

# PROTOCOLOS DE TRANSPORTE

---

## Módulo 8

# Temas a tratar

1. Suite de Protocolos de Interred
2. Protocolos de transporte: multiplexado y demultiplexado de conexiones TCP y UDP
3. Protocolo TCP
  1. Formato de la Trama
  2. Establecimiento de la conexión
  3. Transferencia de datos
  4. Cierre de la conexión
  5. Temporizadores
  6. Control de Congestión
4. Protocolo UDP

# Objetivos del módulo

Al finalizar el presente módulo el alumno debe ser capaz de:

- ✓ Comprender cual es la problemática que los protocolos de transporte deben resolver
- ✓ Conocer cuales son los componentes de la implementación del protocolo TCP
- ✓ Entender cuales son los procedimientos que se implementan para el control de la congestión en las redes
- ✓ Conocer el alcance de la funcionalidad del protocolo UDP

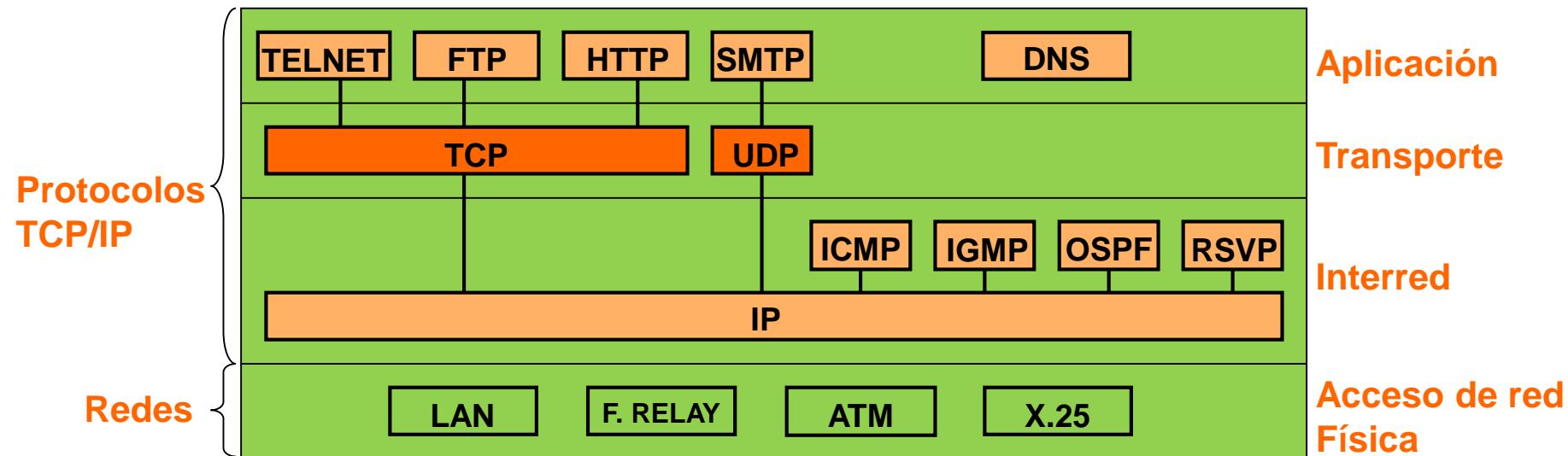
# Protocolos de Transporte

## Suite de protocolos de interred

Ubicación: el protocolo de transporte opera **encima de la capa de interred** y **debajo de la capa de aplicación**

Función: proporcionar el **Servicio de Transporte** a los datos que generan las aplicaciones (FTP, SMTP, TELNET, HTTP, ...)

TCP (Transmission Control Protocol) y UDP (User Datagram Protocol)



# Protocolos de Transporte

## Transmission Control Protocol (TCP)

- ✓ RFC 793 (1981)
- ✓ Orientado a conexión
  - ✓ Antes del intercambio de datos se establece una conexión extremo a extremo
- ✓ Confiable
  - ✓ Implementa Secuenciamiento de paquetes y confirmaciones (ACK's)
  - ✓ Control de Flujo
- ✓ Byte Stream
  - ✓ Está orientado bytes, no a tramas
- ✓ Ventana
- ✓ Control de Congestión
- ✓ Full Duplex

# Protocolos de Transporte

## User Datagram Protocol (UDP)

- ✓ User Datagram Protocol (RFC 768)
- ✓ Best Effort:
  - ✓ Los segmentos pueden perderse
  - ✓ Los segmentos pueden llegar desordenados
- ✓ Sin conexión
  - ✓ No hay “handshake”
  - ✓ Los segmentos son independientes
- ✓ A cada mensaje de la aplicación le corresponde un segmento

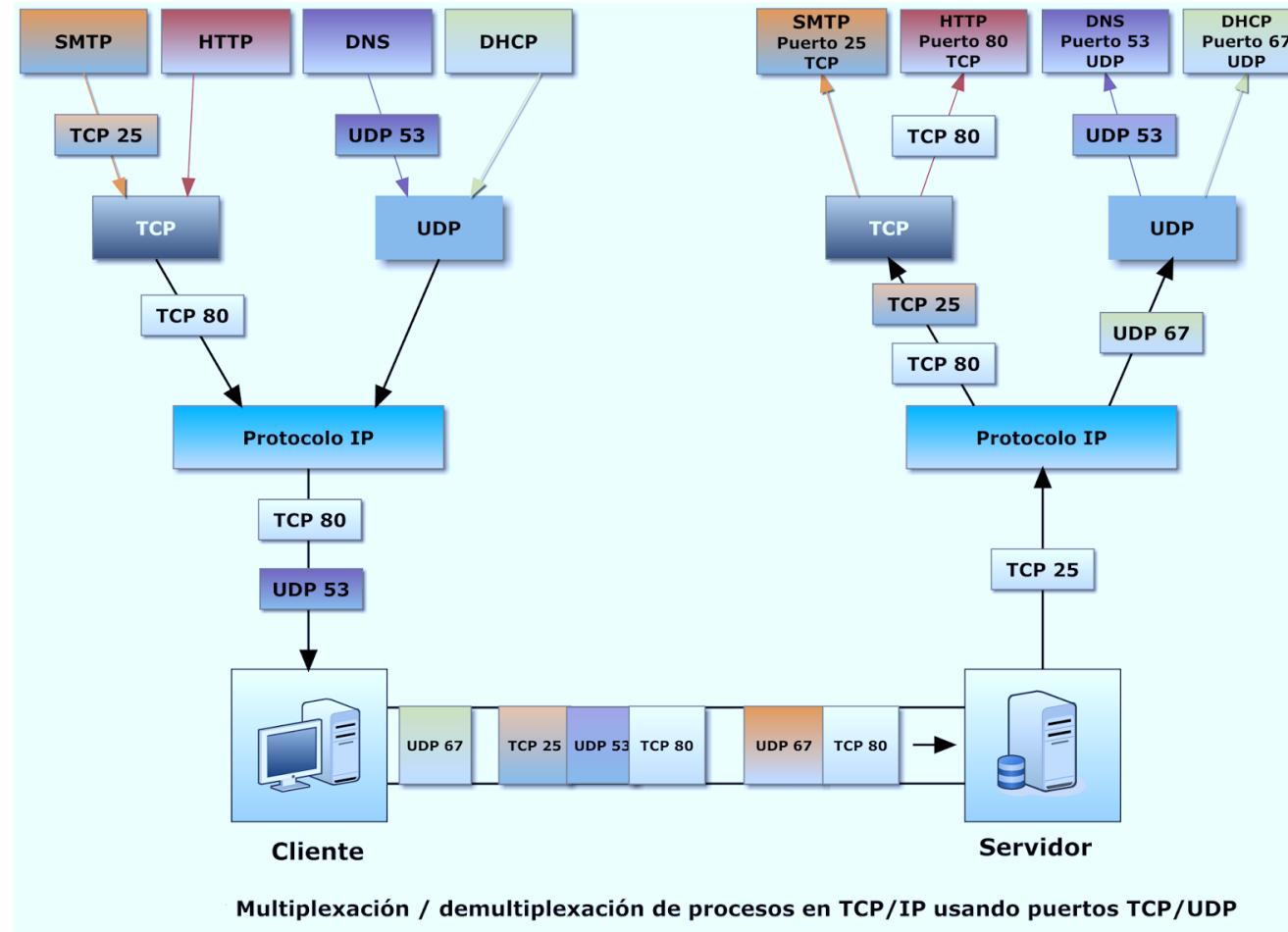
# Protocolos de Transporte

## Multiplexado y demultiplexado de conexiones

- ✓ Multiplexación
  - ✓ Colectar datos de múltiples procesos
- ✓ Demultiplexación
  - ✓ Entrega los segmentos a los procesos
- ✓ Identificación del extremo (host y aplicación):
  - ✓ Imposibilidad de extender las direcciones IP
  - ✓ No quedan bits disponibles
  - ✓ No se puede utilizar parámetros dependientes del SO
    - ✓ Process ID
    - ✓ Task number
    - ✓ Job name
- ✓ Debe funcionar en todos los sistemas

# Protocolos de Transporte

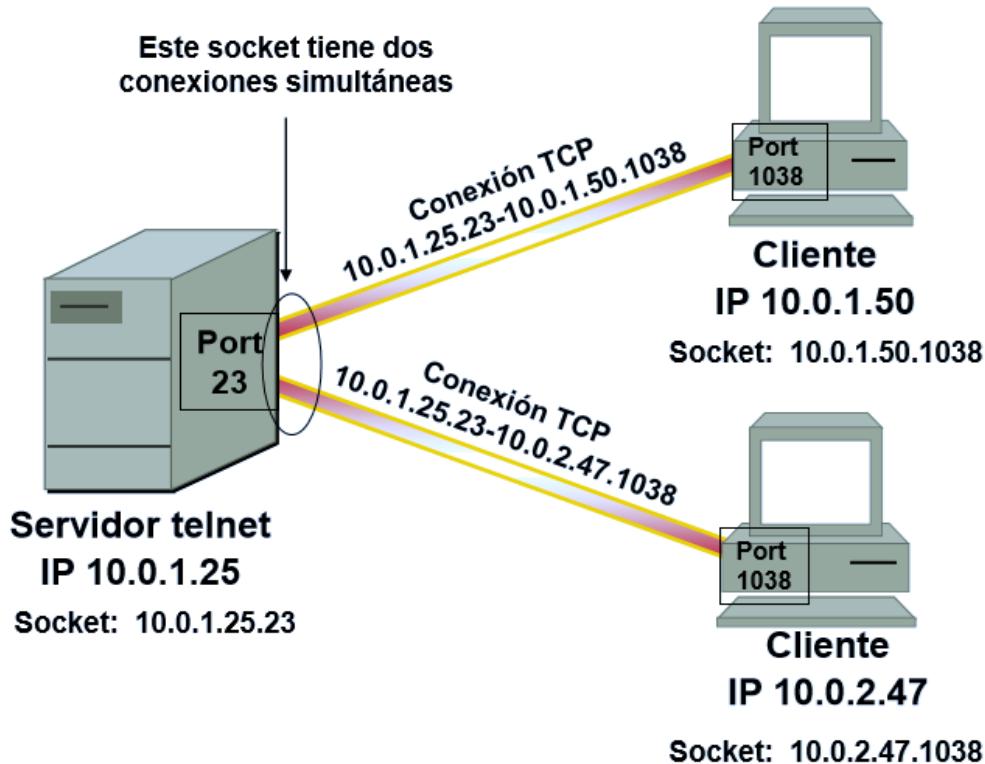
## Multiplexado y demultiplexado de conexiones



