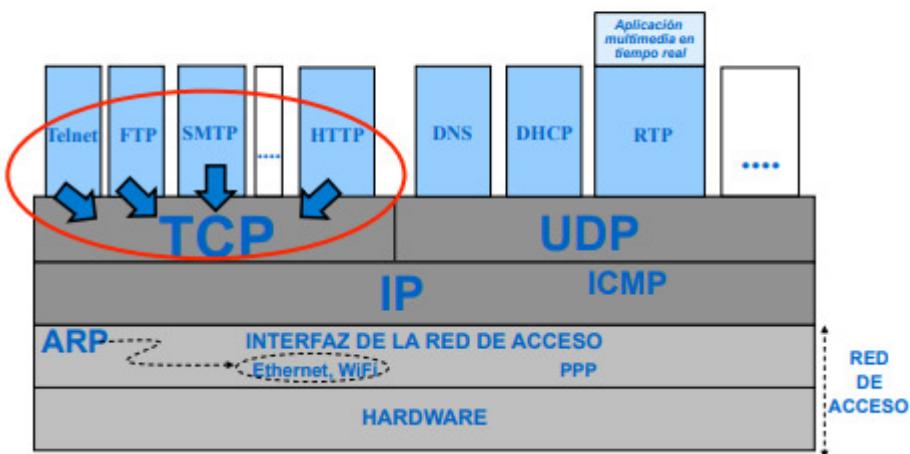
- i. Si **no** se recibe el segmento cuyo nº de secuencia coincide con el límite inferior de W_R , **no** se desplaza W_R , se devuelve el mismo CONF (tampoco se confirman los bytes recibidos) y se **reduce** W_R porque el segmento recibido permanecerá en el buffer de recepción.
- ii. **Entrega ordenada:** hasta que no se reciban los segmentos anteriores al recibido, permanecerá en el buffer y no se subirá al nivel de aplicación.

b. Servicios TCP:

- i. **Flujo de octetos (byte-stream):** el proceso de aplicación montado sobre TCP no delimita sus mensajes, envía un flujo continuo de bytes (**byte-streams**) que la entidad TCP emisora almacena en el **buffer de transmisión**, enumera y agrupa en segmentos TCP para su envío a IP.
- ii. **Fiable:**
 1. **Control de fallos o errores:**
 - a. **Lógicos:** bytes perdidos, desordenados o duplicados del campo DATOS de los segmentos TCP.
 - i. **Detección:** nº de secuencia.
 - ii. **Corrección:** temporizadores y retransmisión.
 - b. **Físicos:** bits cambiados en el segmento TCP (campo datos de IP)
 - i. **Detección:** suma de comprobación (*checksum*).
 - ii. **Corrección:** temporizadores y retransmisión.
 2. **Control de flujo:** mecanismo de ventana deslizante del TCP receptor para que el TCP emisor no desborde el buffer de recepción.
- iii. **Multiplexado:** transmisión simultánea y diferenciada de distintos procesos de aplicación mediante los números de puerto.

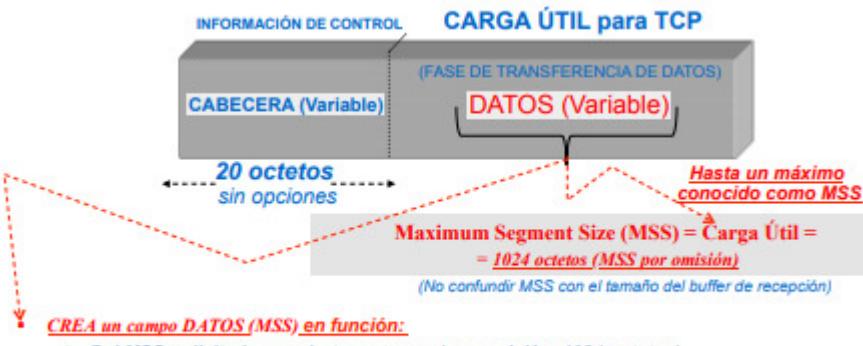
Servicio Multiplexado TCP

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

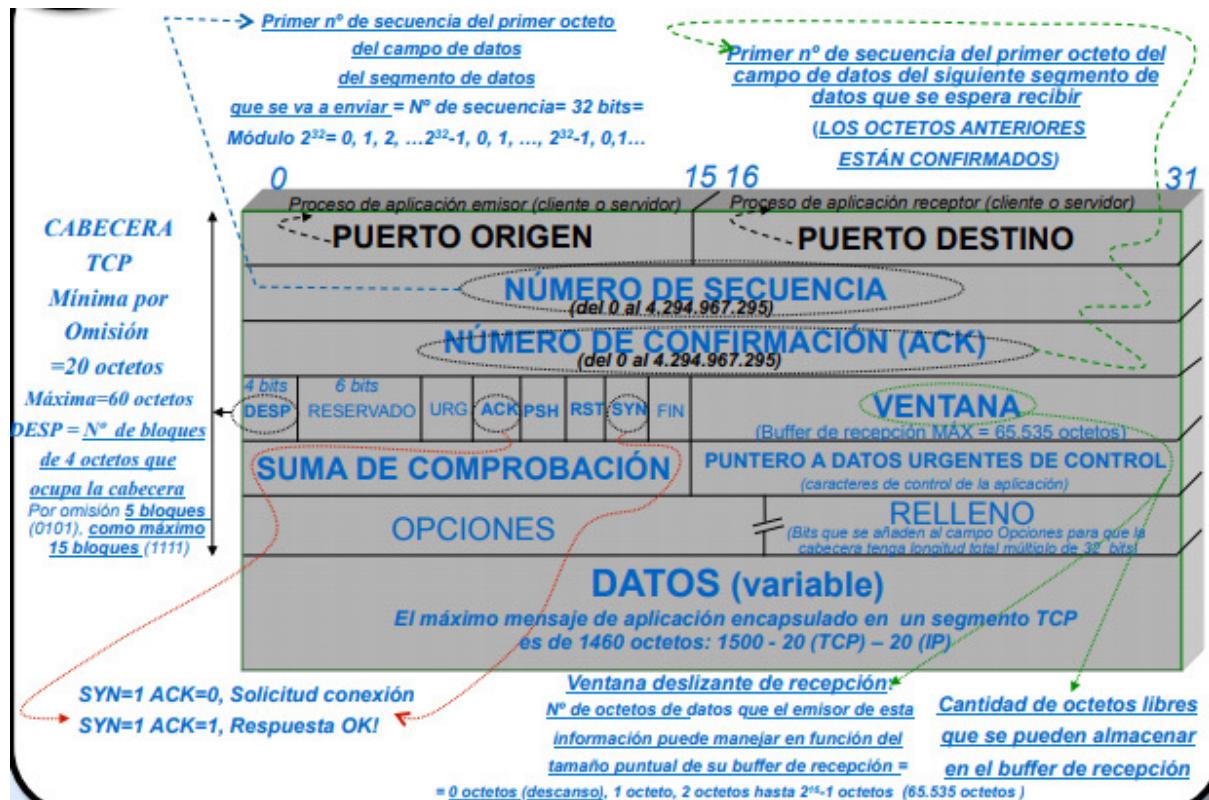


- 1. Cada proceso de aplicación tiene **sus propios recursos** (mecanismos de fiabilidad, buffers de transmisión y recepción, ...).
- iv. **Dúplex:** transferencias simultáneas bidireccionales.

c. Formato de un segmento TCP:

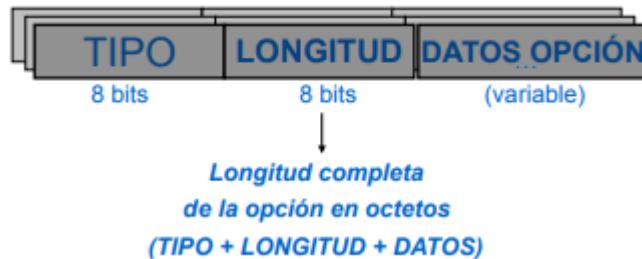


- MSS (Maximum Segment Size):** tamaño máximo del campo DATOS.
- Tamaño máximo DATOS:** 1500 (MTU) - 20 (CPI IP) - 20 (CPI TCP) = **1460 B** (para evitar fragmentación IP).
- Cabecera TCP:**



1. **DESP** = nº de bloques de 4 bytes que ocupan la cabecera [5 - 15].
2. **Nº secuencia** = identificación del primer byte del campo DATOS.
3. **Nº confirmación** = confirmación de entidad TCP receptora.
4. **URG** = bit de datos urgentes de control → nº de secuencia + puntero a datos urgentes de control = último byte de datos urgentes (tratamiento especial por la aplicación). Delimita datos urgentes de datos normales.
5. **ACK** = bit de nº de confirmación.
6. **SYN** = bit de sincronización → establecimiento de una conexión TCP.
 - a. **Solicitud conexión TCP:** SYN = 1, ACK = 0
 - b. **Respuesta OK a solicitud TCP:** SYN = 1, ACK = 1

7. **RST** = bit de RESET → abandonar la comunicación y borrar todo lo almacenado en los buffers de transmisión y recepción, o respuesta para **rechazar solicitud TCP**.
8. **FIN** = bit de finalización o liberación de conexión establecida, una vez transmitidos todos los datos en fase de transferencia de datos.
9. **Ventana** = nº de bytes que el buffer de recepción puede manejar. Si ventana = 0, el buffer necesita reposo para liberar espacio.
10. **PSH** = bit de PUSH (empuje) → el proceso de aplicación indica a la entidad TCP emisora que fuerce **envíos rápidos** de mensajes de aplicación para recibir **respuestas rápidas** a los envíos.
 - a. **Envíos rápidos**: evitar que la entidad TCP emisora espere más bytes para construir un segmento mayor según MSS y que la entidad TCP receptora reciba cuanto antes los bytes y pasarlos al proceso de aplicación (sin almacenarlos en buffer de recepción).
 - i. **Ejemplo**: getFichero() → descarga rápida de página web.
 - b. **Respuestas rápidas**: a un envío rápido transmitido previamente.
11. **Opciones TCP**: se especifican en fase de establecimiento de conexión (cuando bit SYN = 1).
 - a. **Formato TLV**: Tipo-Longitud-Valor



b. Descripción:

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

- c. **Tamaño máximo de segmento (MSS)**: número máximo de bytes del campo DATOS. Por omisión, 1024 bytes. No confundir con el tamaño del buffer de recepción (W_R).
- d. **Factor de escala de la ventana**: permite ampliar el campo VENTANA de 16 bits (2^{16} bytes) a 30 bits (2^{30} bytes = 1 Gbyte), multiplicando su valor por el de la ventana (no varía su longitud).

- e. **Sello de tiempo (timestamp):** permite al emisor configurar de forma más exacta sus temporizadores de espera de confirmación.

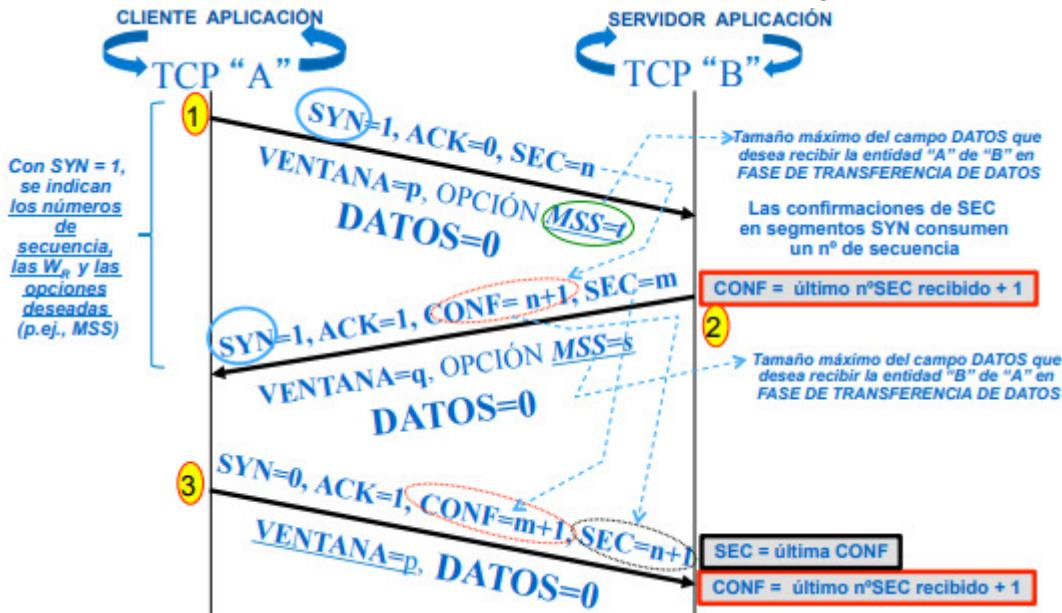
Tipo = 8	Longitud = 10	Valor actual del reloj emisor	Eco del valor actual del reloj emisor
----------	---------------	-------------------------------	---------------------------------------

- i. **Emisor:** pone el valor de su reloj.
 - ii. **Receptor:** pone el valor de su reloj y devuelve el valor del reloj del emisor (eco) en el ACK del segmento.
 - iii. El eco es válido si bit ACK = 1.
- iv. **Temporizadores de espera de confirmación:** su valor se calcula según algoritmos autoadaptativos que ajustan sus valores a la dispersión geográfica y al estado de la red.
1. **Algoritmo autoadaptativo de Karn para el cálculo del RTT (Round Trip Time):** se toma el promedio de varios RTT (envío de segmento y recepción de confirmación) y se le añade un **margin de seguridad**.

d. **Fases del servicio fiable de TCP (servicio orientado a conexión):**

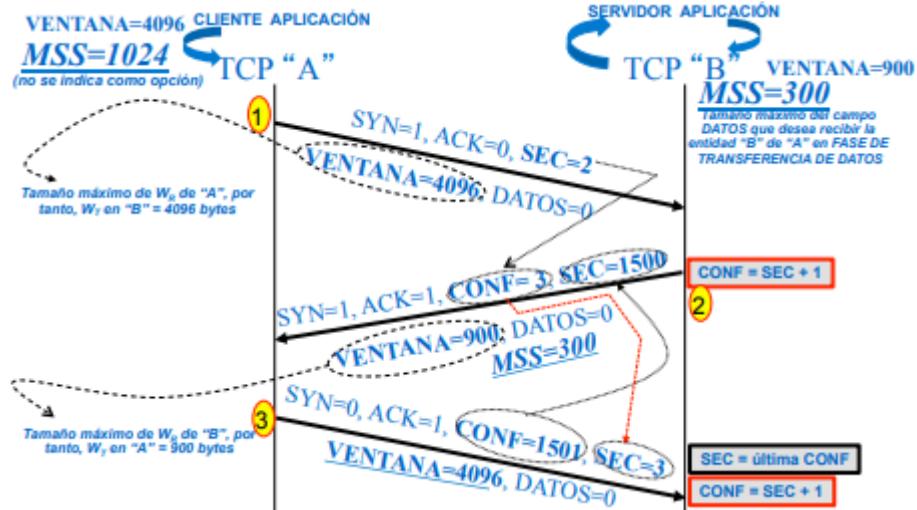
- i. Establecimiento de conexión
- ii. Transferencia de datos
- iii. Liberación de conexión

e. **Fase de establecimiento de conexión TCP:** intercambio 3 segmentos sin datos



- i. **Primer segmento:** $(\text{SYN}, \text{ACK}) = (1, 0)$ = solicitud de conexión TCP.
Tamaño de $W_T = p$
- ii. **Segundo segmento:** $(\text{SYN}, \text{ACK}) = (1, 1)$ = solicitud aceptada.
Tamaño de $W_R = q$
- iii. **Tercer segmento:** $(\text{SYN}, \text{ACK}) = (0, 1)$ = sincronización/negociación finalizada
- iv. **SEC inicial:** generado aleatoriamente.
- v. **CONF = SEC anterior + 1**
- vi. **SEC no inicial = CONF anterior**
- vii. **ACK:** confirma los parámetros recibidos previamente.
- viii. **MSS:** máximo de bytes de datos que puede recibir su buffer por segmento.
- ix. **Rechazo de solicitud de conexión TCP** → bit RST (reset) activado.

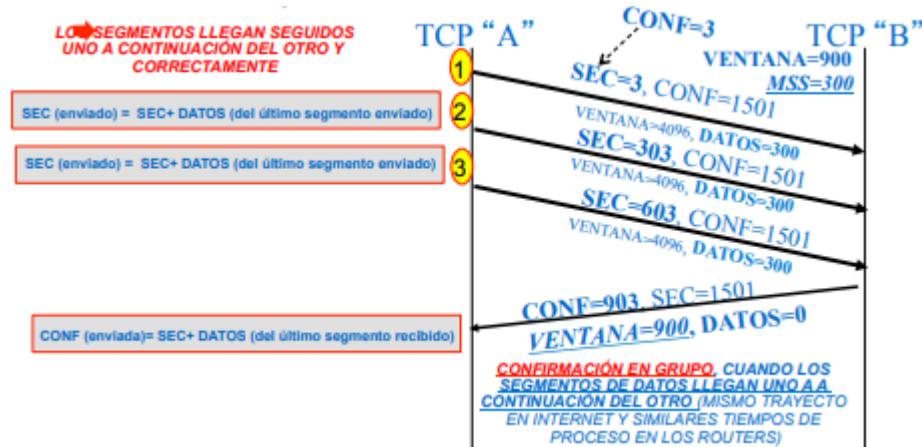
x. Ejemplo:



1. SEC es independiente a VENTANA (puede ser mayor).
 2. MSS sí es dependiente de VENTNA (debe ser menor).

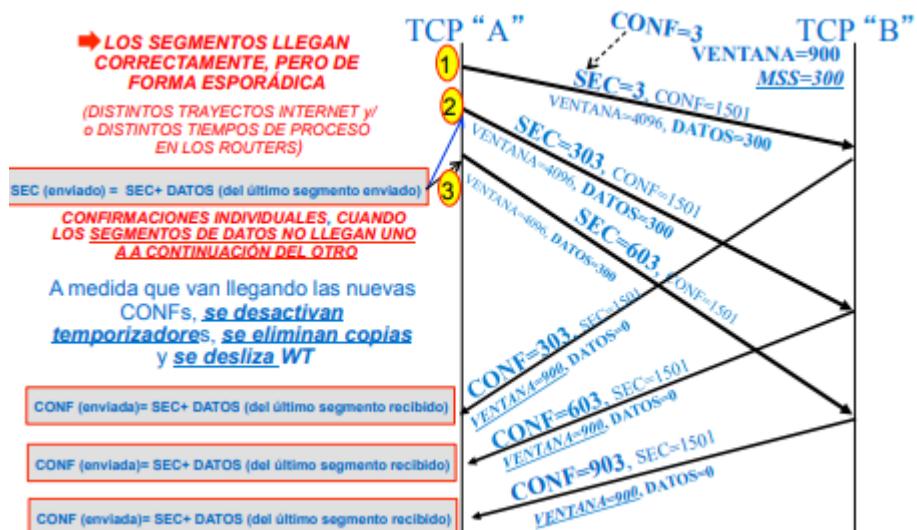
f. **Fase de transferencia de datos en conexión TCP:** transferencia unidireccional

i. Transferencia seguida sin pérdidas + confirmaciones en grupo

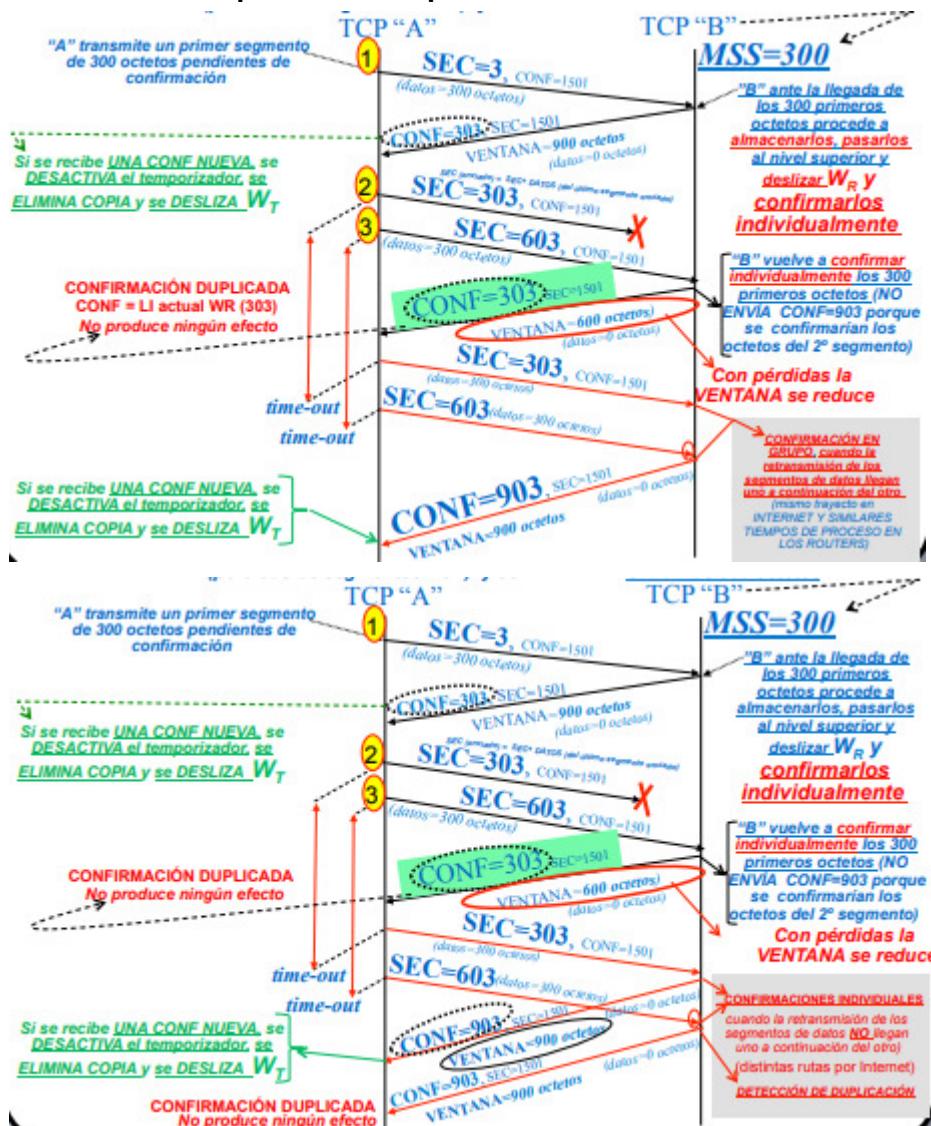


1. **Emisor:** SEC sucesivos ($3;+300=303;+300=603$).
 2. **Receptor:** SEC confirma último segmento (SEC = $3 + 3 \times \text{DATOS} = 900$)

ii. Transferencia esporádica sin pérdidas + confirmaciones individuales



1. Receptor: SEC confirma cada segmento ($SEC + DATOS = 300$).
 iii. Transferencia esporádica con pérdidas:

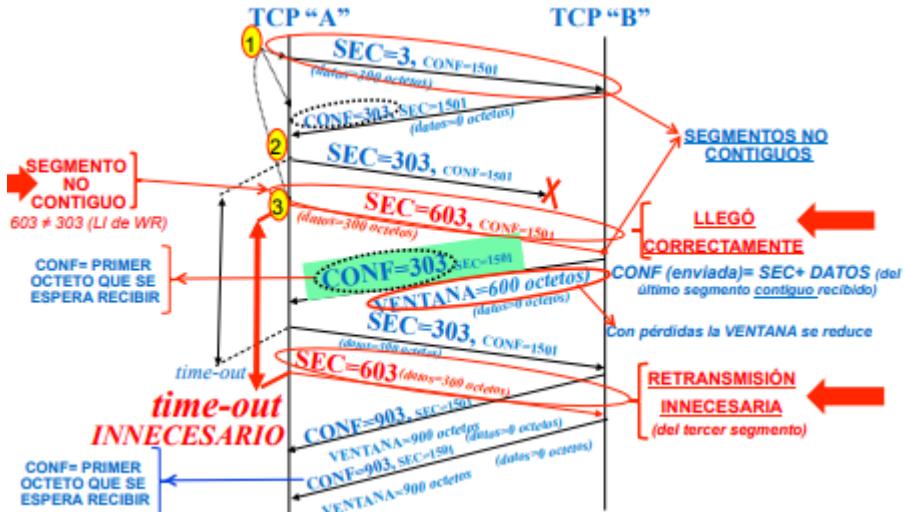


1. Solo si $SEC = LI$ de W_R , se recibe una nueva CONF, se desactiva el temporizador, se elimina la copia y se desliza W_T (como W_R).
2. Si no recibe el segmento sucesivo, su confirmación mantendrá CONF y disminuirá VENTANA ($900 - DATOS = 600$).
3. Cuando se agoten los temporizadores de los segmentos posteriores a CONF, se retransmitirán. Como el segmento SEC=606 ya está en el buffer de recepción, el equipo receptor lo desechará por duplicidad.

iv. Opciones TCP (continuación): SACK (Selective Acknowledgement) o confirmación selectiva.