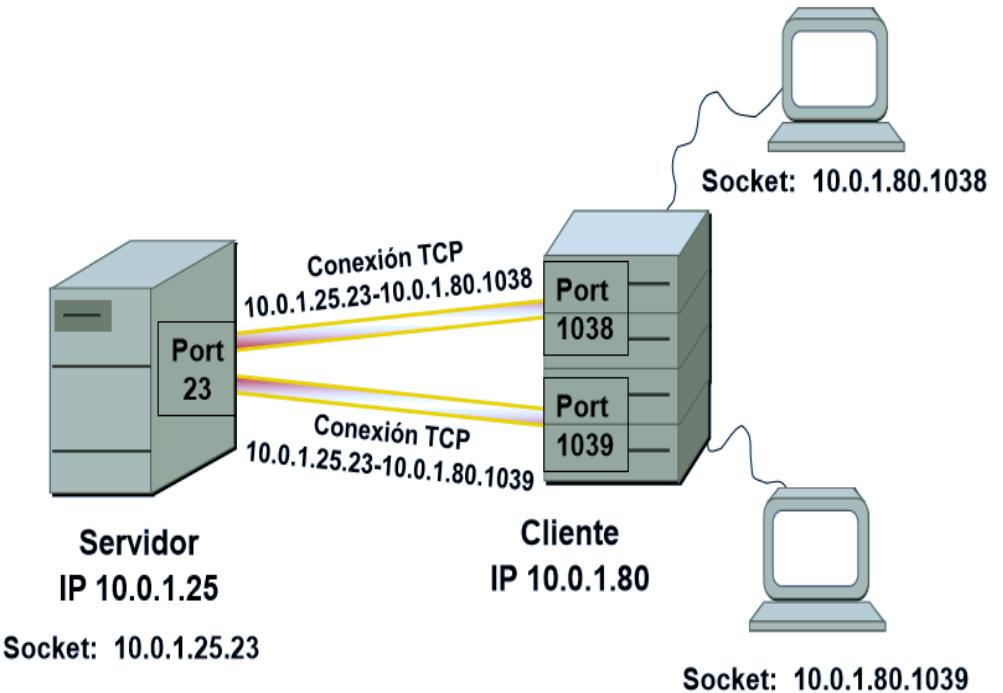
# Protocolos de Transporte

## Multiplexado y demultiplexado de conexiones

2 conexiones TCP al mismo puerto desde 2 clientes distintos



2 Conexiones a un mismo socket desde una única dirección IP



# Protocolos de Transporte

## Multiplexado y demultiplexado de conexiones: direccionamiento

Para que el módulo de transporte pueda entregar en destino los datos del segmento, es necesario que conozca:

1. Dirección IP del host destino (lo resuelve IP con el valor de en la cabecera del campo “Destination IP Address”).
2. El servicio que usará el protocolo de transporte: está definido en la cabecera de IP, en el campo “Protocol Number”:
  - TCP = 6
  - UDP = 17
3. La aplicación a la que tiene que entregar los datos. Los hace a través del valor del campo “Destination Port” (Puerto) en la cabecera del segmento TCP.

Al valor que resulta de concatenar la Dirección IP y el Puerto, se lo conoce con el nombre de socket: 192.168.32.5:110

# Protocolos de Transporte

## Multiplexado y demultiplexado de conexiones: puertos

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

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

# Transmission Control Protocol (TCP) – RFC 793 (1981)

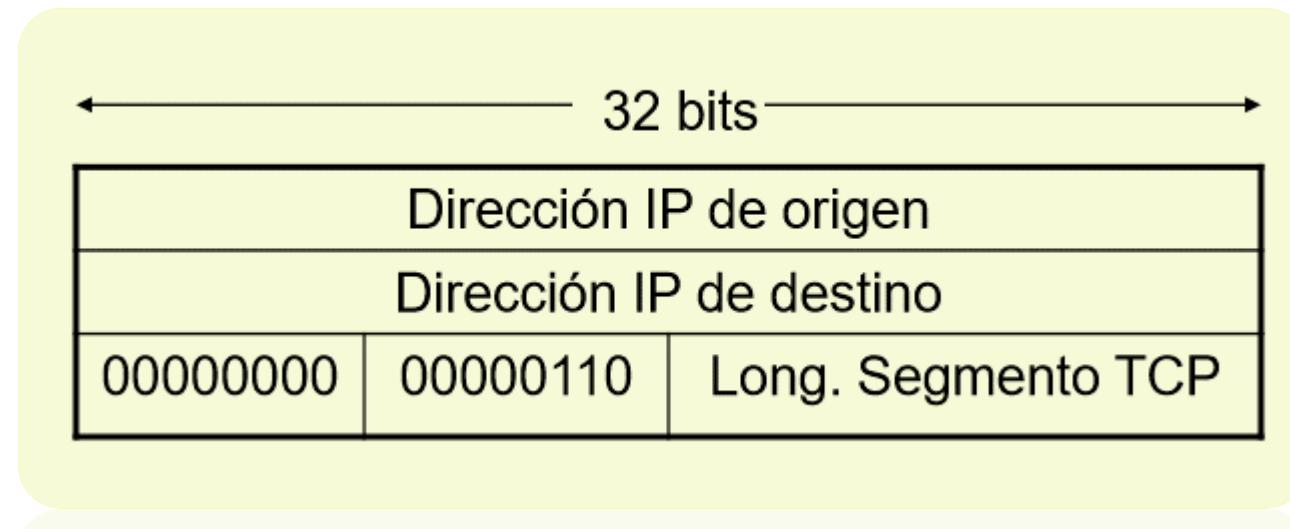
## Formato de la trama

Offsets Octeto		0								1								2								3							
Octeto	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Puerto de origen																Puerto de destino															
4	32	Número de secuencia																															
8	64	Número de acuse de recibo (si ACK es establecido)																															
12	96	Longitud de Cabecera	Reservado	N S R	C W E	E C G	U R K	A C H	P S T	R S I	S Y N	F I N	Tamaño de Ventana																				
16	128	Suma de verificación																Puntero urgente (si URG es establecido)															
20	160	Opciones (Si la Longitud de Cabecera > 5, relleno al final con "0" bytes si es necesario)																...															
...	...																	...															

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Formato de la trama

- ✓ Se añade al principio del segmento solo para el cálculo del Checksum, no se envía. Permite a TCP comprobar que IP no se ha equivocado (ni le ha engañado) en la entrega del segmento.
- ✓ El valor 6 = 0x06 (Hex) indica que el protocolo de transporte es TCP
- ✓ El valor 17 = 0x11 (Hex) indica que el protocolo de transporte es UDP



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Transmisión de datos usando TCP: establecimiento de la conexión

- ✓ TCP es un protocolo orientado a Conexión
- ✓ Pasos para transmitir datos: Establecer conexión, Transferir datos, Cerrar conexión
- ✓ Objetivos del establecimiento de la conexión
  - ✓ Permitir que los extremos comprueben su existencia
  - ✓ Permitir el establecimiento de parámetros iniciales (Número de Secuencia, Tamaño de ventana (Windows), entre otros) y de parámetros opcionales(tamaño máximo del segmento, factor de escala de ventana, etc.)
  - ✓ Inicializar la asignación de recursos de la entidad de transporte (buffers, entradas en tabla de bloques de segmentos, etc.)

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: parámetros

### Obligatorios

- ✓ Número de Secuencia en ambos sentidos (SN: Secuencial Number, AN: Acknowledgement Number)
- ✓ MSS (“Maximum Segment Size”) (536 Bytes Default)
  - ✓ Corresponde solo al PAYLOAD TCP
- ✓ Tamaño de la Ventana en ambos sentidos (W: Windows)

### Opcionales

- ✓ Factor de Escalado de Ventana
- ✓ SACK (Selective Acknowledgement) (No se desarrolla en la asignatura)
- ✓ Método de cómputo de Checksum alternativo
- ✓ Time Stamp (sellos de tiempo)

# Transmission Control Protocol (TCP) – RFC 793 (1981)

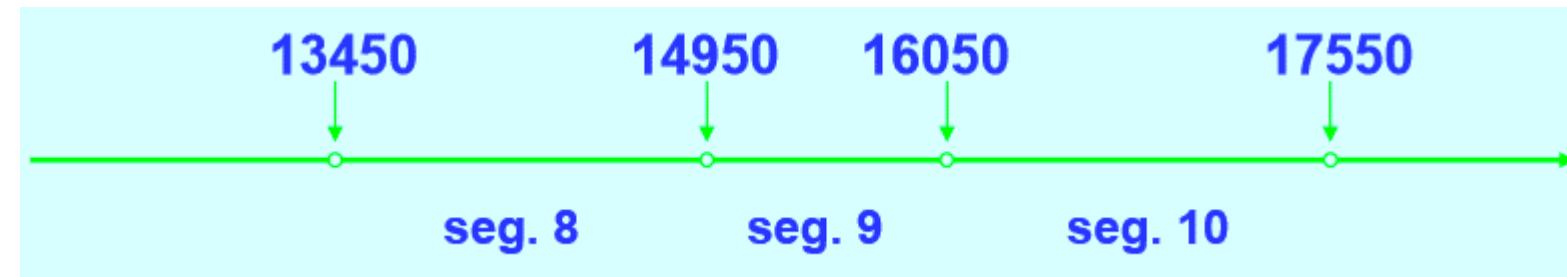
## Establecimiento de la conexión: parámetros opcionales

End of Options	kind=0	1 byte	El campo "len" indica la longitud en bytes de la opción: kind + len + data
NOP (no operation)	kind=1	1 byte	
Maximum Segment Size	kind=2	len=4	$1+1+2 = 4$
Window Scale Factor	kind=3	len=3	shift count
Timestamp	kind=8	len=10	timestamp value      timestamp echo reply
	1 byte	1 byte	4 bytes      4 bytes

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: número de secuencia

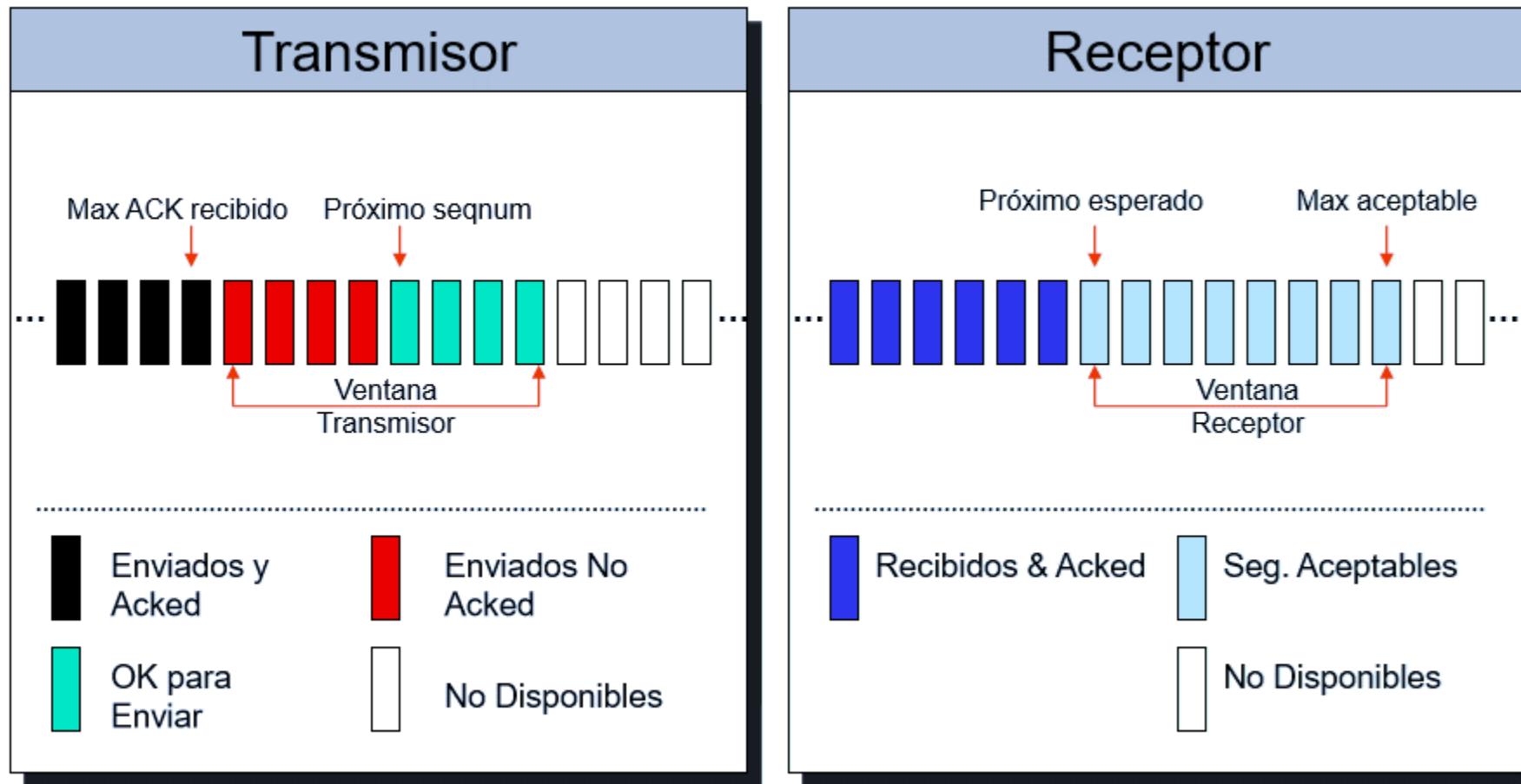
- ✓ Número de 32 bits.
- ✓ Valor inicial fijado en el establecimiento de la conexión, (ISN).
- ✓ TCP desagrega el Stream de bytes en segmentos.
- ✓ Cada segmento tiene asociado el número de secuencia.
- ✓ Indica su ubicación en el Stream de bytes.



En este caso el segmento 8 comienza en el byte 13450 y termina en el byte 14949

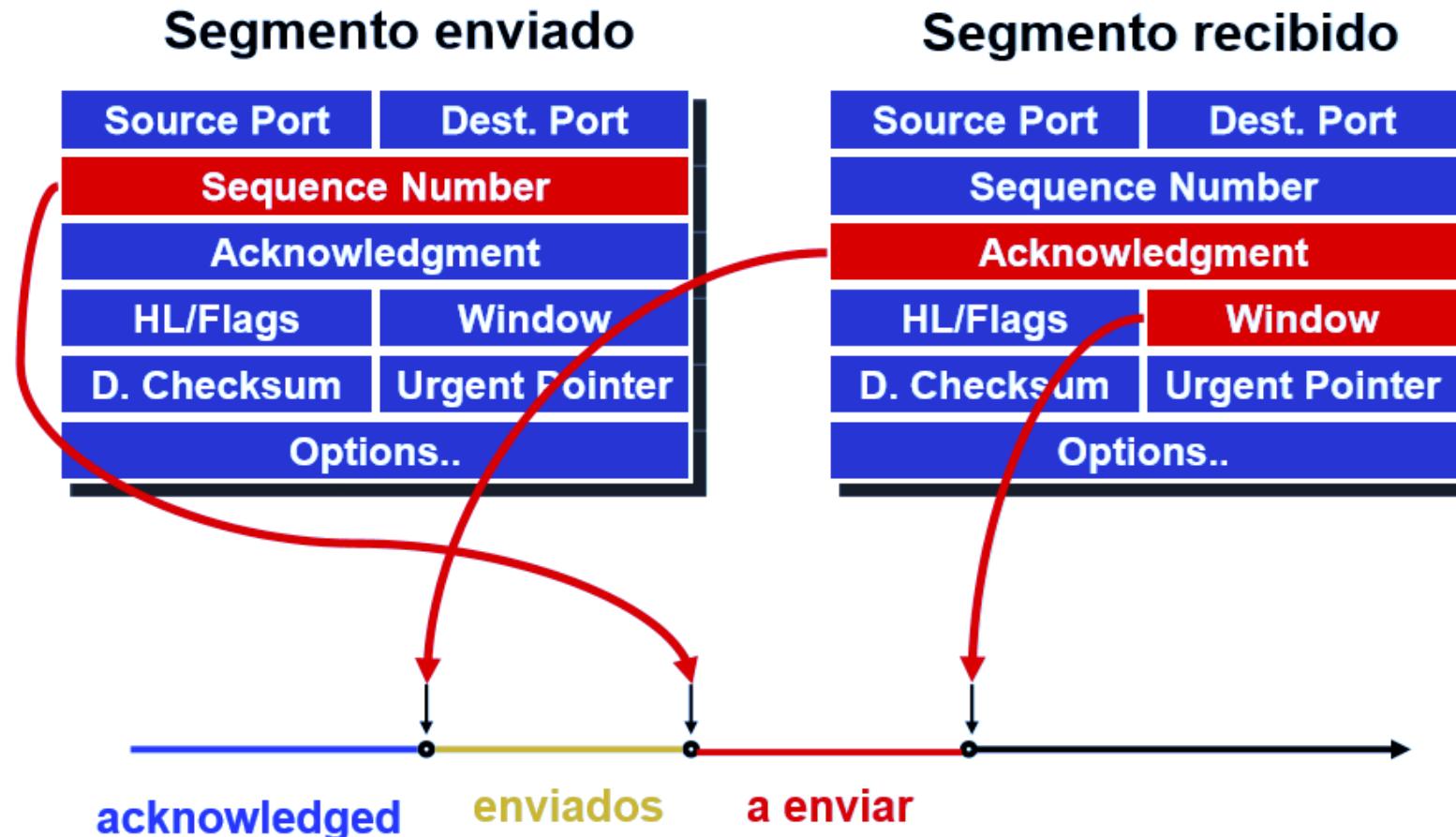
# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Tamaño de la Ventana: Ventana deslizante, concepto



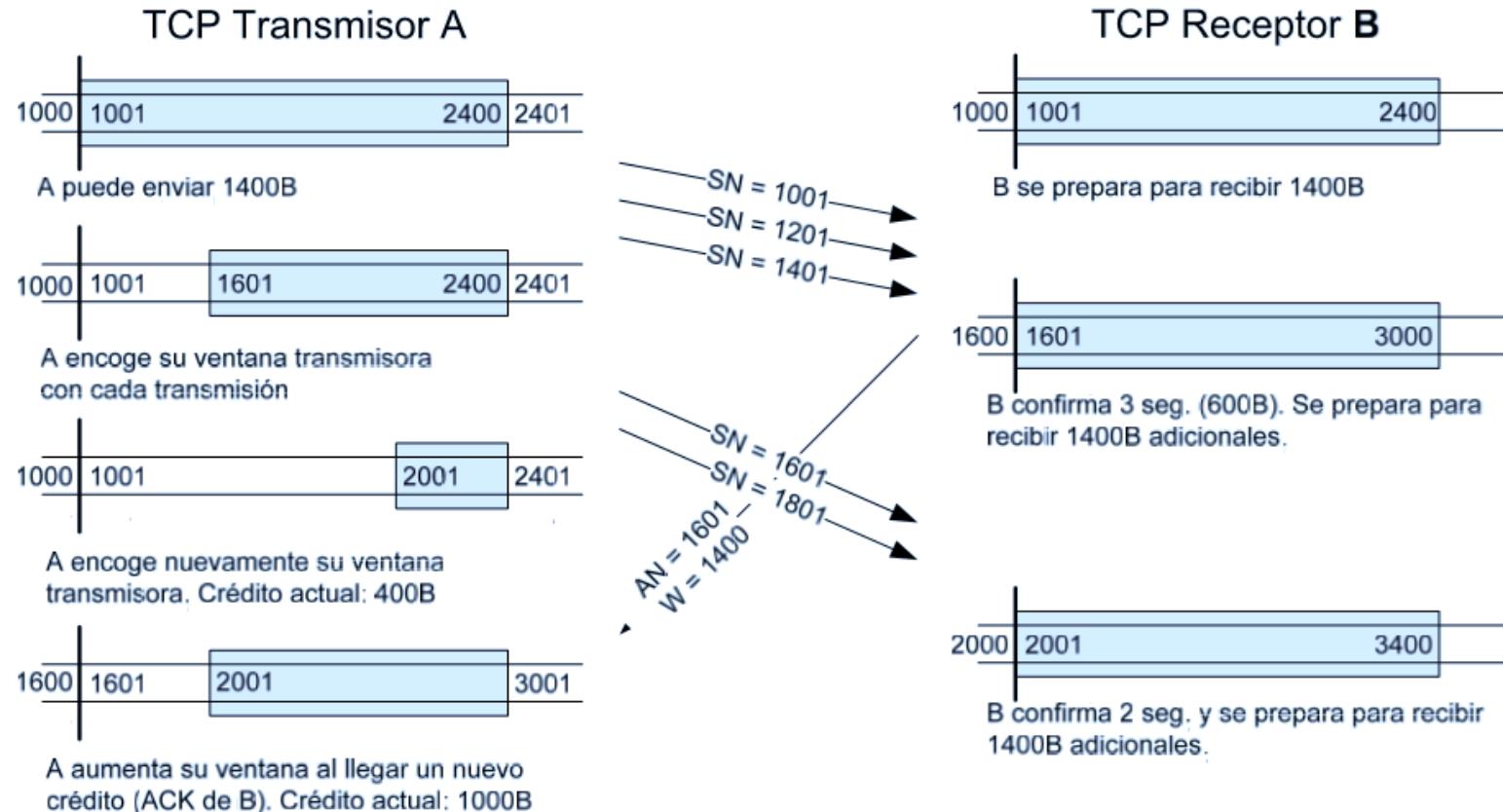
# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Tamaño de la Ventana: Ventana deslizante, variables



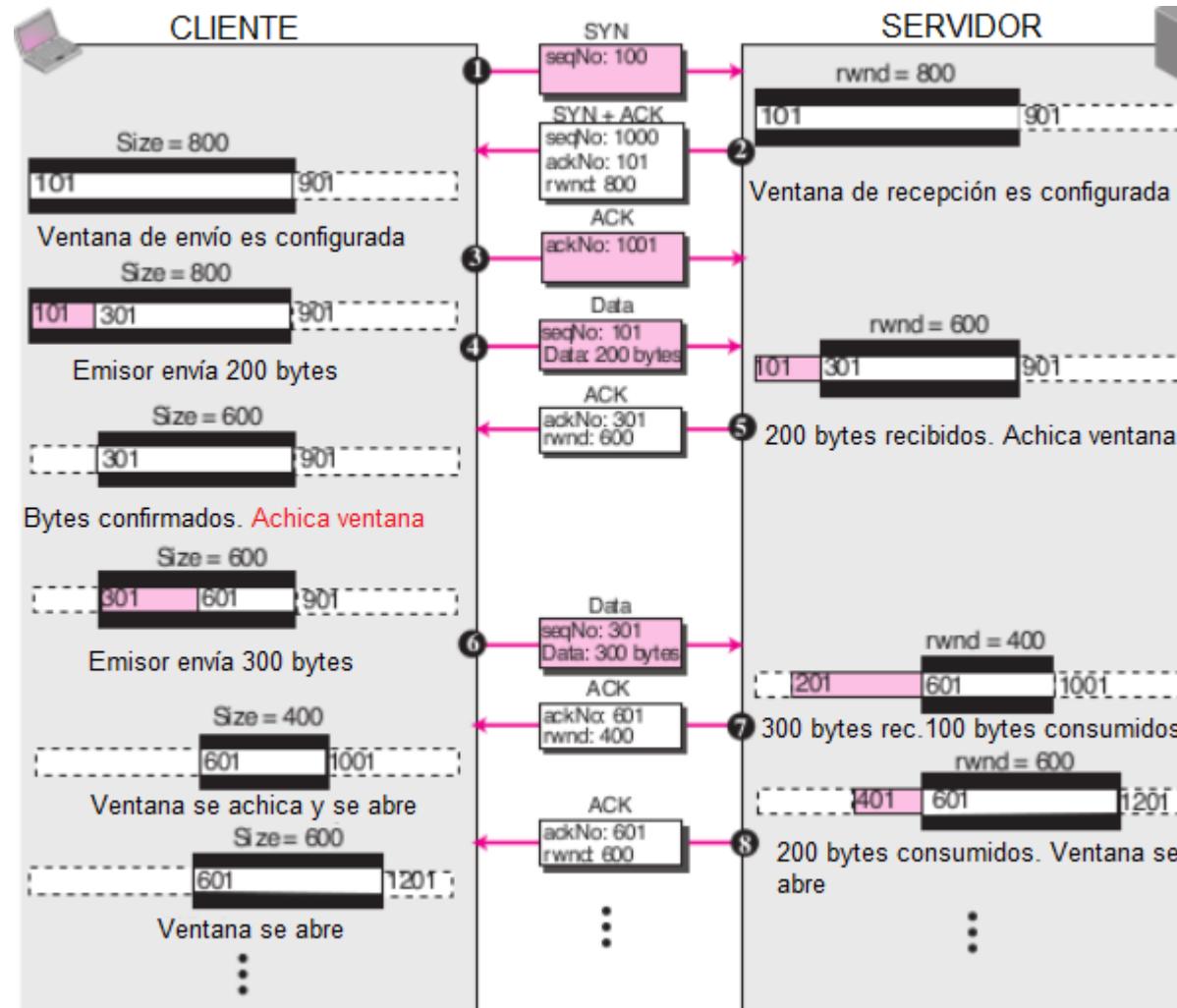
# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Tamaño de la Ventana: Ventana deslizante: un ejemplo



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Tamaño de la Ventana: Ventana deslizante, control de flujo