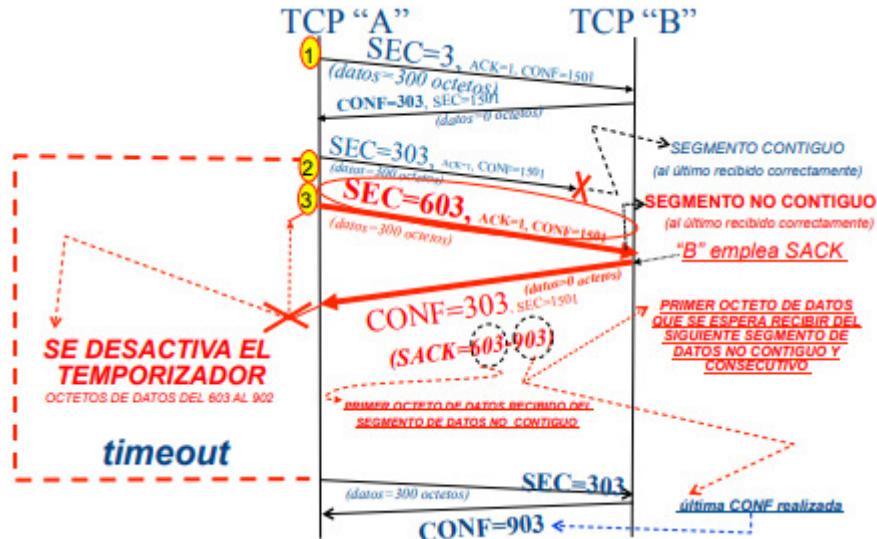
TIPO	LONGITUD (octetos)	DATOS	Descripción
2	4	Valor MSS	MSS
3	3	Tamaño de la Ventana	Escala de la Ventana
4 <i>Fase de Establecimiento de la Conexión</i>	2	DATOS = 0	<u>SACK PERMITTED</u>
<i>Fase de Transferencia de Datos</i>	Variable	$X_1 - Y_1 \dots X_n - Y_n$	<u>SACK</u>
8	10	Valor actual reloj emisor (4 octetos) + Respuesta Echo (4 octetos)	Marca de Tiempo
...

1. Para usar la opción Tipo 5 SACK (fase de transferencia de datos), se **debe** indicar previamente con la opción Tipo 4 SACK-PERMITTED (fase de establecimiento de conexión) en un segmento con bit SYN = 1.
2. Aumenta la eficiencia de protocolo TCP en caso de perder el segmento contiguo y recepción correcta de un **segmento no contiguo** porque evita el vencimiento de temporizadores y retransmisiones innecesarias (reduce el tráfico y proceso en nivel de enlace y red en los routers).
 - a. **Segmento no contiguo:** cuando su SEC no es el primero que se espera recibir ($SEC \neq LI$ de W_R).
3. **Ejemplo (transferencia esporádica con pérdidas):** la retransmisión del segmento TCP de nº SEC = 603 es innecesaria dado que ya está almacenado en el buffer de recepción.



4. **Solución:** siempre que la entidad TCP receptor reciba un segmento TCP **no contiguo**:
 - 1) Almacena los datos.
 - 2) Transmite segmento TCP de confirmación selectiva (SACK sin datos que indica los 300 bytes confirmados: "SEC - SEC+DATOS" = 603-903)
 - 3) CONF = 303 (LI de W_R) → SEC de segmento que se espera recibir.

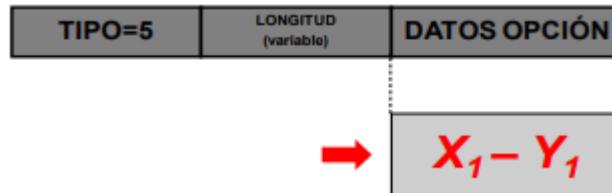
4) Reduce contenido de campo VENTANA (900 - 300 (DATOS) = 600).



5. La entidad TCP emisora sabe que se ha producido pérdida (menor W_R) y que puede desactivar el temporizador del segmento de datos confirmados ($SEC = 603$).

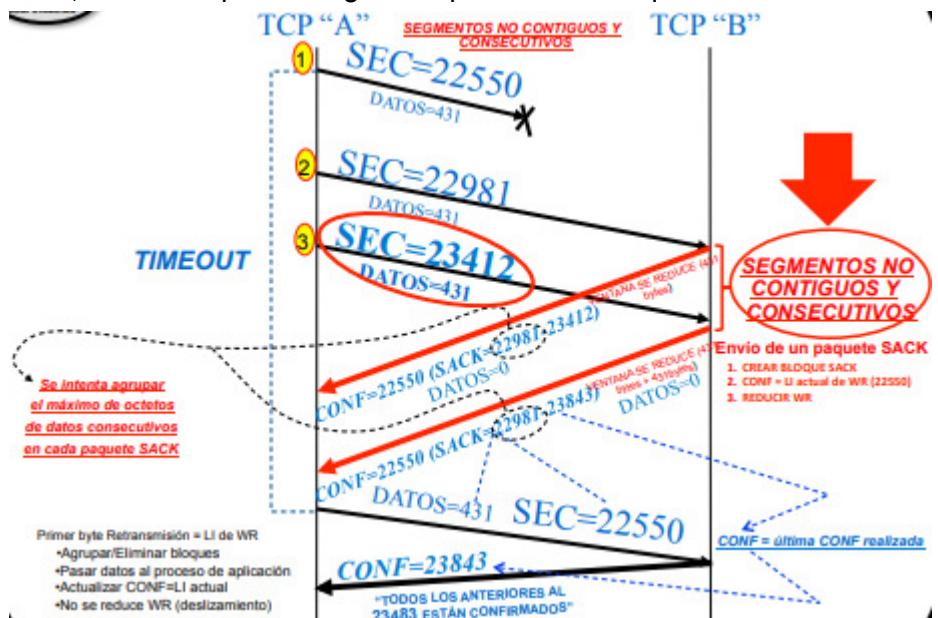
6. Tipos de SACK

- a. Para segmentos no contiguos y consecutivos (sin pérdidas): agrupa SEC de segmentos consecutivos y los confirma en bloque

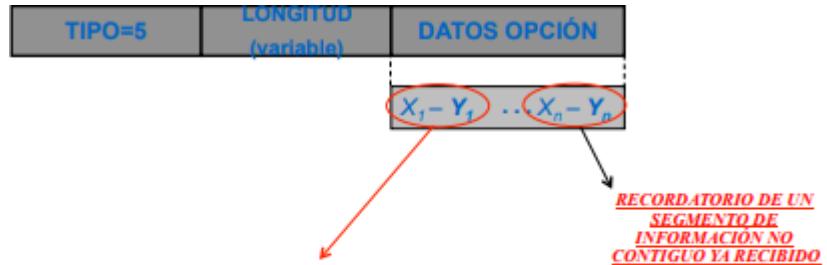


X_1 = SEC de primer segmento de bloque.

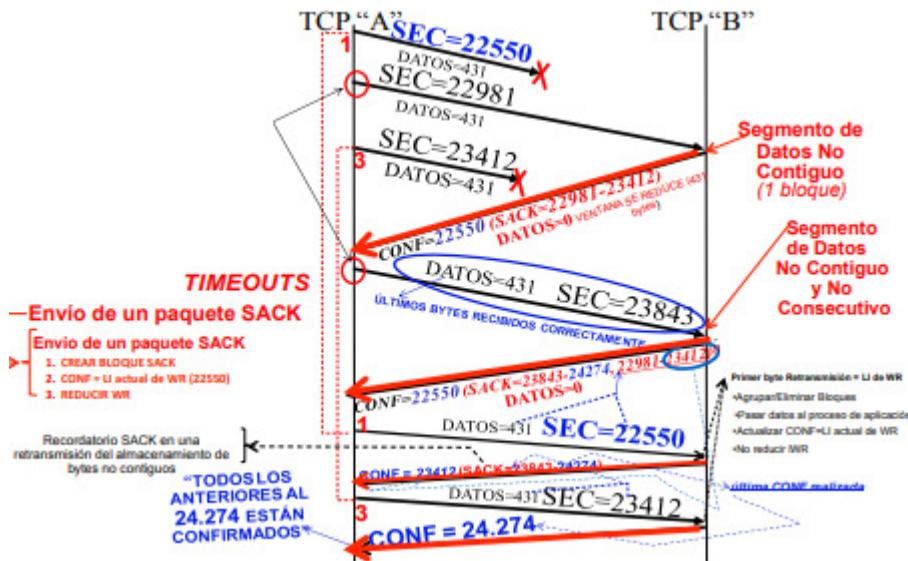
Y_1 = SEC de primer segmento posterior al bloque.



b. Para segmentos no contiguos y no consecutivos (pérdidas):



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



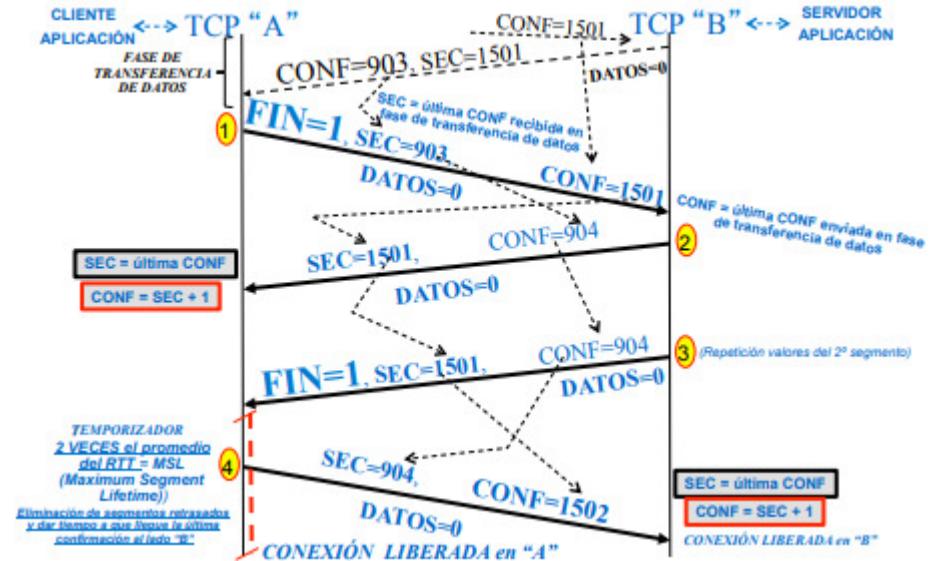
- Los SACK de los segmentos 22981 y 23843, desactivan sus temporizadores de retransmisión.
- Los segmentos 22550 y 23412 se retransmiten al no incluir el SACK sus datos.
- Cuando se recibe el segmento 22550, se desplaza **ventana** hasta 23412 (pasando los segmentos 22550 y 22981). Lo mismo sucede cuando se recibe el segmento 23412.

g. Fase de liberación de conexión TCP:

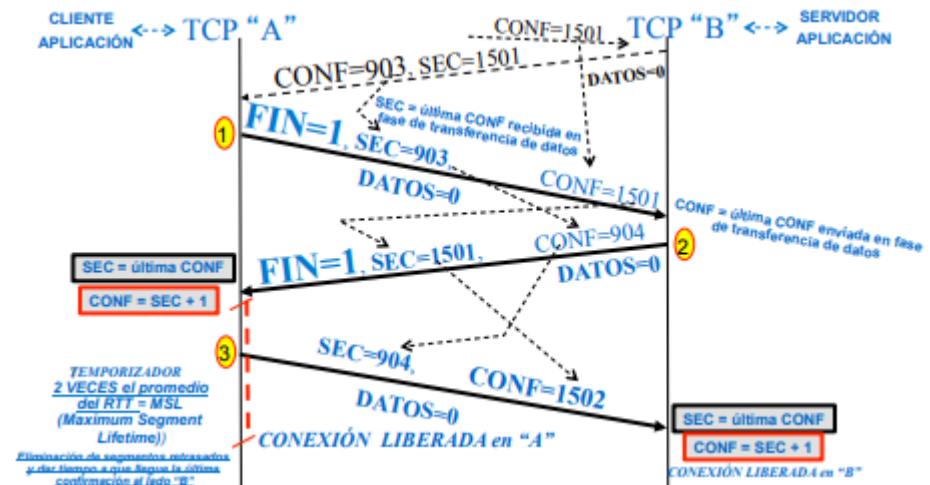
- (Tipos) Liberación unilateral o abrupta:** la conexión TCP se libera cuando una entidad TCP envía un segmento TCP sin datos con el bit RST activado, sin recibir la confirmación de la otra entidad TCP, que estará obligada a cerrar la conexión y borrar todos los datos en buffers.
- (Tipos) Liberación bilateral u ordenada:** la conexión TCP se libera cuando cada entidad TCP, después de haber transmitido todos sus datos, envía en cada sentido un segmento sin datos con el bit FIN activado.
 - Liberación basada en 4 envíos (estándar de liberación):**
 - Enviar segmento TCP con bit FIN.
 - Confirmar recepción de segmento.
 - Devolver segmento TCP con bit FIN y los valores del segmento 2. La recepción de este segmento activa un **temporizador de desconexión** ($2x$ promedio de RTT = **MSL (Maximum Segment Lifetime)**) → tiempo

para eliminar segmentos retrasados y dar tiempo a que llegue la última confirmación a la otra entidad TCP).

4) Último segmento pendiente de enviar (OK explícito al cierre).