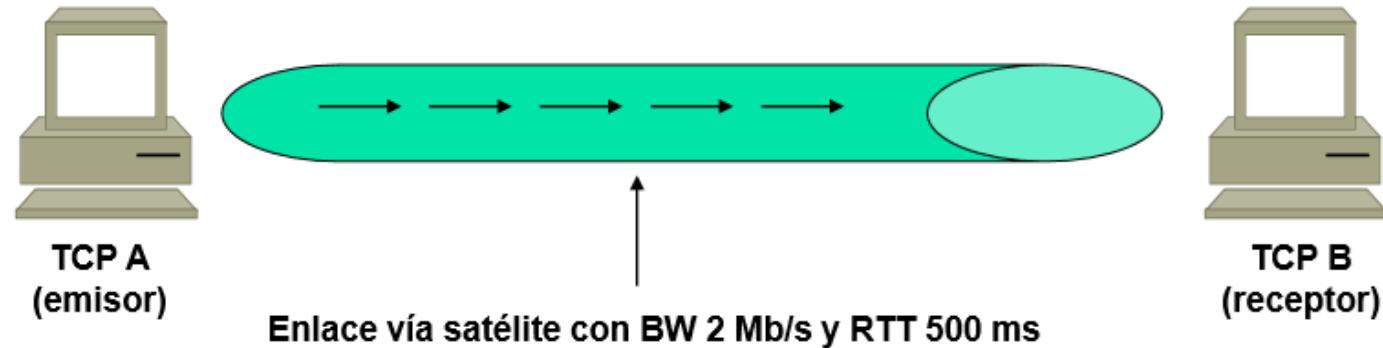
# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: factor de escalado de ventana

- ✓ La ventana de TCP es un campo de 16 bits. Su valor máximo es 65535 bytes. En TCP no es posible enviar más de 65535 bytes seguidos sin haber recibido un ACK
- ✓ Las redes LFN (Long, Fat pipe Networks/Elephant Networks) son las que tienen un gran ancho de banda, pero un elevado RTT (retardo). Son un problema
- ✓ Ejemplo, un enlace vía satélite de 2 Mb/s y retardo 500 ms:  $BW * RTT = 1 \text{ Mb}$  (se desperdicia el 50 % del enlace)
- ✓ En una red LFN cuando  $BW * RTT > 64 \text{ Kbytes}$  el rendimiento se puede ver limitado por este motivo. La limitación es tanto mayor cuanto mayor es el  $BW * RTT$  de la red
- ✓ Cuando se diseñó TCP nunca se imaginó llegar a los anchos de banda de la actualidad

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: factor de escalado de ventana



- 0 ms: TCP A empieza a enviar datos a 2 Mb/s
- 262 ms: TCP A ha enviado 64 KB y tiene que parar
- 500 ms: TCP A empieza a recibir los ACK y transmite los siguientes 64 KB
- 762 ms: TCP A ha enviado el segundo grupo de 64 KB y tiene que parar
- 1000 ms: TCP A empieza a recibir los ACK del segundo grupo y transmite
- 1262 ms: TCP A tiene que parar
- ...

$$\text{Eficiencia: } 262/500 = 52,4 \% = 1,048 \text{ Mb/s (64 KB/ RTT)}$$

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: factor de escalado de ventana

- ✓ Solución al problema planteado:
  - ✓ Tener ventanas mayores que 64 KB. Pero el campo es de 16 bytes y no se puede ampliar
- ✓ Pero se puede:
  - ✓ Aplicar un factor de escala al tamaño de ventana.
  - ✓ Será soportado por los dos TCP que establecen la conexión.
  - ✓ Lo acordaran al principio de esta y lo mantienen durante toda la conexión
  - ✓ Se define en el campo “options” de la cabecera TCP (Windows Scale Options)

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: factor de escalado de ventana

- ✓ Caudal máximo = Tamaño de ventana / RTT
- ✓ Con RTT = 43 ms y ventana 524280 bits:
- ✓ Caudal máximo =  $524280 / 0,043 = 12,2 \text{ Mb/s}$

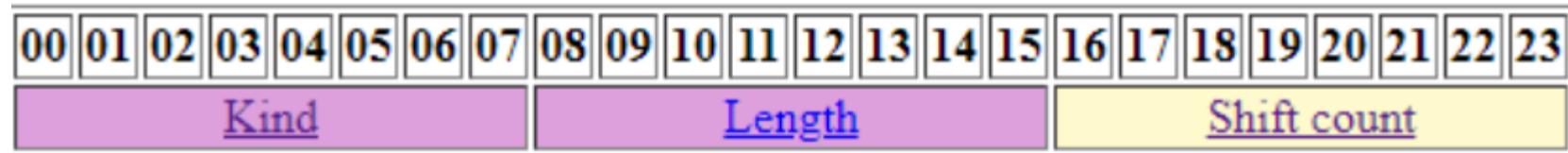
Factor de escala	Tam. Ventana (bits)	Caudal max. (Mb/s)
1	524280	12,2
2	1048560	24,4
4	2097120	48,8
8	4194240	97,6
16	8388480	195,2

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: factor de escalado de ventana



### TCP Option 3:



**Kind.** 8 bits. Set to 3.

**Length.** 8 bits. Set to 3.

**Shift count.** 8 bits.

This value is used as a multiplier to the window value.

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: fases

Un módulo TCP origen decide iniciar una conexión con un módulo TCP destino y aprovecha para enviar D bytes de datos a partir del número de secuencia X

Cuando TCP inicia una conexión para transmitir un bloque de datos, no necesariamente el número de secuencia inicial es 0.

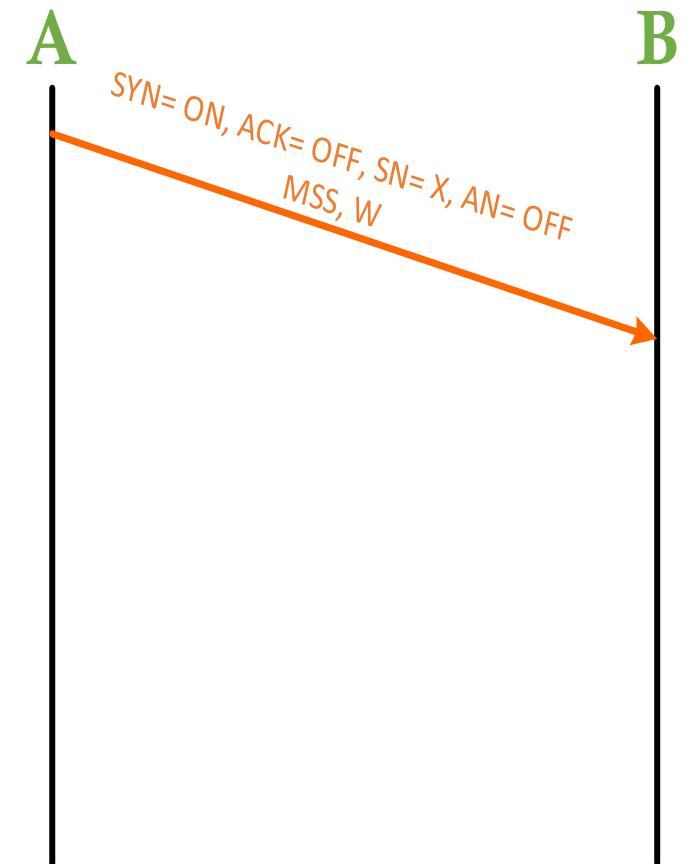
Fase1: Solicitud de conexión: TCP origen envía un segmento con los indicadores configurados de la siguiente manera:

***SYN = ON***

***ACK = OFF***

***SN = X***

***AN = OFF***



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: fases

### Fase2: Confirmación destino → origen

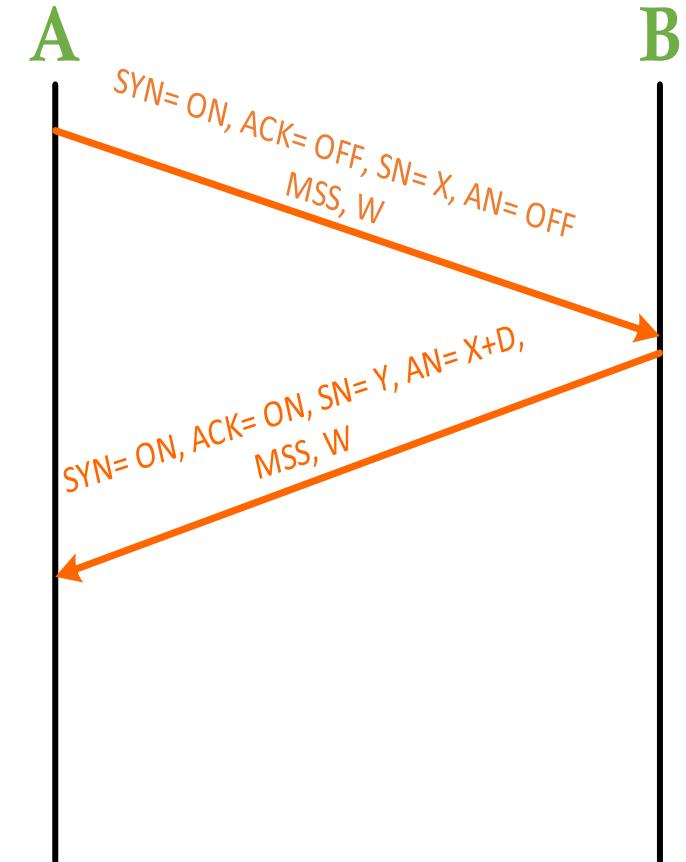
Cuando el segmento de solicitud de conexión es recibido por el módulo TCP destino, éste envía un segmento de confirmación aprovechando, además, para enviar datos al origen con el número de secuencia Y. El segmento de confirmación contiene:

***SYN = ON***

***ACK = ON***

***SN = Y***

***AN = X + D*** (donde D es la cantidad de bytes enviados en la primera fase. Con esta información TCP destino le indica a TCP origen que ha recibido D bytes y espera recibir datos en el próximo segmento, el que deberá tener ***SN = X + D***)



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: fases

### Fase3: Confirmación origen → destino

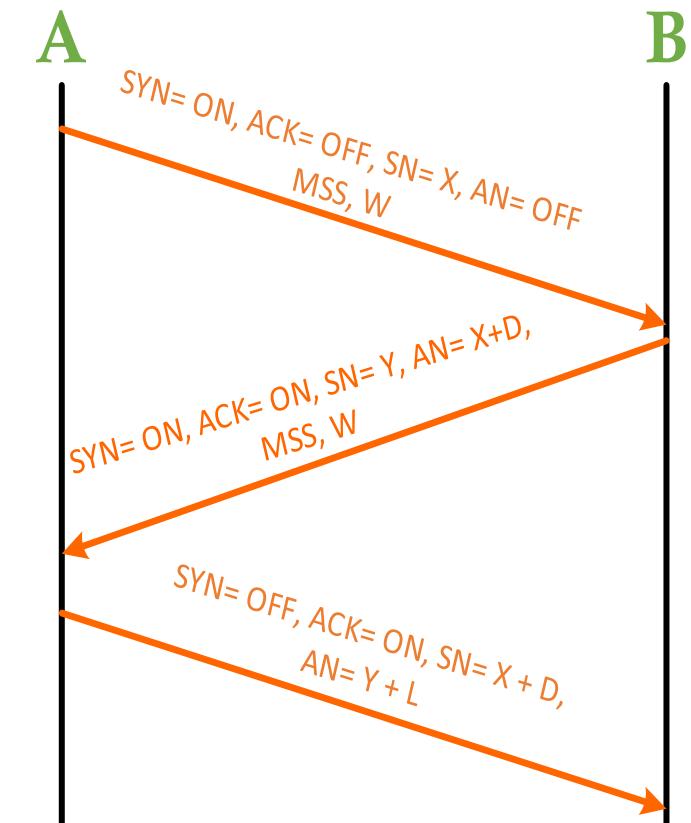
Cuando el módulo TCP origen recibe el segmento de la 2da. fase enviado por el TCP destino, el módulo TCP origen envía un segmento con:

$SYN = OFF$

$ACK = ON$

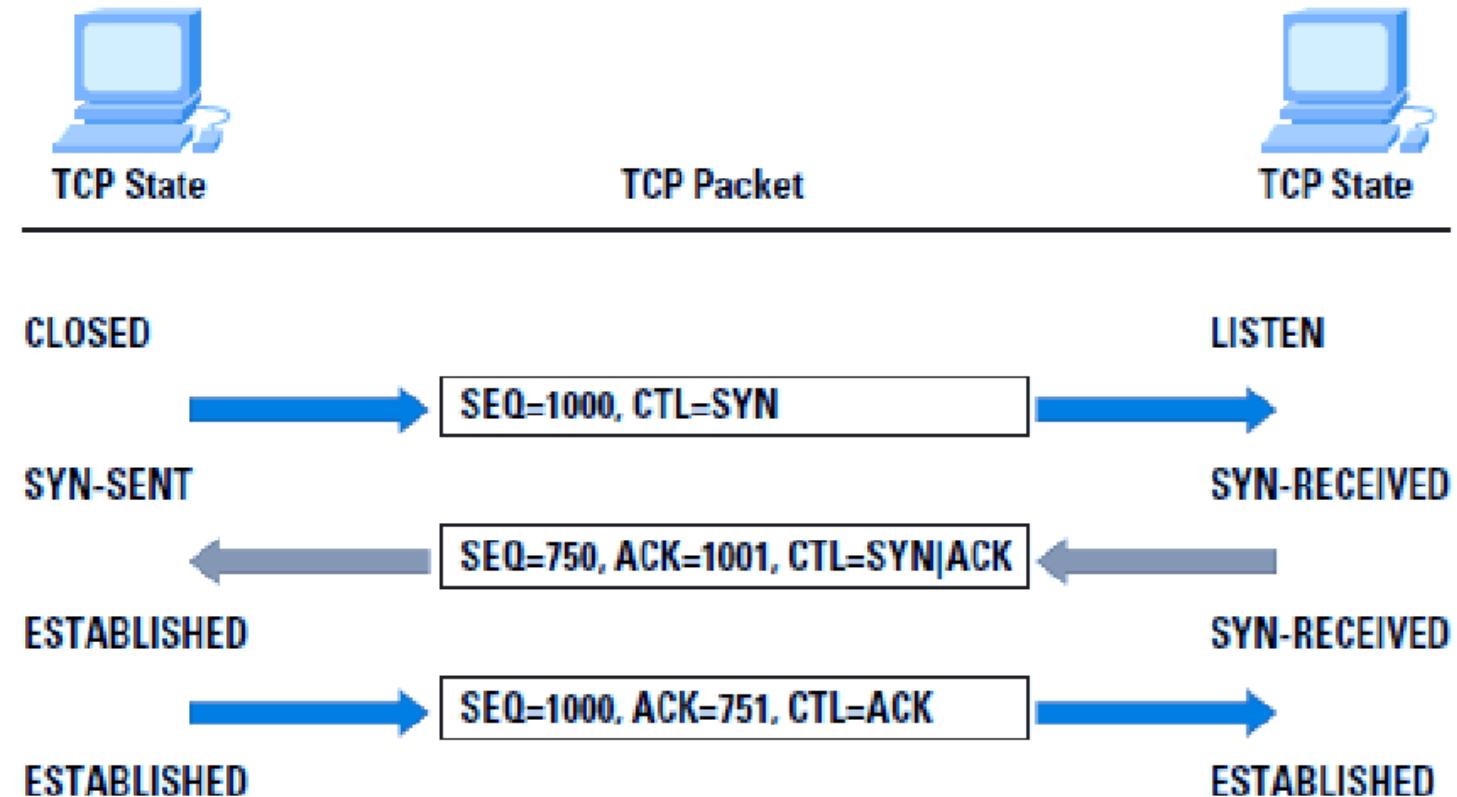
$SN = X + D$  (si envía datos. D es la cantidad de bytes enviados en el segmento anterior)

$AN = Y + L$  (donde L es la cantidad de bytes enviados en la Fase 2 por el extremo destino al origen). De nuevo, con esta información el TCP origen le indica al TCP destino que el próximo segmento con datos deberá tener  $SN = Y + L$



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Establecimiento de la conexión: estados de la conexión



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Transferir Datos

- ✓ Los datos de un mensaje (una cantidad dada de bytes) son transmitidos en segmentos en los cuales TCP lleva la cuenta byte por byte, usando el parámetro **SN**.
- ✓ Por ello, se puede ver la transferencia de datos, realizada por TCP, como un **“flujo de bytes”**
- ✓ Cada segmento contiene en su cabecera el número de secuencia (**SN**) que usa una numeración basada en módulo  $2^{32}$
- ✓ Los datos que recibe TCP (o UDP) de cada aplicación para ser transmitidos, son almacenados separadamente en bloques de memoria temporal (buffer) por el módulo de transporte
- ✓ De la misma forma, los segmentos que le llegan de la red son almacenados en un otro buffer específico (diferente al anterior)

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Transferir Datos

- ✓ Entrega de datos hacia la capa de aplicación y transmisión (hacia la capa de red)

Durante una transferencia de datos normal, tanto **TCP** o **UDP** envía los datos que recibe de las aplicaciones a la capa de red y recibe de la capa de red los datos para subir a las aplicaciones, de acuerdo a los tiempos establecidos en las implementaciones de los protocolos.

No obstante, se puede forzar a TCP o UDP a realizar estas operaciones mediante los indicadores **PUSH** y **URG**.

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Transferir Datos

**PUSH:** Indicador que es activado por el **usuario emisor**. Cuando está activado:

1. Obliga al TCP **emisor** a que tome los datos que le entrega, arme los segmentos correspondientes, y los envíe al otro extremo de inmediato
2. Ordena al módulo TCP **receptor** que comunique a la aplicación correspondiente que tiene datos para entregar
3. Especifica el **fin-de-bloque** al TCP local. Es decir, le indica que es el fin del mensaje generado por la aplicación para ser transmitido

**URG:** Bit (activado por la aplicación emisora)

1. Indica al TCP local que el campo **Puntero Urgente** tiene un **valor** que debe ser utilizado para conocer dónde terminan los datos que se deben entregar en forma urgente (**SN + Puntero Urgente = Fin de datos urgentes**)
2. Indica al TCP remoto, que informe a la aplicación destinataria que, en el flujo de datos entrantes, **existen datos “urgentes”**

# B R E A K

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Fin de la Conexión

La finalización de la conexión puede ser decidida tanto por TCP como por la Aplicación/Usuario

### 1. Finalización de la conexión decidida por TCP

RST (Bit): indicador que utiliza el módulo TCP de un extremo para indicar al otro que se da por finalizada la conexión

Se trata de una finalización **abrupta**. Esto sucede si, durante el intercambio de datos, llega un segmento que no pertenece a la conexión actual, o es un segmento duplicado o está retrasado fuera del límite establecido por el timer.

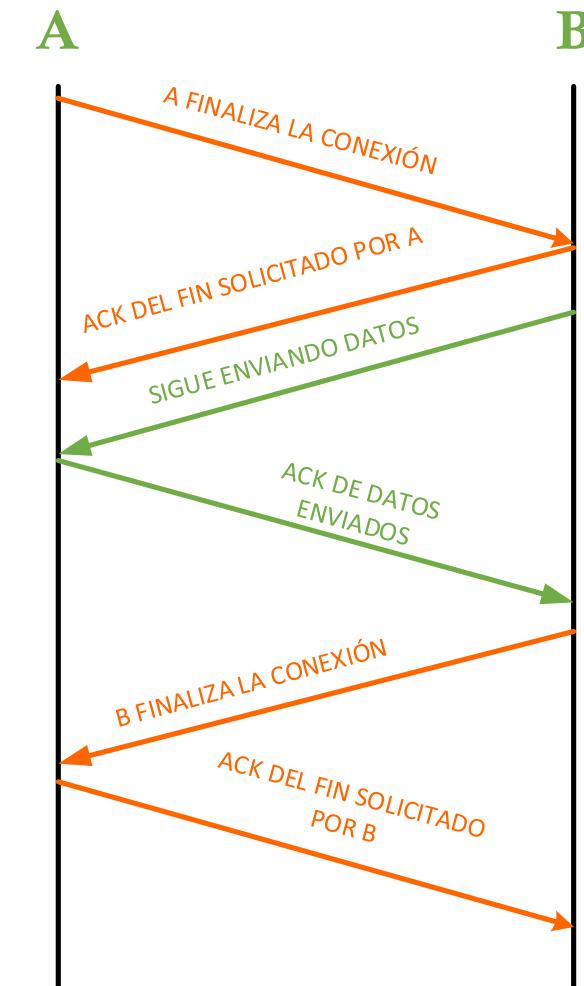
# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Fin de la Conexión

### 2. Finalización de la conexión decidida por el usuario

Una conexión puede finalizar de 2 maneras diferentes:

- ✓ Cierre ordenado (modo normal): el usuario ST (aplicación) activa una primitiva CLOSE. El procedimiento de cierre es similar al de establecimiento de la conexión: se realiza en 3 fases
- ✓ Cierre abrupto: el usuario ST puede requerirlo activando una primitiva ABORT. En ese caso TCP abandona todos los intentos para enviar o recibir más datos y descarta los datos almacenados en los buffers en memoria
- ✓ Para finalizar la conexión se necesitan 4 segmentos.



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Temporizadores (Timers)

1. Retransmisión (RTO)
2. Persistencia (Persist Timer)
3. “Keep Alive”
4. Temporizador del Establecimiento de la Conexión
5. Temporizador de Time Wait
6. Temporizador de Fin Wait
7. ACK Retardados (Controla Flujo)

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Temporizadores: Retransmisión Time Out (RTO)

- ✓ Es el temporizador utilizado para retransmitir segmentos cuando no llega el acuse de recibo correspondiente.
- ✓ Este temporizador es diferente en cada conexión y cambia durante una misma conexión. Se recalcula cada vez que se recibe un acuse de recibo:  $2 \times RTT$
- ✓ El RTT (Round-Trip Time) se puede calcular con:
  - ✓ La opción “marca de tiempo” (opción “Time Stamp”)
  - ✓ Enviando un segmento y midiendo el tiempo hasta el acuse:

Como una fórmula:  $RTT = \alpha \times RTT \text{ previo} + (1 - \alpha) \times RTT \text{ actual}$

Los acuses de recibo de segmentos retransmitidos no se tienen en cuenta en este cálculo (Algoritmo de Karn).

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Temporizadores: Persist Timer

- ✓ Cuando el emisor recibe un mensaje con  $W=0$ , queda en espera de un nuevo acuse indicando la apertura de la ventana.
- ✓ Si ese acuse con  $W \neq 0$  se pierde, los dos extremos de la conexión quedan bloqueados
- ✓ Para evitar esto, el emisor inicia un temporizador y cuando expira envía un segmento especial de prueba para que el receptor le informe del estado de su ventana
- ✓ El valor de este temporizador empieza siendo igual al de retransmisión pero se duplica cada vez, hasta un máximo de 60s.

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Temporizadores: Keep Alive

- ✓ Este temporizador se utiliza para evitar que una conexión que no trae datos, quede abierta indefinidamente.
- ✓ El servidor inicia el temporizador con un valor de 2 horas. Cada vez que recibe un segmento del cliente, reinicia el temporizador
- ✓ Si el temporizador expira, el servidor envía una interrogación al cliente. Si después de 10 pruebas enviadas cada 75 s, el cliente no contesta, el servidor cierra la conexión
- ✓ No es parte de la especificación TCP

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Temporizadores: Temporizador de establecimiento de la conexión

- ✓ Espera respuesta a un SYN durante este timer(75 seg).
- ✓ Para este tipo de segmentos (SYN) no se utiliza el RTO (no transporta datos el segmento).
- ✓ Si no recibe respuesta en 75 seg. abandona la conexión
- ✓ Este temporizador se usa en dos situaciones:
  - ✓ Después de que el cliente envía el SYN, ingresa al estado SYN\_SENT y espera el SYN + ACK del servidor.
  - ✓ El servidor recibe el SYN creado por la conexión, responde a SYN + ACK, ingresa al estado SYN\_RECV y espera el ACK del cliente.