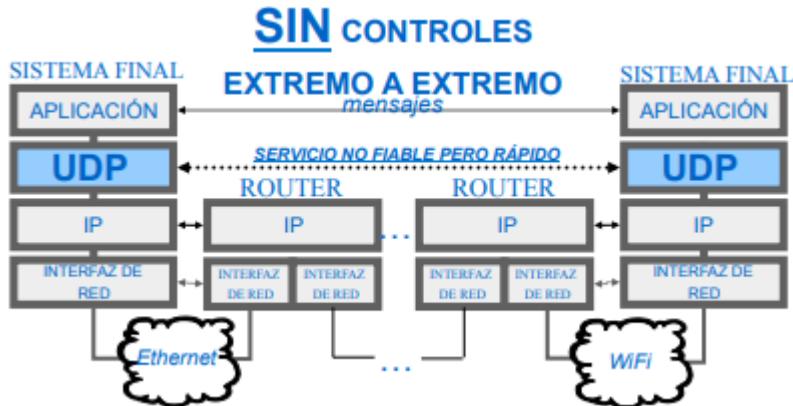
2. Liberación basada en 3 envíos: igual que en la liberación basada en 4 envíos, pero se elimina la primera confirmación de segmento (2).



iii. Problemática de fiabilidad TCP:

- **TCP ofrece un servicio FIABLE pero "LENTO" en función de que ofrece un servicio orientado a conexión (3 FASES)**
- **Las funciones de FIABILIDAD, con todos sus mecanismos implementados en la cabecera TCP y las retransmisiones; incrementan el retardo extremo a extremo y hacen que el transporte de los mensajes de aplicación sea LENTO**
 - ✓ **Hay aplicaciones, especialmente en TIEMPO REAL, muy sensibles a los retardos y no toleran el retardo extremo a extremo producido por las confirmaciones, los temporizadores, las retransmisiones y controles de TCP.**
- **Si la aplicación requiere un TRANSPORTE RÁPIDO SIN FIABILIDAD se monta sobre UDP**
 - ✓ **SIEMPRE EN FASE DE TRANSFERENCIA NO FIABLE DE DATOS**

- h. Transporte rápido UDP (*User Datagram Protocol*): no fiable (**sin controles**) pero rápido de mensajes de aplicación, encapsulados en datagramas UDP.



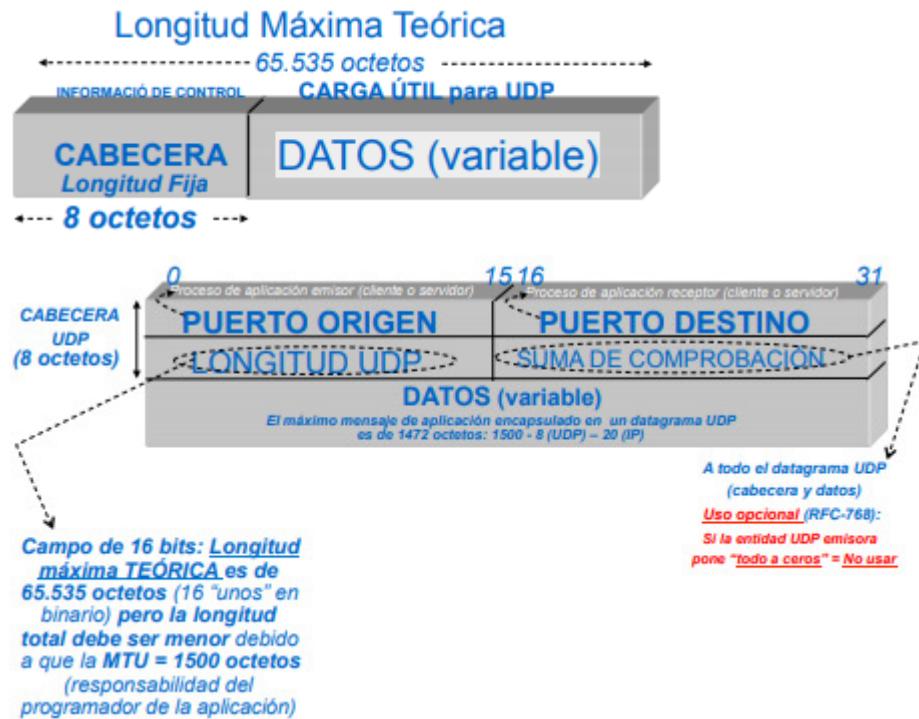
i. Uso de UDP:

- ✓ En aplicaciones en tiempo real
 - ✓ interactivas (videoconferencias o videollamadas, VoIP, etc.) y no interactivas (streaming de audio y vídeo, etc.)
- ✓ Cuando el intercambio de mensajes es muy escaso y los mensajes son cortos, se producen regularmente y no importa si se pierde alguno
- ✓ Envíos DHCP, consultas DNS, mensajes SNMP de administración de una red de comunicaciones, mensajes NTP para sincronizar relojes de equipos, etc.
- ✓ Cuando se envía tráfico de broadcast (establecer y liberar conexiones con todos los equipos vecinos)

ii. Servicios UDP:

1. No fiable:
 - a. Detección opcional y no recuperación de **errores físicos**.
 - b. Sin control de **errores lógicos** (ni detección ni recuperación).
 - c. Sin **control de flujo**.
 - d. **Servicio no orientado a conexión**: siempre está en **fase de transferencia** no fiable de datos.
2. **Multiplexado**: transmisión simultánea y diferenciada de distintos procesos de aplicación mediante los números de puerto.
3. **Dúplex**: transferencias simultáneas bidireccionales.

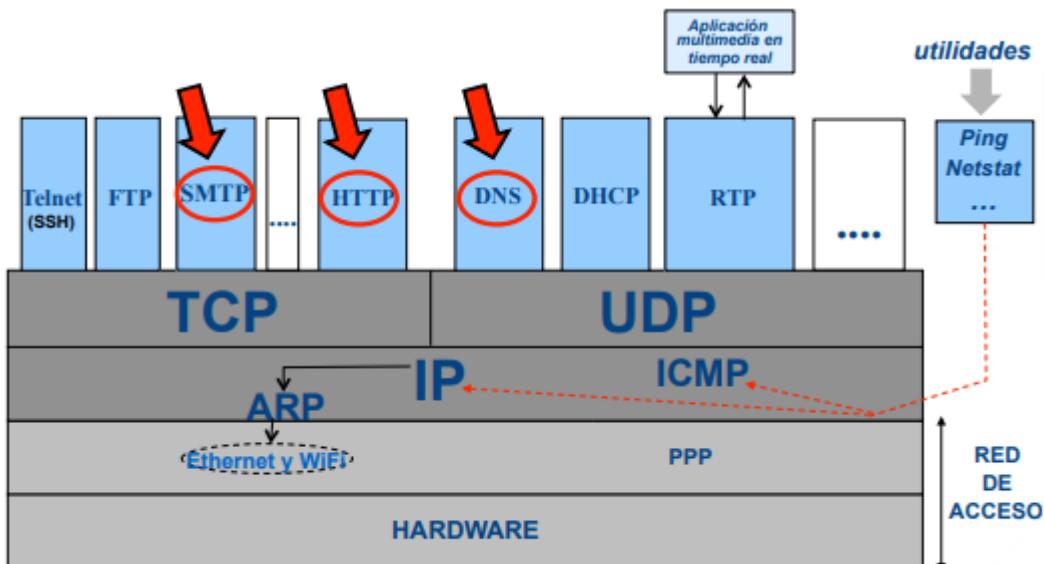
iii. **Formato datagrama UDP:** cabecera no tiene **opciones** (longitud fija).



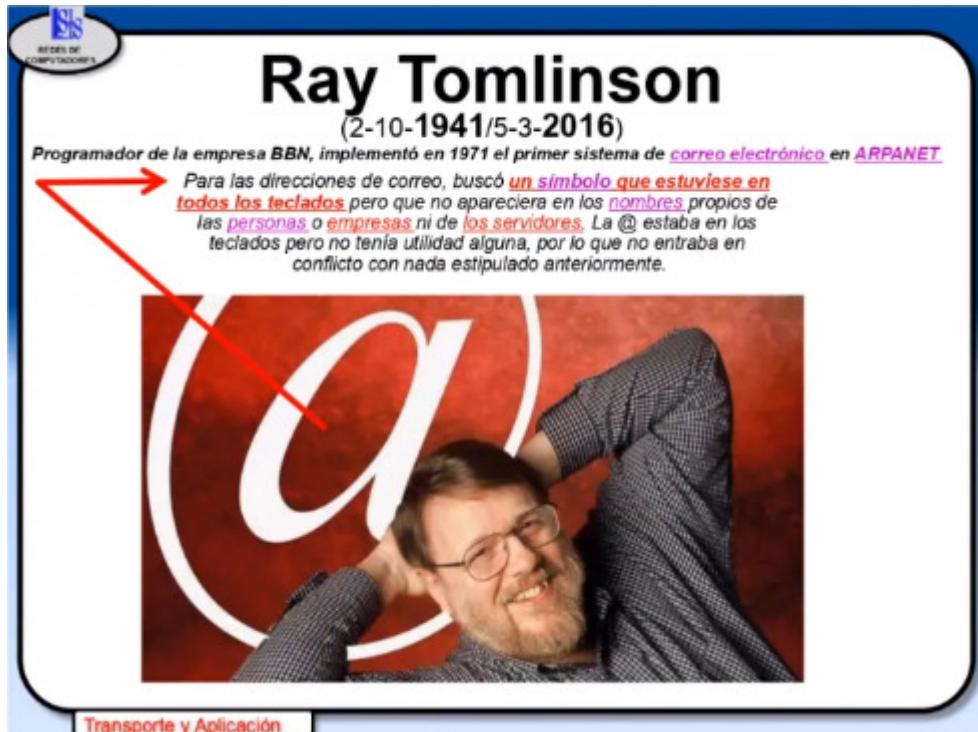
1. TCP ajusta la longitud de su campo datos para que IP no tenga que fragmentar su segmento (MTU = 1.500 bytes). UDP no lo ajusta, mete hasta (65.535 - 8) bytes e IP se ocupa de fragmentar su datagrama.
2. **Suma de comprobación:** uso opcional. Si no se desea que se realice la suma de comprobación (para aumentar velocidad de transporte), se pone todo a ceros y la entidad receptora no la calcula.

2. Nivel de aplicación

a. Protocolos de aplicación montados sobre TCP/UDP:



b. Correo electrónico [TCP]:



i. Componentes:

1. **Agente de usuario:** entorno de correo en el equipo del usuario (por ejemplo, Microsoft, Outlook, Kmail, Evolution, ...).
 - a. **Editor de texto:** componer mensajes.
 - b. **Cliente POP3/IMAP4:** acceder al buzón y leer mensajes.
 - c. **Codificador/decodificador (CODEC) MIME:** enviar y recibir contenido en un mensaje de correo.
 - d. **Cliente SMTP:** enviar correos.
2. **Servidor de correo SMTP:** gestiona el buzón de correo de cada usuario.
 - a. Ejecutado en el equipo de la organización del usuario o en la red IP de su operador (ISP).
3. **Protocolo SMTP (*Simple Mail Transfer Protocol*):** comunica al cliente con el servidor de correo para enviar correo por Internet.

ii. Ejemplo:



0) **Configuración previa:** manual → ID/contraseña, dirección de correo,...

1) **Traducción DNS smtp.origen.com:** IP 210.1.0.2

1bis) **Establecimiento de conexión:** envía nombre de usuario y contraseña al servidor SMTP para acceder al **buzón** de correo que controla.

2) **Composición de mensaje a enviar:** usuario emisor, receptor y mensaje.

3) **Transferencia de correo:** de cliente SMTP a servidor SMTP local.

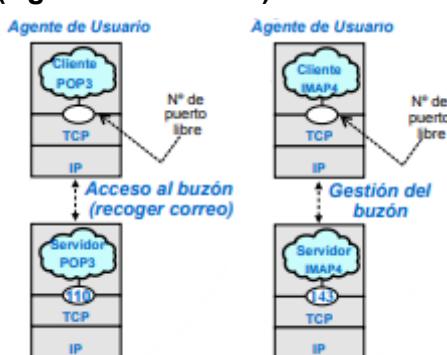
4) **Traducción DNS smtp.destino.com:** no tiene su IP, pero sí la IP de un servidor de intercambio de correo electrónico 138.100.0.1 (mx.destino.com).

- **Servidor de intercambio de correo electrónico:** almacena previamente los mensajes para el servidor SMTP remoto destinatario si este no está ejecutándose o si el equipo no está encendido.