# Transmission Control Protocol (TCP) – RFC 793 (1981)

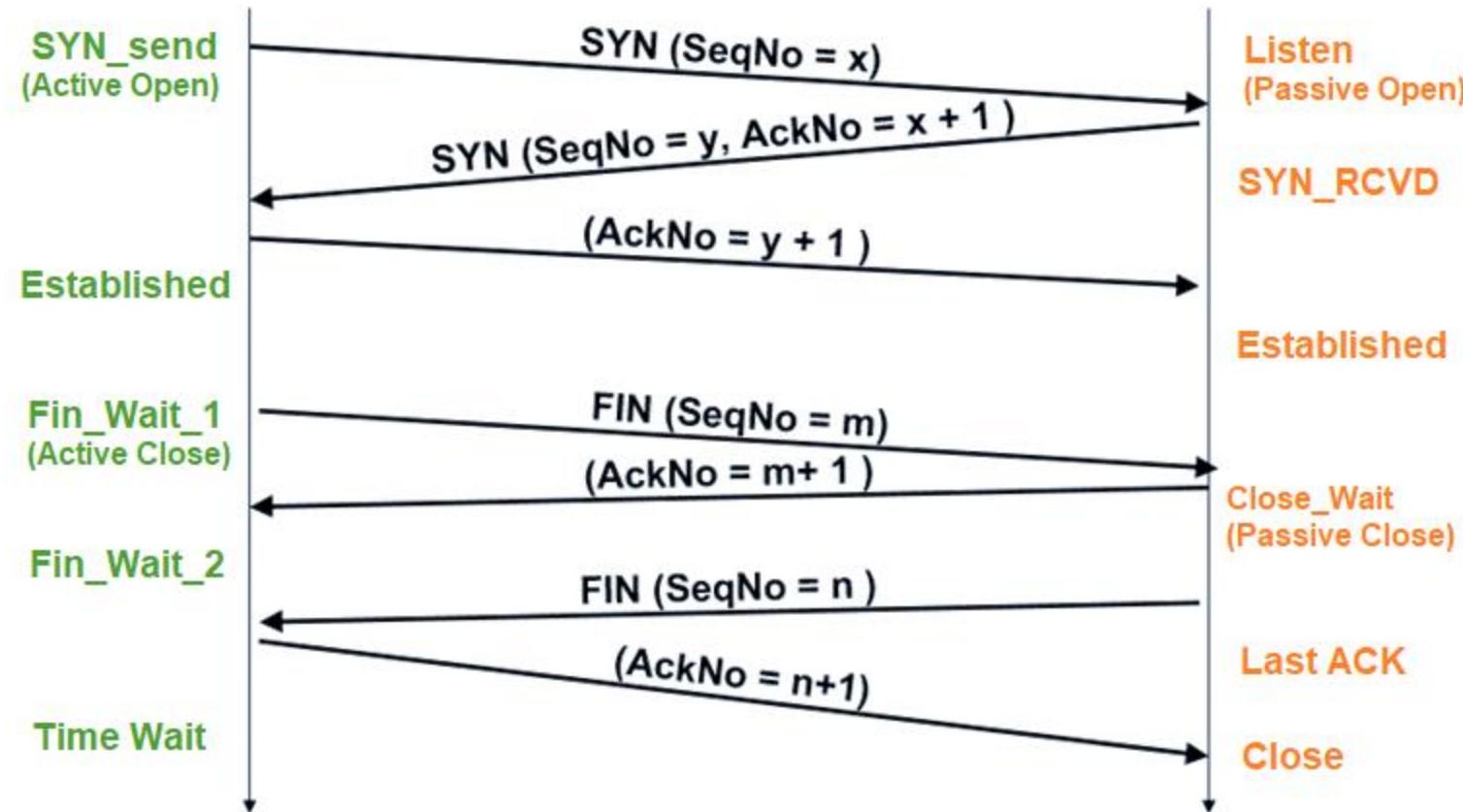
## Temporizadores: Temporizadores de fin la conexión

- ✓ Estado Fin\_Wait\_1: primer paso, el cliente ha enviado un requerimiento de cierre de la conexión al extremo remoto, el servidor.
- ✓ Estado Fin\_Wait\_2: El servidor ha enviado un acknowledgement al previo requerimiento de cierre de conexión hecho por el cliente. Por defecto este tiempo es de 10 minutos.
- ✓ Time Wait: En caso que el cliente ha recibido el segmento de FIN del servidor, o se ha consumido el tiempo del estado Fin-Wait\_2, inmediatamente el cliente inicia el temporizador de espera (Time Wait) para mantener la conexión “viva” durante un lapso igual a 2 MSL

MSL (Maximum Segment Lifetime) es por defecto igual a 2 minutos. Se usa para reenviar si fuera necesario el último ACK del cliente al servidor, y para evitar que la conexión sea reutilizada inmediatamente.

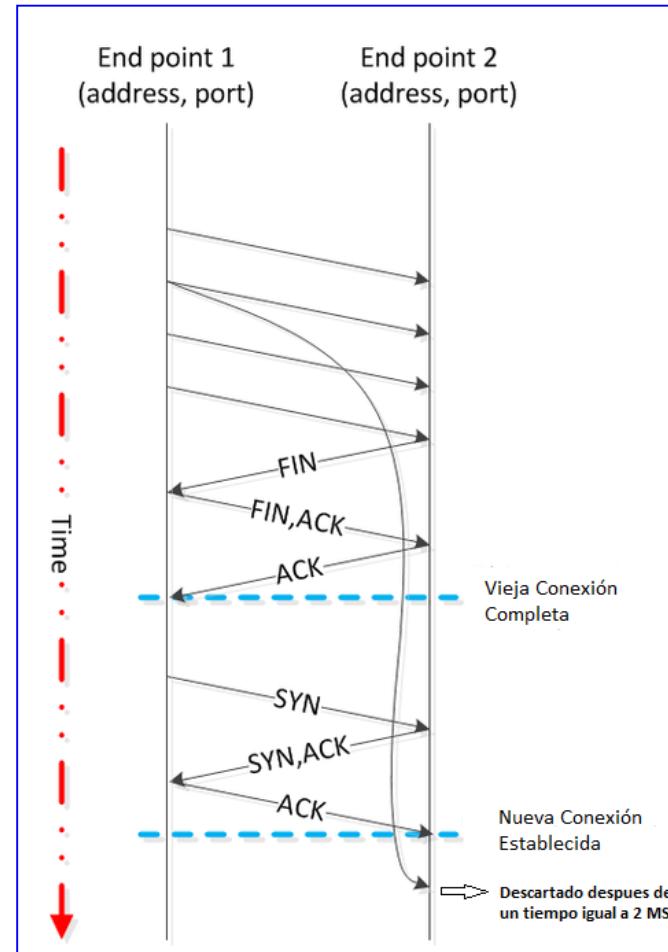
# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Temporizadores: Todos los temporizadores de fin la conexión



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Temporizadores: Vencido el tiempo, Time Wait descarta segmentos



# Transmission Control Protocol (TCP) – RFC 793 (1981)

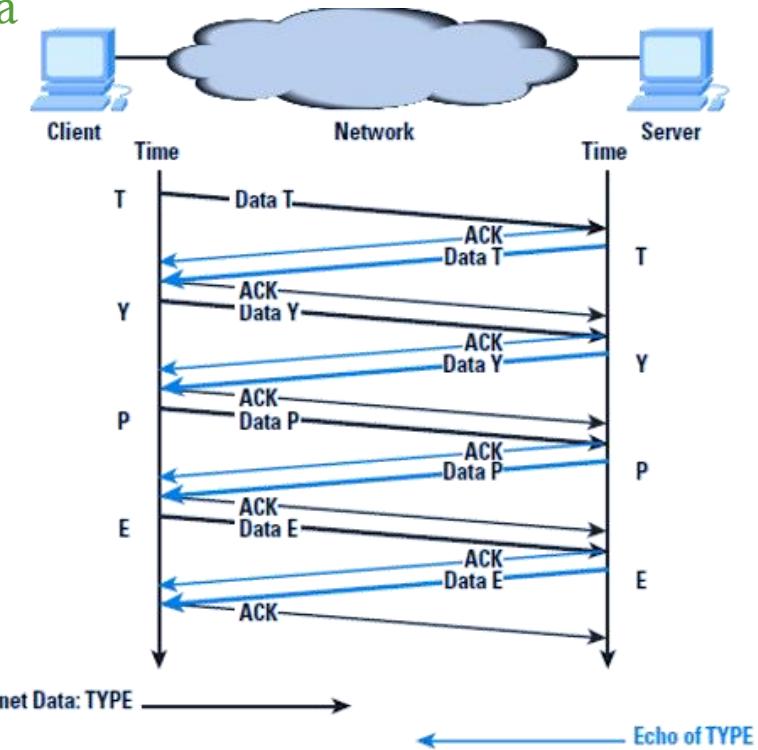
## Temporizadores: ACK Retardados: Síndrome de Silly Windows

### Problema

- ✓ El síndrome de la ventana trivial (*silly window*) se produce cuando:
  - ✓ La aplicación emisora genera datos a un ritmo muy lento (ej. byte a byte)
  - ✓ La aplicación receptora consume datos a un ritmo muy lento
- ✓ Ventana trivial en el emisor (ej. Aplicaciones interactivas)
  - ✓ Cada carácter necesita 4 mensajes TCP/IP (40bytes de cabeceras)
  - ✓ Un carácter (1 bytes) usa más de 160 bytes

### Solución: Algoritmo de Nagle

- ✓ Emisor envía el primer mensaje que llega de la Aplicación (aunque sea un sólo byte)
- ✓ Los siguientes mensajes que llegan de la aplicación, se retrasan hasta que:
  - ✓ Se recibe un ACK del receptor (la red está libre)
  - ✓ Se completa el buffer en el emisor (generalmente el buffer tiene un tamaño igual MSS bytes, que se llenará con datos de la aplicación)
  - ✓ Expira un temporizador ACK Retardado



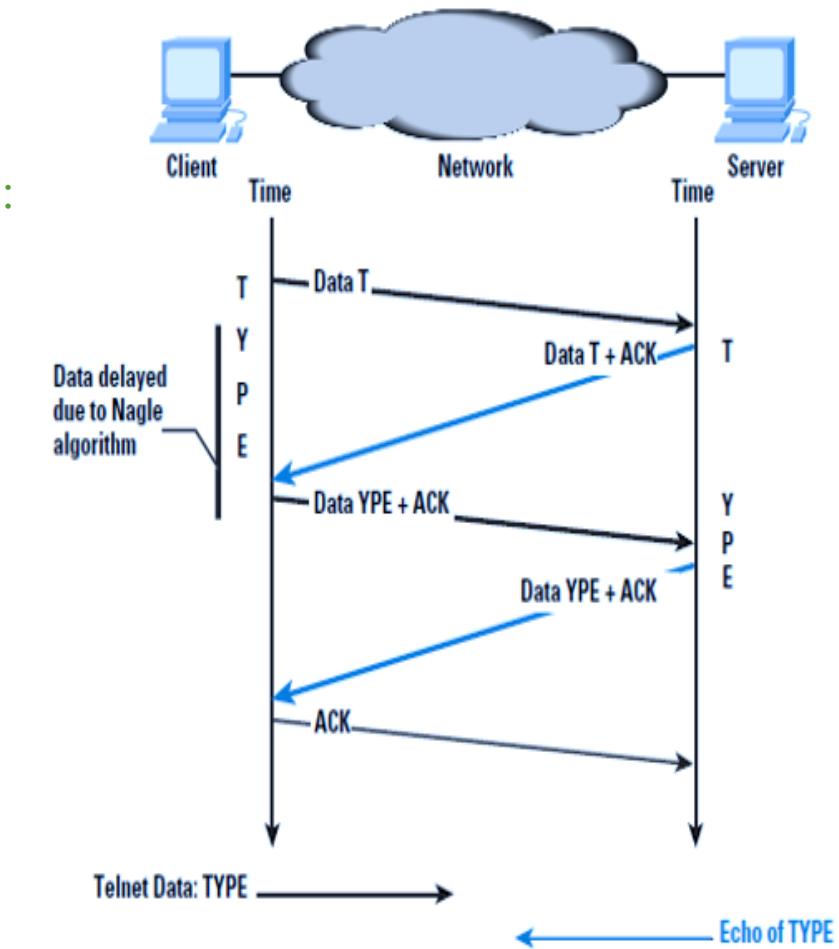
# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Temporizadores: Ack Retardados: Síndrome de Silly Windows

### Nagle

```
if hay nuevos datos para enviar:  
    if el tamaño de ventana y los datos disponibles >= MSS:  
        enviar un segmento completo de tamaño de MSS  
        ahora  
    else:  
        if hay datos sin confirmar en espera:  
            encolar los datos en el buffer hasta recibir un  
            ACK  
        else:  
            enviar los datos ahora
```

- ✓ Retardar el timer del Ack hasta 500 ms
- ✓ Piggybacking el Ack con el tráfico en sentido inverso



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Control de Congestión, conceptos

- ✓ El emisor utiliza el ritmo de llegada de confirmaciones para regular el ritmo de envío de segmentos de datos
- ✓ Esto se implementa mediante la **ventana de congestión (CW)**
  - ✓ La ventana de congestión es complementaria a la **ventana de recepción (RW)** usada para el control de flujo
  - ✓ En una situación de no congestión (sin pérdida o retraso de segmentos) la ventana de congestión alcanza el mismo tamaño que la ventana de recepción (**CW=RW**)
  - ✓ Cuando se produce una situación de congestión el tamaño de CW se va reduciendo progresivamente
  - ✓ Cuando la situación de congestión desaparece, el tamaño de CW se va aumentando progresivamente
  - ✓ El número máximo de bytes que puede enviar el emisor (**AW, Allowed Window**) es el mínimo de ambos tamaños de ventana:

$$\mathbf{AW = \min \{ RW, CW \}}$$

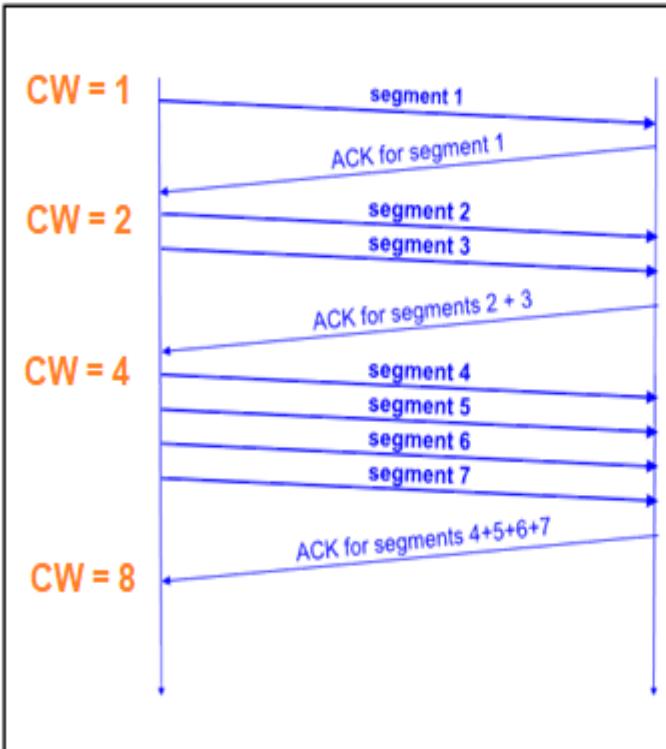
# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Control de Congestión

- ✓ La red está sin congestión cuando no se pierden o retrasan segmentos
- ✓ La transmisión comienza con un tamaño de ventana de congestión CW = 1
  - ✓ El emisor envía un único segmento de tamaño máximo igual a MSS
- ✓ A continuación, la CW va aumentando, pasando por tres fases distintas:
  - ✓ **Fase de arranque lento (*slow start*)**
    - ✓ La CW se incrementa en uno por cada segmento enviado y confirmado
    - ✓ Esto provoca un crecimiento exponencial (CW = 1, 2, 4, 8, 16, 32, ...)
    - ✓ Esta fase termina cuando el tamaño de CW alcanza un cierto umbral, denominado umbral de arranque lento (**STT, Slow Start Threshold**)
    - ✓ Inicialmente, el valor del STT suele ser de 64 Kbytes
  - ✓ **Fase evitando la congestión (*congestion avoidance*)**
    - ✓ A partir del STT, la **CW** se incrementa en 1 cada vez que se envía y se confirma una ventana completa (es decir, **CW** segmentos)
    - ✓ Esto provoca un crecimiento lineal
    - ✓ Esta fase termina cuando la **CW** alcanza el tamaño de la ventana de recepción (**RW**)
  - ✓ **Fase constante**
    - ✓ En esta fase, la CW se mantiene a un valor constante (**CW = RW**)

# Transmission Control Protocol (TCP) – RFC 793 (1981)

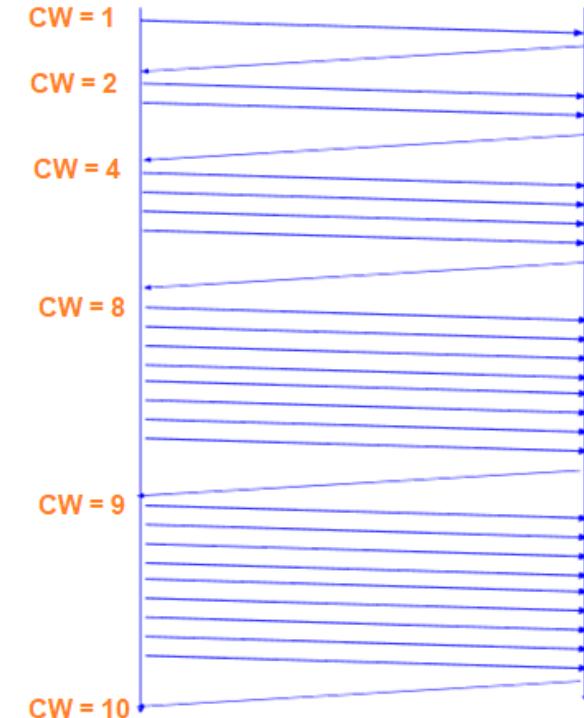
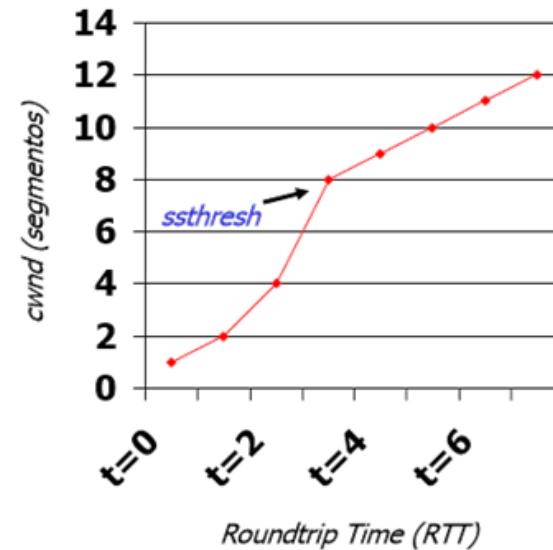
## Control de Congestión



**Slow Start**

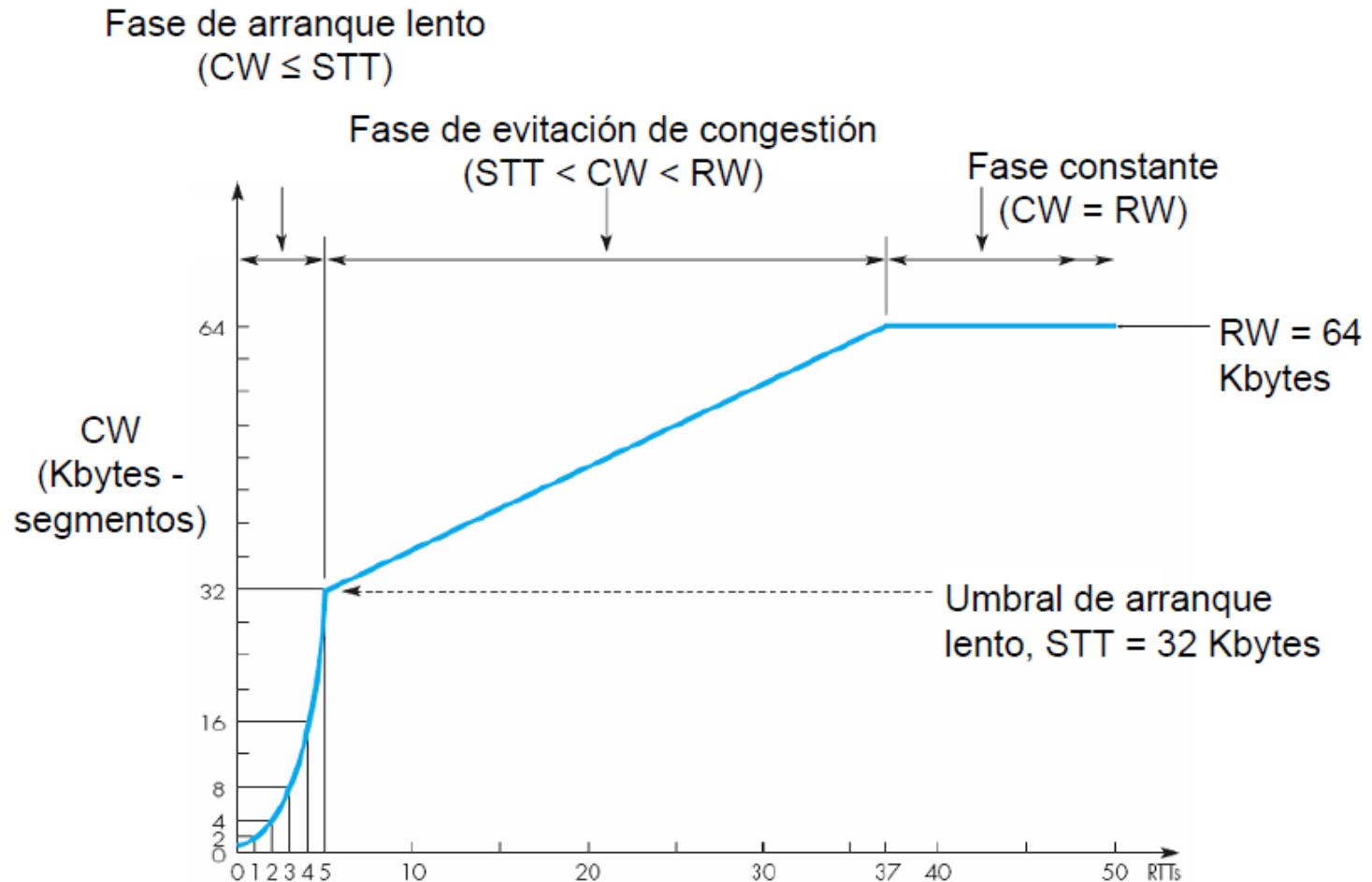
TCP detiene el crecimiento de CW cuando:  
**CW = STT**

**Evitando la congestión**



# Transmission Control Protocol (TCP) – RFC 793 (1981)

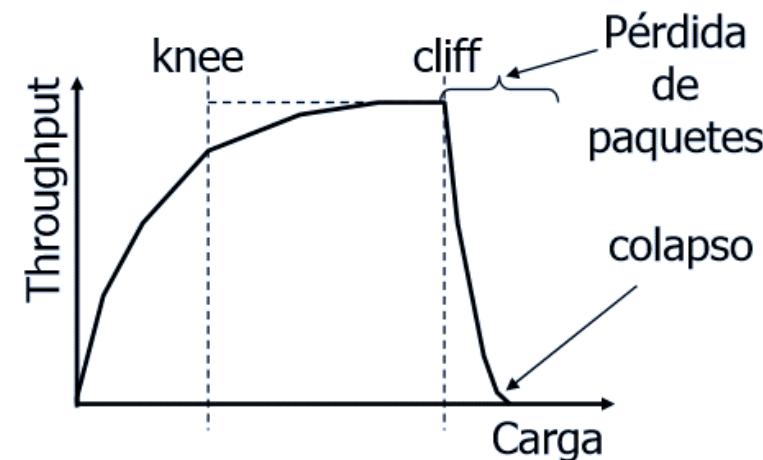
## Control de Congestión



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Control de Congestión: el objetivo es “controlar”

- ✓ Controlar
  - ✓ Permanecer a la izquierda del cliff
- ✓ Evitar
  - ✓ Permanecer a la izquierda del knee



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Control de Congestión: mejoras

- ✓ Esperar RTO (Retransmission Timeout). Después de transcurrido este tiempo TCP considera que hay congestión.
- ✓ RTO es usualmente dos veces RTT.
- ✓ Esperar RTO puede degradar de performance.
- ✓ No esperar RTO
  - ✓ Utilizar mecanismos alternativos.
  - ✓ Utilizar RTO si fallan los anteriores.