5) **Envío de mensaje SMTP (correo) a mx.destino.com.**

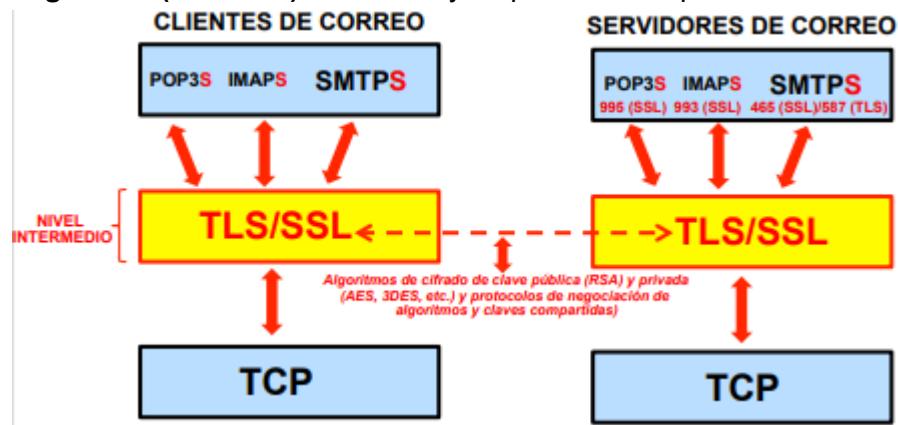
6) **Envío de mensaje SMTP a smtp.destino.com:** el servidor de intercambio de correo electrónico (mx.destino.com) pasa de servidor a cliente SMTP. El servidor SMTP remoto comprobará que la parte derecha del @ coincide con su dirección simbólica (smtp.destino.com).

iii. (Agente de usuario) POP3 e IMAP4:

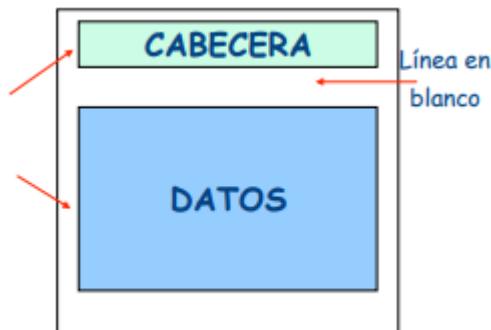


- iv. **POP3 (Post Office Version 3):** servicio de **recogida del correo electrónico** en el buzón del usuario, del servidor de correo al disco duro del equipo local.
 - 1. Siempre que se quiera realizar alguna gestión en el correo electrónico, se deberán llevar todos los mensajes al disco duro local.
- v. **IMAP4 (Internet Message Access Protocol 4):** servicio de **gestión del correo electrónico** que permite al usuario clasificar, eliminar y distribuir su correo en el disco duro del **equipo servidor** de correo. Al contrario de POP3, no recoge los mensajes y los trae al disco duro local, aunque permite copiar o mover mensajes desde el buzón hasta el disco duro local.
 - 1. Con IMAP4, los correos no se leen a través de la red, sino que lleva las cabeceras temporalmente al disco duro local. Cuando el usuario selecciona la cabecera de un mensaje, descarga sus datos.
- vi. **Nº de puerto de servidores SMTP:**
 - 1. **25:** nº puerto servidor SMTP **no seguro.**
 - El nº de puerto 25 fue el primer nº de puerto utilizado y es el estándar, por omisión, para identificar a un servidor SMTP no seguro
 - Aún lo siguen utilizando los operadores en sus servidores SMTP y en Servidores de Intercambio
 - Permisivo
 - Autenticación no segura: Identificador de usuario y contraseña visible (vía un analizador de protocolos: Wireshark o Tcpdump)
 - Reenvía cualquier mensaje transmitido por su cliente SMTP
 - ✓ Peligro de que el cliente SMTP transmita voluntariamente o involuntariamente por infección previa: spam o virus
 - ✓ Peligro de que el servidor de correo origen entre en una Lista Negra de Servidores SMTP y sus correos sean rechazados
 - 2. **465:** nº puerto servidor SMTP **seguro** usado por Arquitectura de Seguridad **SSL (Secure Socket Layer).**
 - Aunque SSL está siendo reemplazado por la Arquitectura de Seguridad TLS (Transport Layer Security), aún se sigue usando en muchas organizaciones (p.ej., la ETSII) y en los servidores SMTP de los operadores
 - ✓ Por tanto hay muchos usuarios que tienen sus clientes SMTP configurados para usarlo con servidores SMTP 465.
 - Autenticación segura: Cifrado del Identificador de usuario y Contraseña
 - Cifrado de todo el correo enviado del cliente SMTP al servidor SMTP y viceversa
 - Permite el uso de Firewalls para filtrar direcciones IP de clientes SMTP para el puerto 4656
 - Permite el uso de Filtros Antispam, Antivirus y Listas Negras de Servidores SMTP
 - 3. **587:** nº puerto servidor SMTP **seguro** basado en Arquitectura de Seguridad **TLS (Transport Layer Security).**
 - TLS se basa en SSL y ambos son compatibles. TLS (v1.3) es más seguro, flexible y eficiente que su antecesor SSL (v3.0).
 - 4. **2525:** nº puerto opcional servidor SMTP **seguro** basado en **TLS.**
 - a. **Puerto alternativo al 587**, si este nº puerto está bloqueado.
- vii. **Puertos Seguros de Servidores de Correo:** para poder cifrar y descifrar los mensajes de correo y autenticación previa entre cliente y servidor SMTP, el cliente y el servidor SMTPS deben disponer de un **Nivel Intermedio de**

Seguridad (TLS/SSL) entre TCP y su proceso de aplicación SMTP.



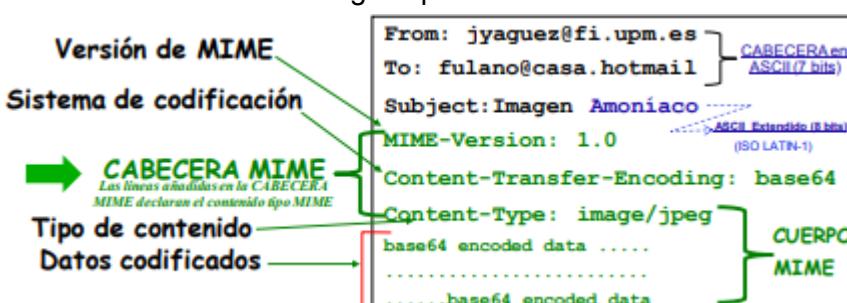
viii. Formato mensaje SMTP:



1. **Cabecera:** *To:, Cc:, Bcc:, From:, Subject:, ...*
2. **(línea en blanco)**
3. **Datos:** mensaje del correo en formato **ASCII de 7 bits**. Para enviar mensajes NO ASCII (tildes, diéresis, ficheros multimedia, ...) se utiliza el **CODEC MIME** → añade a DATOS cabecera MIME y cuerpo MIME.

ix. CODEC MIME (*Multipurpose Internet Mail Extensions*): define el formato del campo DATOS del mensaje de correo añadiendo una cabecera MIME y un cuerpo MIME tanto al texto como a cada fichero en el correo (*attachment*).

1. **Sistema de codificación:** base64 o radix64 (subconjunto de 6 bits del ASCII de 7 bits; $2^6 = 64$ caracteres base64) → se sustituyen grupos de 6 bits del texto o fichero original por un carácter base64.



- c. Sistema DNS (*Domain Name System*) [UDP]:

El Sistema DNS

Domain Name System

RFC-1034 y RFC-1035

Paul Mockapetris

Enero de 1985

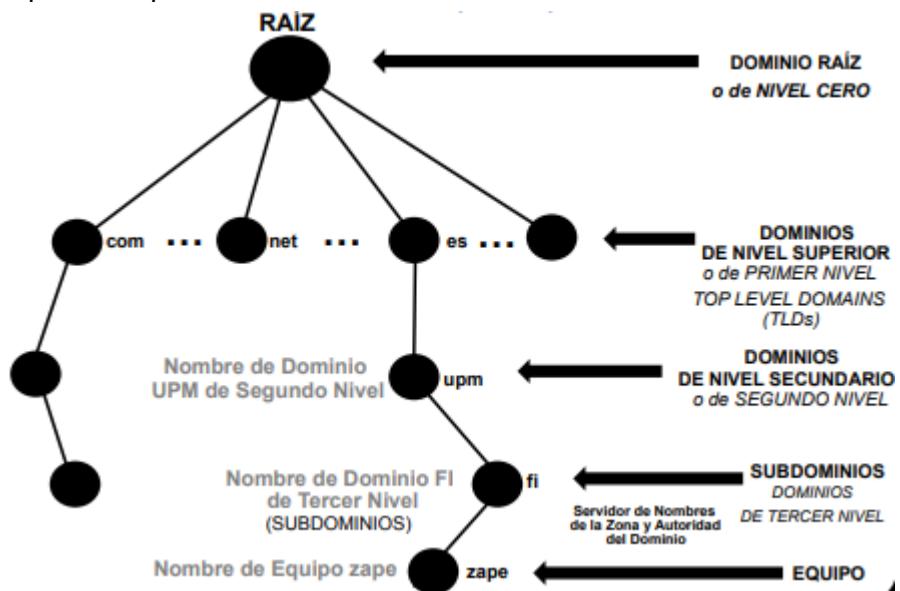


i. Componentes de sistema DNS:

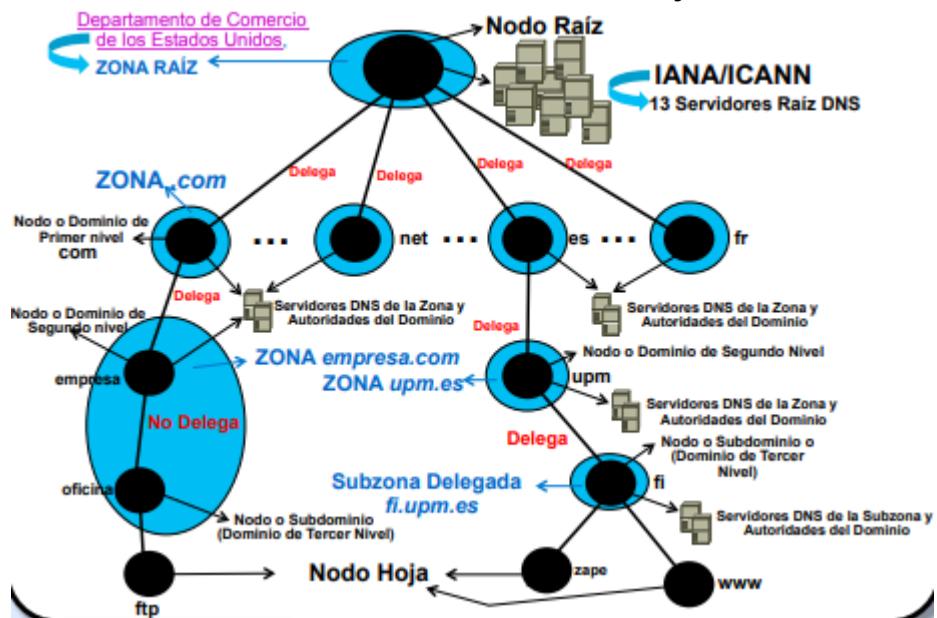
1. **Base de datos DNS en Internet:** BD **distribuida** mediante **servidores DNS locales** a las organizaciones conectadas a Internet y que gestionan sus **BDs locales DNS**:
 - a. **Registros locales** con las asociaciones entre nombres simbólicos y direcciones IP.
 - b. **Ningún servidor DNS** contiene la BD completa.
2. **Protocolo DNS:** se necesita un cliente DNS y un protocolo DNS para acceder a un servidor DNS para la resolución de nombres simbólicos en direcciones IP.

ii. (Componente 1 de sistema DNS) **Base de datos DNS en Internet:**

1. Estructura jerárquica de dominios DNS: la BD DNS en Internet se representa por un árbol.



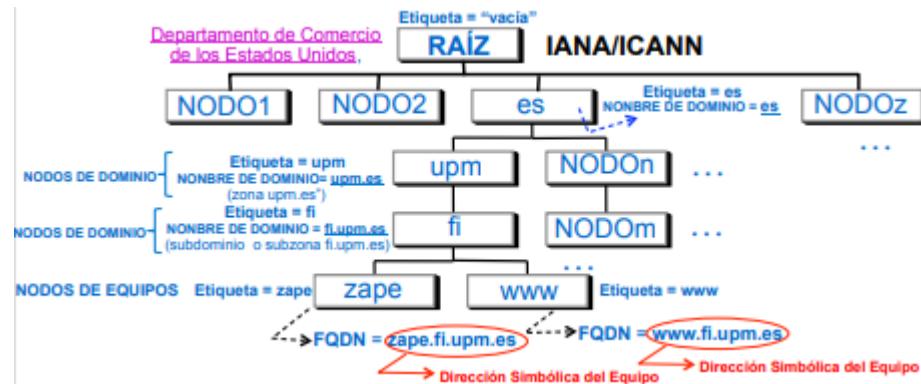
- Rango: [0, 127] niveles.
 - Ejemplo: fi.upm.es sería un nodo DNS a nivel 3.
2. Relación de zonas, subzonas, dominios, nodos y servidores DNS:



- Zona: agrupación de nodos.
- Subzonas: zona de un subdominio.
- En cada zona se gestionan servidores DNS con las asociaciones de direcciones inferiores en jerarquía.
- Los nodos pueden delegar en nodos inferiores las asociaciones. Si no delegan, deberán almacenar las asociaciones de los nodos inferiores dentro de su zona.
- Ejemplo: el nodo raíz almacena las asociaciones .com, .net, .es, .fr, ... y delega dominios inferiores a sus nodos hijo. Por ello, .upm.es guarda las asociaciones .xx.upm.es, .fi.upm.es las asociaciones .xx.fi.upm.es, ... Por otro lado, .empresa.com no delega en

oficina.empresacom y tendrá que almacenar las asociaciones DNS de la empresa y de sus oficinas.