# Transmission Control Protocol (TCP) – RFC 793 (1981)

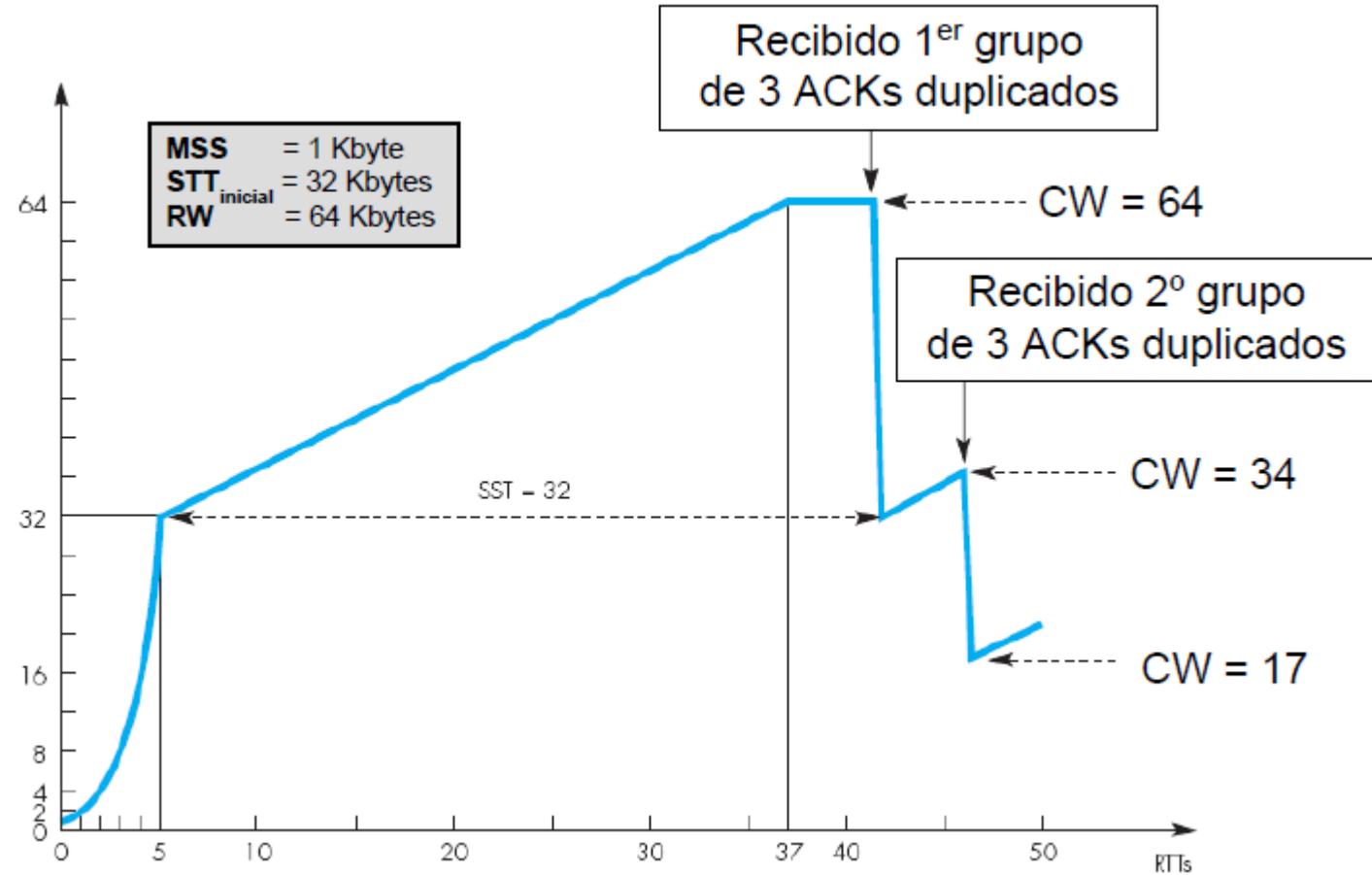
## Control de Congestión: mejoras

No espera un RTO, para suponer una congestión, se tiene en cuenta lo siguiente:

- ✓ Frente a un segmento fuera de orden se debe enviar un ACK.
- ✓ Provoca duplicación de ACK's.
- ✓ Esta **duplicación** se ve como debida a:
  - ✓ Paquetes perdidos (1)
  - ✓ Reordenamiento de paquetes (2)
- ✓ Con 2 ACK duplicados, no se sabe si es 1 o 2
- ✓ Si se reciben 3 ACK's duplicados se **considera** que se debe a un **paquete perdido** (1)
  - ✓ Al recibir el tercer ACK repetido se retransmite sin esperar el RTO y se hace  $CW = CW / 2$ . Eso es **Fast Retransmit**
  - ✓ Luego se ejecuta “congestion avoidance”, no slow start. Eso es **Fast Recovery**

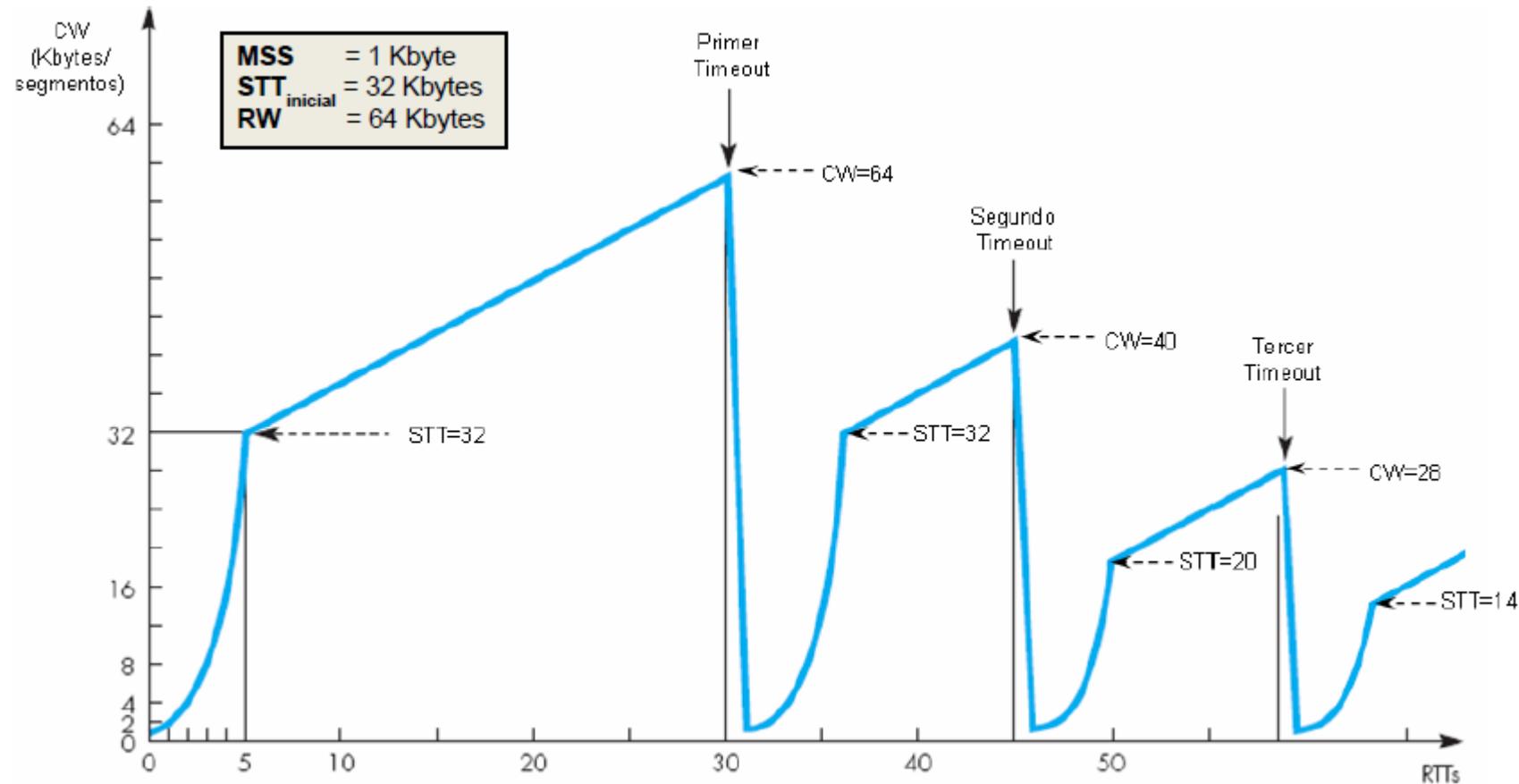
# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Control de Congestión: congestión leve (tres ACK's duplicados)



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Control de Congestión: congestión elevada (expiración RTO)



# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Control de Congestión: integrando todo

- ✓ Slow Start.
- ✓ Congestion Avoidance.
- ✓ Si aparecen ACK's duplicados
  - ✓ Fast Retransmit y Fast Recovery.
  - ✓ Congestion Avoidance.
- ✓ Si RTO
  - ✓ Slow Start.
- ✓ Resumiendo: TCP Reno

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Control de Congestión: muchas soluciones

- ✓ TCP Vegas
- ✓ TCP New Reno
- ✓ TCP Hybla
- ✓ TCP BIC
- ✓ TCP CUBIC
- ✓ Agile-SD TCP
- ✓ Compound TCP
- ✓ TCP Proportional Rate Reduction
- ✓ TCP BBR
- ✓ Etc.

# Transmission Control Protocol (TCP) – RFC 793 (1981)

## Control de Congestión: cooperación entre IP y TCP

- ✓ El campo ECN (cabecera IP) se pone a 10 o 01 en los paquetes transmitidos por el emisor para indicar que ECN es soportado por las entidades de transporte para estos paquetes.
- ✓ Un router con capacidad ECN detecta una congestión inminente y detecta que un ECN está establecido en 10 o 01 en el paquete que está a punto de descartar. En lugar de descartar el paquete, el router elige establecer el campo a 11 en la cabecera IP y reenvía el paquete.
- ✓ El receptor recibe el paquete con el valor 11, y establece en 1 el campo ECN-Echo (cabecera TCP) en su siguiente ACK TCP enviado al emisor.
- ✓ El remitente recibe el ACK TCP con ECN-Echo activado y reacciona a la congestión como si se hubiera perdido un paquete.
- ✓ El emisor establece el valor 1 al campo CWR en la cabecera TCP del siguiente paquete enviado al receptor para notificarlo de haber recibido el aviso y haber reaccionado ante el valor del ECN-Echo. Normalmente, la reacción es reducir la ventana de congestión.

# User Datagram Protocol (UDP) – RFC 768

## Conceptos y Usos

- ✓ Brinda a las aplicaciones un servicio No Orientado a conexión, no confiable: no están garantizadas la entrega de los segmentos.
- ✓ Necesita muy poca información. Es rápido y liviano. Se usa para tareas de gestión en la interred
- ✓ Es usado por aplicaciones que realizan tareas de recolección de datos de entrada. Supone una actividad periódica o muestreo pasivo de datos, tales como los procedentes de sensores, de informes entre equipos de red (routers y demás componentes).
- ✓ Implementado para diseminar datos de salida. Incluye la difusión de mensajes a los usuarios de la red o interred: Ejemplo, anuncio de la puesta en servicio de un nuevo router, el cambio de la dirección de un servicio, la distribución de los valores de un reloj en tiempo real

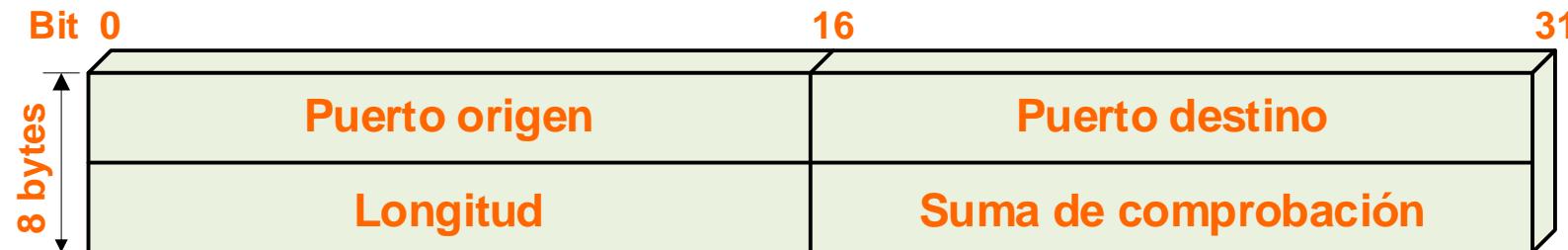
# User Datagram Protocol (UDP) – RFC 768

## Conceptos y Usos

- ✓ **Petición-respuesta.** Usado por aplicaciones para resolver nombres, como DNS (Domain Name System) o aplicaciones para controlar y recolectar información de dispositivos usando protocolos como el SNMP.
- ✓ **Aplicaciones en tiempo real.** Aplicaciones típicas son las transmisiones de voz, de video en vivo y la tele medición:
  - ✓ Voz y video en vivo: deben transmitirse en tiempo real
  - ✓ El carácter de estas aplicaciones no aceptan mecanismos como la retransmisión o la confirmación, que son utilizados por transmisiones orientadas a conexión.

# User Datagram Protocol (UDP) – RFC 768

# Formato de la Cabecera del Segmento



- ✓ Debido a que UDP no provee confiabilidad, tiene pocas funciones que realizar y, por ende, la cabecera tiene poca información
  - ✓ La cabecera incluye un **puerto origen (16 bits)** y un **puerto destino (16 bits)**
  - ✓ El **campo de longitud (16 Bits)**, contiene la longitud del segmento UDP entero (cabecera + datos)
  - ✓ La **suma de comprobación (16 Bits)** se usa en el mismo algoritmo de TCP e IP. La suma de verificación se aplica al segmento UDP entero más una pseudocabecera que es la misma que la usada en TCP. Si se detecta un error, el segmento se descarta sin tomar ninguna medida adicional
  - ✓ Obsérvese que el único procesamiento que realiza el módulo UDP es el algoritmo de comprobación de error

# Temas a tratados

1. Suite de Protocolos de Interred
2. Protocolos de transporte: multiplexado y demultiplexado de conexiones TCP y UDP
3. Protocolo TCP
  1. Formato de la Trama
  2. Establecimiento de la conexión
  3. Transferencia de datos
  4. Cierre de la conexión
  5. Temporizadores
  6. Control de Congestión
4. Protocolo UDP

# FINAL DEL MÓDULO 8

---