3. Etiquetas y nombres de dominio:



- Etiqueta:** nombre del nodo en la jerarquía.
- Dirección simbólica (nombre de dominio):** secuencia de etiquetas separadas por puntos de la etiqueta del nodo a la raíz.
- FQDN (Fully Qualified Domain Name):** dirección simbólica de equipo.

4. TLD (Top Level Domain o dominios de nivel superior):

EL DOMINIO DE GESTIÓN DE PRIMER NIVEL (TLD), SE CORRESPONDE CON LAS ETIQUETAS (NOMBRES SIMBÓLICOS) DE PRIMER NIVEL distribuidas en

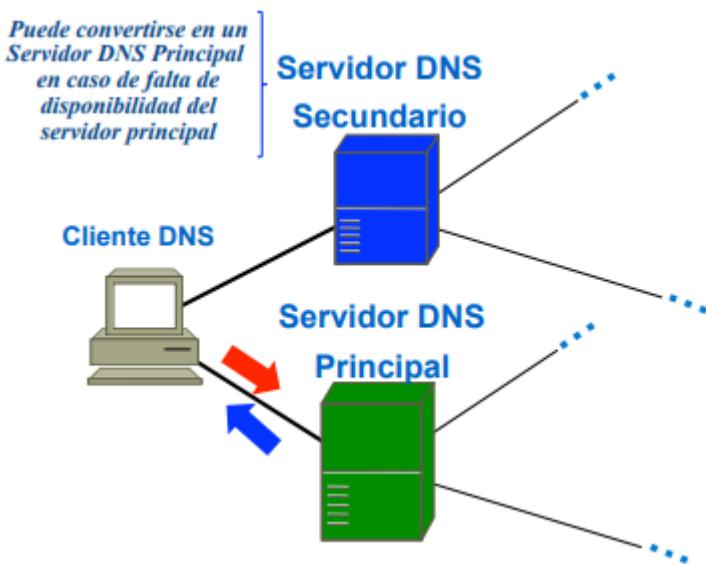
Etiquetas Genéricas (TLDs Genéricos) + Etiquetas de Países (TLDs países)
→ Nombres simbólicos que agrupan direcciones simbólicas por temáticas

TLD (Top Level Domain) = gTLDs (Generic TLDs) + ccTLDs (Country Code TLDs)

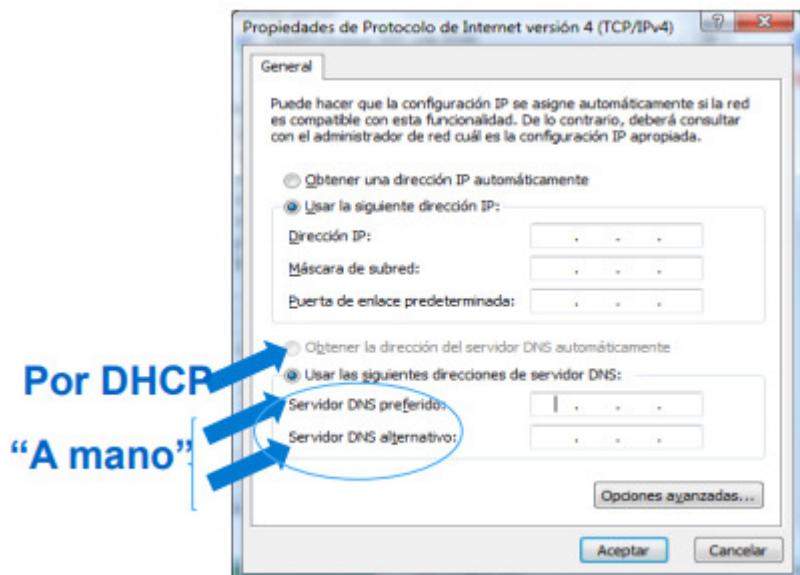


- Cada país tiene un **NIC (Network Information Center)**, agente registrador que controla que no haya etiquetas iguales en el nivel inmediatamente inferior (por ejemplo, dos upm.es distintos).
 - En España, tenemos el **ESNIC**.
 - Los dominios inferiores también pueden tener agente registrador con la misma función (por ejemplo, que no haya dos fi.upm.es).
 - Existen **etiquetas particulares TLD** aprobadas por el IANA por un coste superior (por ejemplo, .madrid, .empresa, .apellido, ...).
- iii. **(Componente 2 de sistema DNS) Protocolo DNS:** resolución de direcciones simbólicas (nombres de dominio) en direcciones IP.
- Transparente para el usuario:** al cliente DNS solo le puede llamar otro proceso cliente o utilidad de nivel de aplicación (por ejemplo, **ping**).
 - Sirve de **soporte** para otros protocolos o aplicaciones como HTTP.
 - Funcionamiento:**
 - Cliente DNS:** pregunta a **servidor DNS** por una asociación con una dirección simbólica. Si tiene su dirección IP, la devuelve.

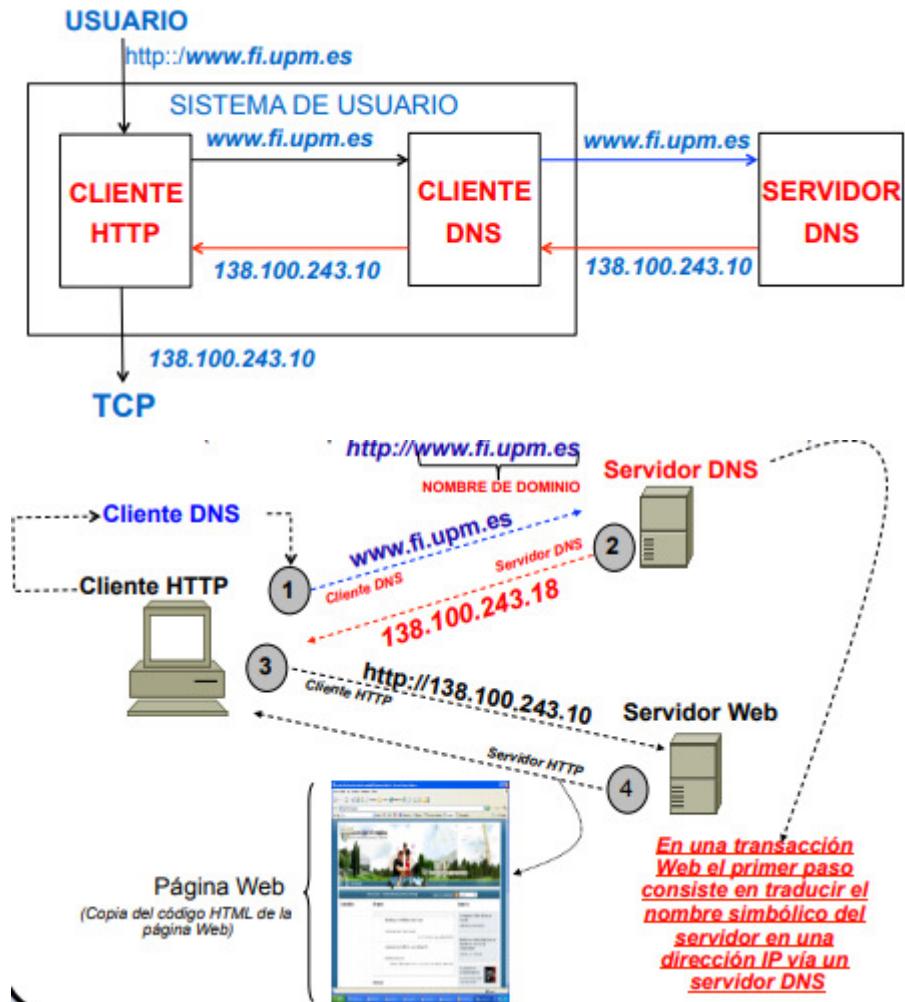
- b. **Servidor DNS:** si no tiene la dirección solicitada, pasa a ser cliente DNS de otro servidor DNS en la jerarquía DNS.
4. **Tipos de servidores DNS:**



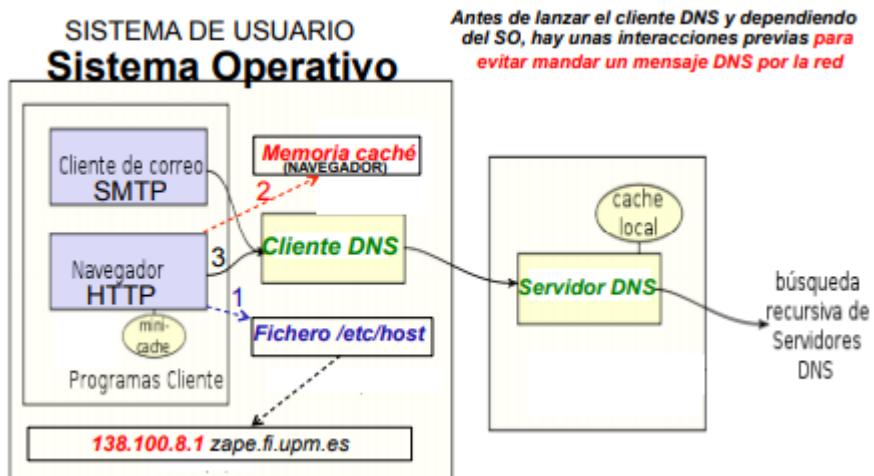
- a. Puede haber entre 0 e infinitos servidores DNS secundarios.
5. **Configuración de servidores DNS:** puede ser **a mano** o por **DHCP**.



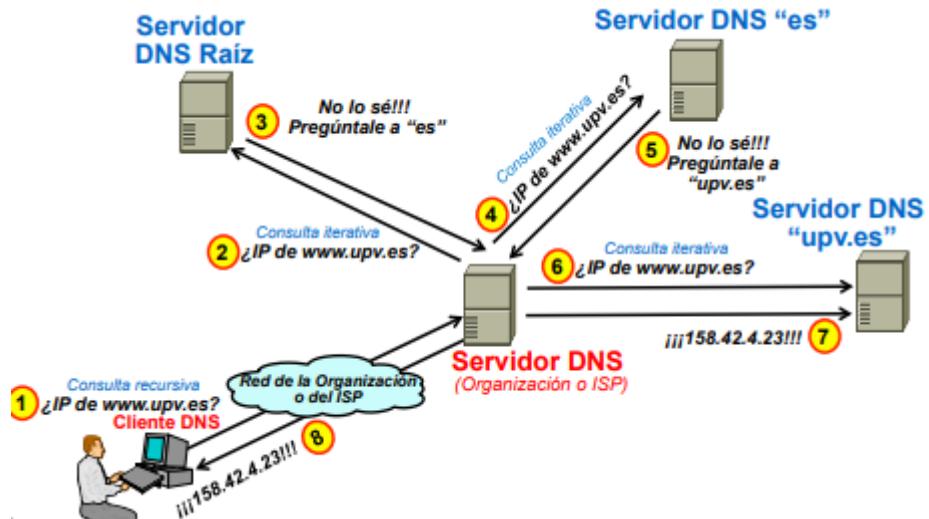
6. Ejemplo: servicio DNS solicitado previamente por cliente HTTP.



7. Para evitar un excesivo tráfico por la red, existen interacciones previas al servicio DNS (comprobación primero en el fichero /etc/host (local) y segundo en la memoria caché del navegador).



8. Consulta recursiva inicial e iterativa entre servidores DNS:



- Consulta recursiva:** llamada de cliente a servidor DNS local.
- Consulta iterativa:** llamada entre servidores DNS → se pregunta al servidor DNS raíz, que si no tiene la dirección IP requerida devuelve la dirección IP del dominio inferior correspondiente, así iterativamente hasta obtener la dirección IP de upv.es.
- El servidor DNS local guardará la dirección IP pedida en su caché para evitar realizar más consultas iterativas.

iv. Tabla DNS o BD del protocolo DNS:

1. Formato:

NOMBRE DE DOMINIO	TTL	CLASE	TIPO	DATOS
www.fi.upm.es	84600 (24h)	IN	A	138.100.243.18

- Nombre de dominio/equipo:** identificador del recurso (dominio) o la clave principal de búsqueda (equipo → FQDN).
- TTL y CLASE:**
 - TTL:** *Tiempo de Vida del Registro. Número de segundos que puede estar el registro en caché antes de ser descartado*
 - CLASE:** *Si el contenido es IN, identifica la clase del registro como clase Internet (o relacionada con los protocolos TCP/IP de Internet). Por ejemplo, para la clase IN existen tipos como: A, PTR, CNAME, MX, etc.* CH (para un sistema no relacionado con Internet)
- DATOS:** información específica del tipo de registro: dirección IP, FQDN o cadena ASCII.
- TIPO:** indica el tipo de recurso descrito por el registro DNS.
 - A:** relaciona nombre de dominio/equipo con dirección IPv4.
 - AAAA:** relaciona nombre de dominio/equipo con IPv6.
 - PTR (FQDN pointer):** resoluciones DNS inversas: devuelve dirección simbólica de una dirección IP (utilidad: **nslookup**).

NOMBRE DE DOMINIO	TIPO	DATOS
18.243.100.138.in-addr.arpa	PTR	www.fi.upm.es

- iv. **CNAME**: resuelve un alias (por ejemplo, fi.upm.es sin **www**)

NOMBRE DE DOMINIO	TIPO	DATOS
fi.upm.es (alias)	CNAME	www.fi.upm.es
www.fi.upm.es (FQDN)	A	138.100.243.10

- v. **MX (Mail eXchange)**: devuelve la dirección del **servidor de intercambio de correo** asociado al servidor SMTP remoto del usuario destinatario del correo.

1. Un servidor SMTP puede estar asociado a varios servidores de intercambio de correo. Se devolverá la primera dirección **activa** en orden de **prioridad**.

- vi. **TXT (plain text)**: información adicional mediante cadenas de texto (longitud máxima de 255 caracteres).

NOMBRE DE DOMINIO	TIPO	DATOS
smtp.destino.com	MX	mx.destino.com
mx.destino.com	A	220.2.0.2
mx.destino.com	TXT	"Servidor de Intercambio de Correo"

- d. **Protocolo HTTP (HyperText Transfer Protocol) (servicio web) [TCP]**: protocolo de presentación y distribución de información en Internet (servidor Web) y acceso a cualquier tipo de servicio (p. ej, correo Web) desde un cliente HTTP (navegador).

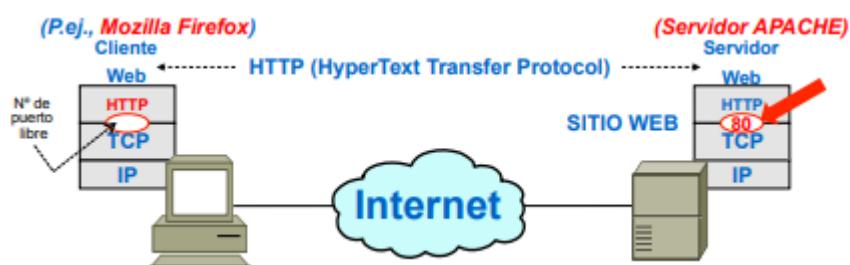
El creador del Servicio Web o Telaraña

SERVIDORES WEB ENLAZADOS vía HIPERENLACES o enlaces HTTP

Tim Berners-Lee

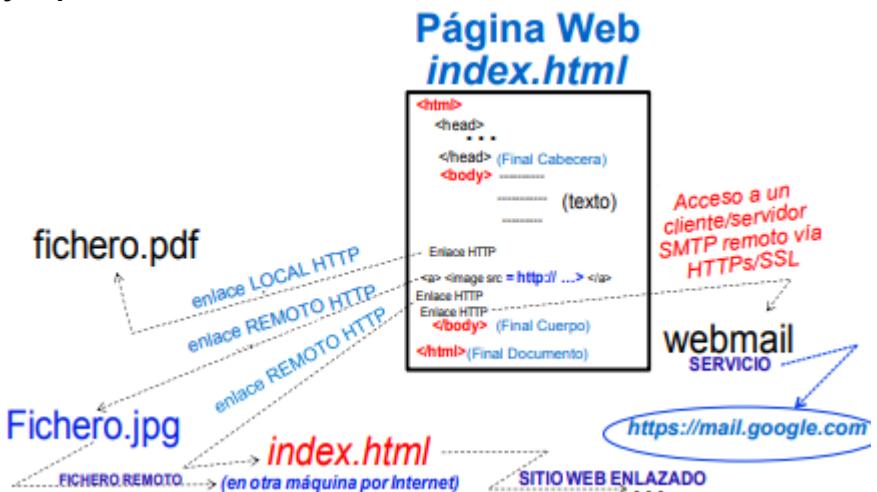


- Físico y científico británico en computación y comunicaciones
 - Diplomado en Física por la Universidad de Oxford
- Creador del protocolo **HTTP** (*HyperText Transfer Protocol*), del **Lenguaje HTML** (*HyperText Markup Language*) o lenguaje de etiquetas de hipertexto; y del formato de localización de ficheros en la web, el formato **URL** (*Uniform Resource Locator*)
- Desarrolló el protocolo HTTP y el Servicio Web en el CERN (*Laboratorio Europeo de Física de Partículas en la frontera franco suiza*) para que los investigadores de este laboratorio pudieran acceder a cualquier tipo de documentación científica.
 - La [primera comunicación entre un cliente y un servidor Web usando el protocolo HTTP](#) se llevó a cabo en [septiembre de 1991](#).



- i. **Cliente HTTP (navegador):** Mozilla Firefox, Safari, Google Chrome, ...
- ii. **Servidor HTTP (servidor web):** configuración específica de un software gratuito de libre distribución (por ejemplo, Apache o nginx).
 1. **APACHE (85,5% de servidores web):** implementación libre de un servidor web/HTTP de código abierto multiplataforma para cualquier kernel (Unix (OS X, GNU/Linux, BSD, etc), Microsoft Windows, ...)
 - a. **Nº puerto:** 80 (reservado para el administrador de la máquina). Si otros usuarios en la misma máquina quieren disponer de su propio servidor web, deberán utilizar otro nº puerto como el 8080.
 2. **nginx (14,5% de servidores web):** implementación libre de un servidor web/HTTP de código abierto multiplataforma para cualquier kernel.
 - a. Alternativa a **Apache**.
 - b. **Creador:** Igor Sysoev.
 - c. **Curiosidad:** la mayoría de periódicos utilizan este servidor web.
- iii. **Contenidos de un servidor web:**
 1. La descarga de contenidos de un servidor web por el cliente HTTP comienza por el **index.html** o **default.html** (fichero **HTML** en la página web inicial) para la interpretación y visualización por un intérprete **HTML** en el cliente HTTP (navegador).

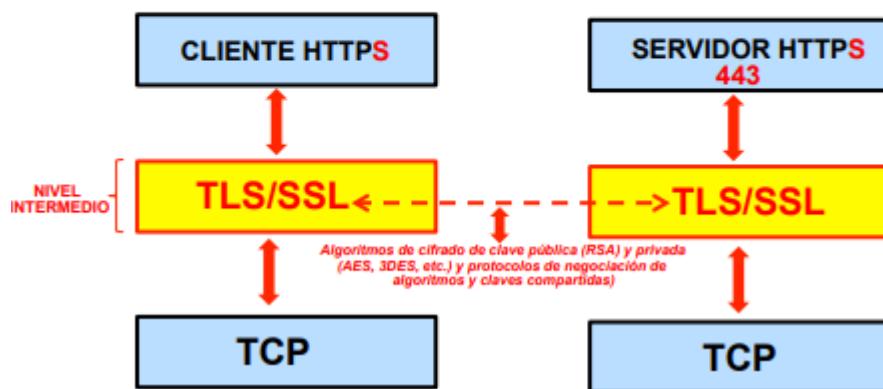
- a. Dispone de texto e hiperenlaces (enlaces HTTP) a ficheros locales o remotos de texto, audio, vídeo, imagen, ... y a cualquier tipo de servicio (por ejemplo, **Webmail**: acceso a cliente/servidor SMTP remoto vía HTTPS).
- 2. **HTML (HyperText Markup Language)**: lenguaje de etiquetas que define la estructura y contenido de la página web para facilitar su interpretación por un intérprete en el cliente HTTP (navegador).
 - a. **Ejemplo:** index.html



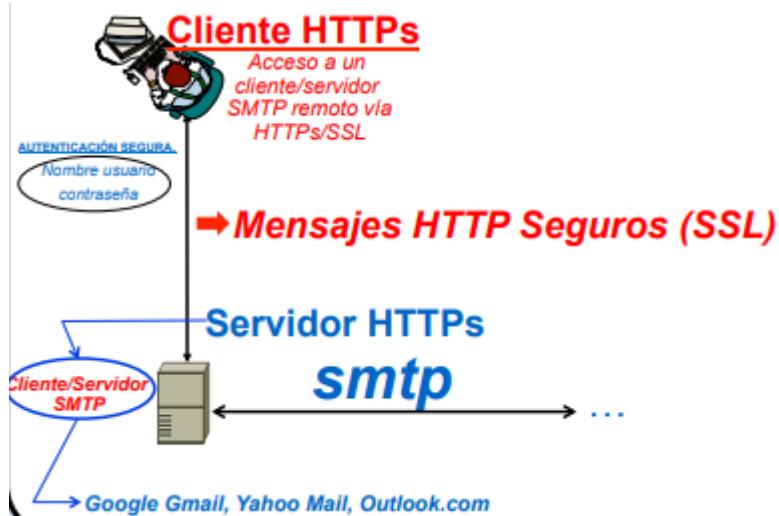
iv. Puertos Seguros de Servidores de Correo:

Para poder cifrar y descifrar los mensajes de correo (y autenticación previa) entre el cliente y el servidor SMTP, es necesario que tanto el cliente como el servidor SMTP dispongan de un Nivel Intermedio de Seguridad (TLS/SSL) entre TCP y el correspondiente proceso de aplicación SMTP.

Idem para los correspondientes clientes y servidores IMAP y POP3 para poder acceder al buzón y leer los correos de forma segura



- v. **WebMail:** acceso vía HTTPS a cliente/servidor SMTP en servidor web remoto



- vi. **URL (Uniform Resource Locator):** utilizado por el protocolo HTTP para localizar un fichero o recurso en un servidor HTTP/web en Internet.

- vii. **Formato URL: protocolo://equipo:puerto/ruta [4 parámetros]**

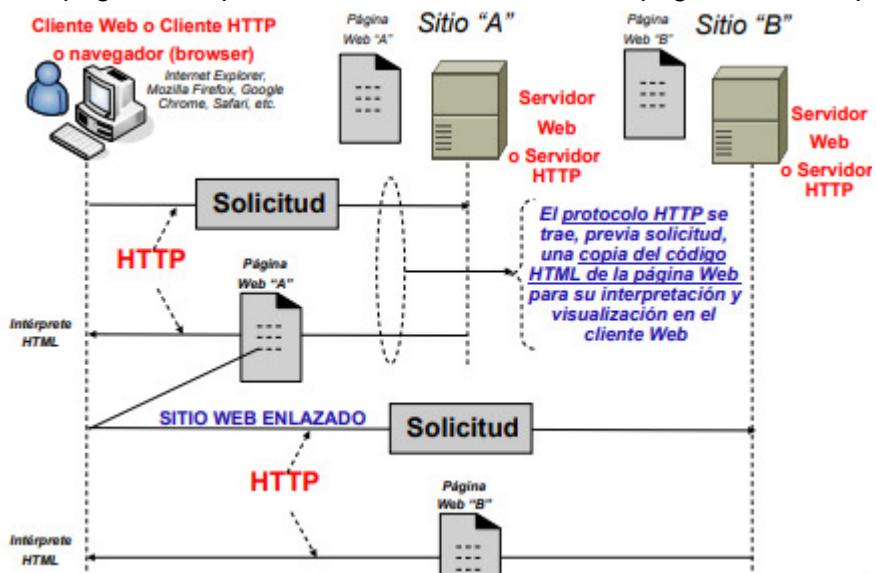
1. **Protocolo:** HTTP por omisión.
2. **Equipo:** alias o dirección simbólica o dirección IP del sitio Web.
3. **Puerto:** identificación de proceso servidor (si es **80**, no se indica).
4. **Ruta (path):** camino de directorios para acceder al fichero o recurso, separados por "/".
5. **Ejemplo:**

`http://pegaso.ls.fi.upm.es/comunicaciones/redes/normas.pdf`

protocolo equipo (puerto 80) ruta de directorios fichero pdf

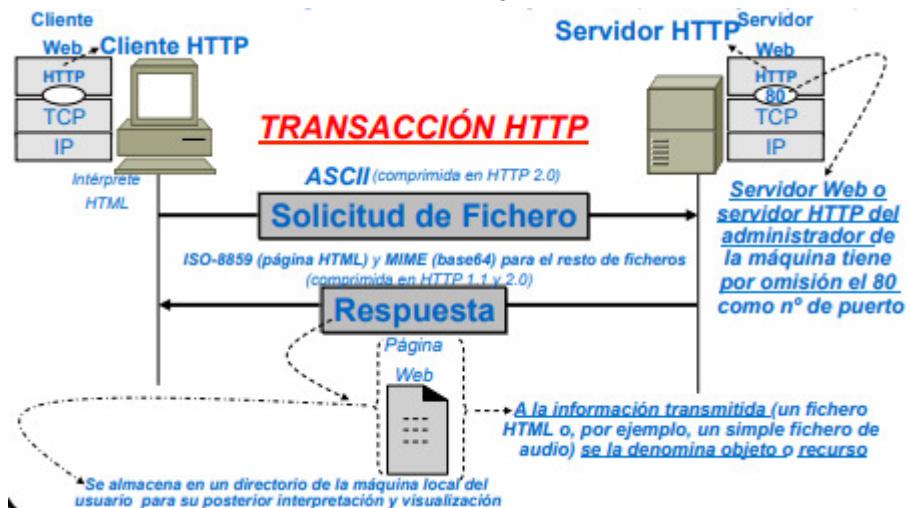
Ejemplo de Dirección HTTP en Formato URL

- viii. **World Wide Web (WWW), Web o Telaraña:** servicio distribuido por el que cada página web puede contener enlaces a otras páginas de cualquier sitio.



1. **Navegar por Internet:** enviar solicitudes a servidores HTTP/Web.

ix. Características protocolo HTTP 2.0 y 1.1:



1. **Persistente:** la conexión TCP no se libera independientemente del número de descargas de ficheros.
 - a. En **HTTP 1.0** solo se podía enviar un fichero por conexión TCP.
2. **Con pipelining:** el cliente puede enviar tantas solicitudes como quiera sin esperar a que solicitudes previas se resuelvan.
 - a. **Sin pipelining:** el cliente no puede enviar una nueva solicitud si no ha recibido el fichero de la solicitud previa.
3. **Sin estado:** cuando se libera una conexión TCP, todos los mensajes de solicitud/respuesta HTTP encapsulados en segmentos TCP se pierden, por lo que desaparecen todas las acciones realizadas en el navegador.
 - a. Para mantener el estado, el programador de la aplicación web tiene que gestionar dicho estado fuera o por encima de HTTP.
 - b. **Cookies:** fichero de texto que contiene las acciones del usuario para cada servidor web visitado y que se almacena en el disco duro del equipo local a través de su navegador y a petición del servidor HTTP. Cuando un cliente HTTP solicita la descarga de la página Web de un servidor HTTP, envía junto a la solicitud, todas las *cookies* que tenga con ese servidor. Esta información permitirá al servidor diferenciar usuarios y actuar distinto según cada uno.
4. **Usos frecuentes:**
 - i. **Seguimiento de usuarios y envío de promociones:** estadísticas de uso, cestas virtuales de compra, ...
 - ii. **Control de usuarios:** almacena nombre de usuario y contraseña en una *cookie* para evitar que el usuario tenga que introducir las credenciales en cada nueva conexión.
 - iii. **Personalización del sitio web:** según las preferencias del usuario, en cuanto a presentación y funcionalidad.
4. **Ejemplo:** persistente sin *pipelining* → establece conexión TCP + transferencia de datos fichero a fichero (uno a uno) + libera conexión

El usuario abre el navegador e introduce la dirección URL: www.periodico.es/deportes/index.html

- Al ejecutarse el cliente HTTP, éste obtiene un nº de puerto y se genera el socket local. El cliente HTTP hace una llamada a TCP solicitando un establecimiento de la conexión y, para ello, le pasa el socket remoto (TCP, dirección IP, 80)

contiene la página Web inicial o fichero HTML inicial con 10 referencias a imágenes jpg.

TCP es un servicio orientado a conexión: 3 FASES

FASE I (TCP):

ESTABLECIMIENTO DE LA CONEXIÓN TCP



- Después de SYN=0 ACK=1, la entidad TCP hace una llamada al cliente HTTP para que le pase UN BYTE STREAM (get index.html) EN

FASE DE TRANSFERENCIA DE DATOS

FASE II (TCP):

TRANSFERENCIA DE DATOS



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

Servidor HTTP recibe el mensaje de solicitud, y envía el MENSAJE DE RESPUESTA que contiene el fichero requerido

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

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

FASE II (TCP):

TRANSFERENCIA DE DATOS (continuación)

Un RTT por cada objeto referenciado



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

SIN PIPELINING: EL CLIENTE NO PUEDE ENVIAR 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



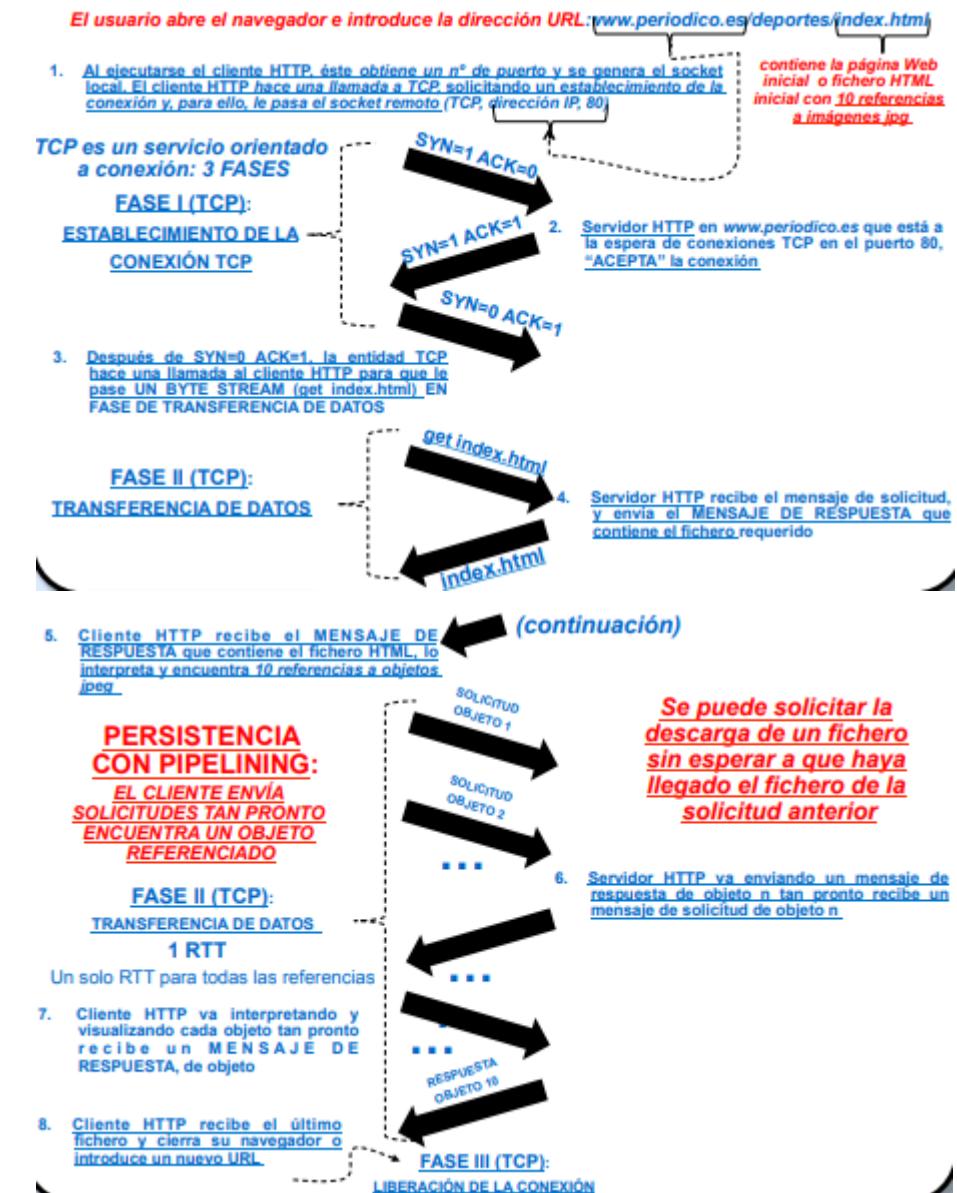
Servidor HTTP el mensaje de solicitud, forma el mensaje de respuesta que contiene el objeto requerido, envía dicho mensaje por su socket

EL SERVIDOR DEJA LAS CONEXIONES ABIERTAS DESPUÉS DE ENVIAR LA RESPUESTA

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

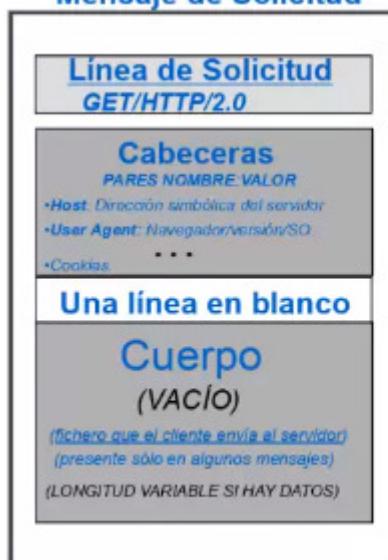
FASE III (TCP):
LIBERACIÓN DE LA CONEXIÓN

- Ejemplo: persistente con pipelining → establece conexión TCP + transferencia de datos (solicitudes/resuestas de fichero sin esperar) + libera conexión.

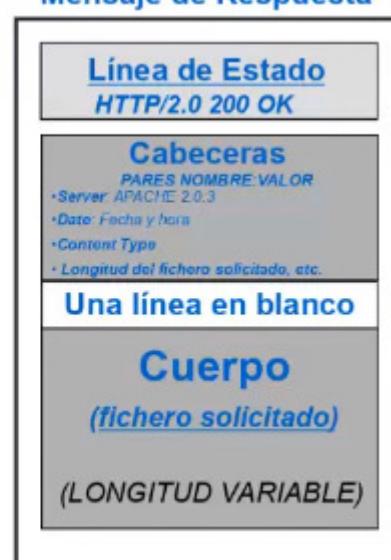


x. Formato mensajes HTTP (solicitud/respuesta):

Mensaje de Solicitud



Mensaje de Respuesta



- 1. Solicitud:** petición de descarga de **index.html** (página web inicial).
- xii. Mensaje de respuesta del servidor:** código de estado + frase de estado

- **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 otra 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 precondition**
- **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**

- xiii. Diálogo HTTP:** solicitud de descarga de **index.html** incluye cómo hacerla (formato (.html o .xml, para ficheros de gran tamaño), idioma, compresión, ...) y respuesta del servidor es el **index.html** de tipo html y charset = iso-8859 (ASCII de 8 bits) comprimido con gzip con línea de estado 200 (OK).

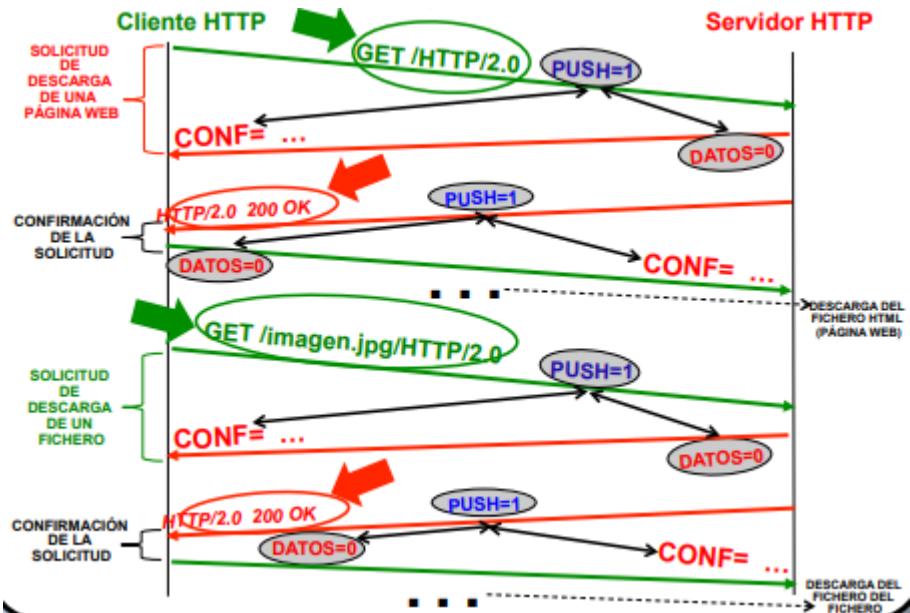


Esta transparencia es un puto caos

- xiii. Métodos HTTP:** hay 8 implementados en el código del cliente HTTP. Indican las acciones sobre el correspondiente recurso.

1. **GET:** solicita un recurso (página web, fichero, información, ...). Generalmente, se ejecuta cuando el usuario hace clic en un enlace.
 - a. **Obtención de index.html:** GET /HTTP/2.0 (protocolo/versión)
 - b. **Obtención de fichero:** GET /images/logo.png/HTTP/2.0
 - c. A veces, GET incluye parámetro visibles que se encapsulan en la barra de direcciones en una búsqueda de información (Google). `/index.php?page=main&lang=es`
2. **POST:** envía parámetros o datos del usuarios no visibles en la URL. El código del formulario exige que los datos metidos se envíen vía post.

3. PUT: envía un fichero al servidor.
- xiv. **Bit PUSH de cabecera TCP utilizado por HTTP:** los GETs (solicitudes) y las líneas de estado HTTP/2.0 200 OK (respuestas) llevan siempre el bit PUSH activado, lo que exige una **confirmación inmediata sin datos**.



Descarga de index.html: la respuesta OK a la solicitud incluye el primer fragmento del fichero pedido (PUSH = 1). El resto de fragmentos no tendrán activado el bit PUSH (= 0), salvo el último.

