TEMA 6

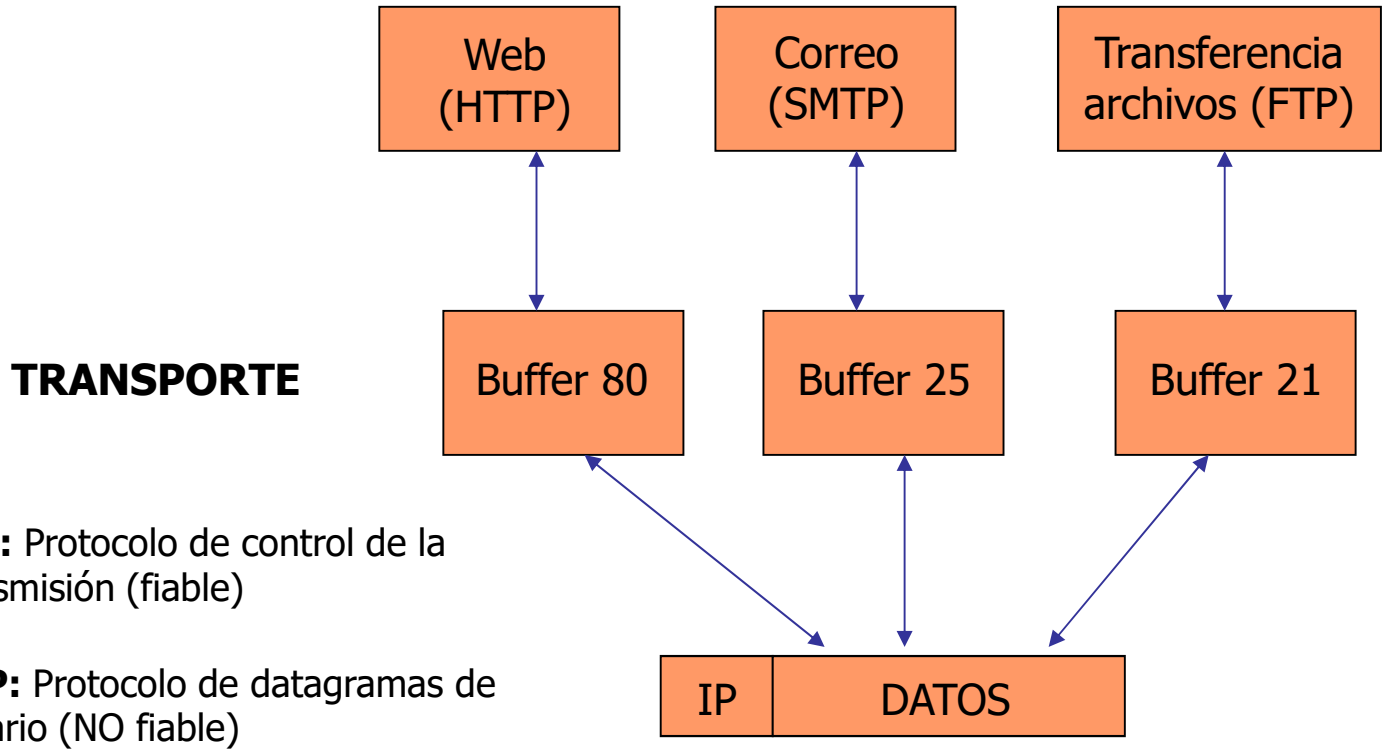
NIVEL DE TRANSPORTE

José Ángel Berná Galiano. DFISTS. Universidad de Alicante.

6.1 Funcionalidades del nivel de transporte

6.1.1 Interfaz capa de aplicación-capa de red

Interfaz entre la capa de aplicación y red para la gestión de comunicaciones extremo a extremo (conexiones) entre equipos de Internet.



TCP: Protocolo de control de la transmisión (fiable)

UDP: Protocolo de datagramas de usuario (NO fiable)

6.1 Funcionalidades del nivel de transporte

6.1.2 Multiplexión de conexiones

Un socket está identificado por el conjunto:

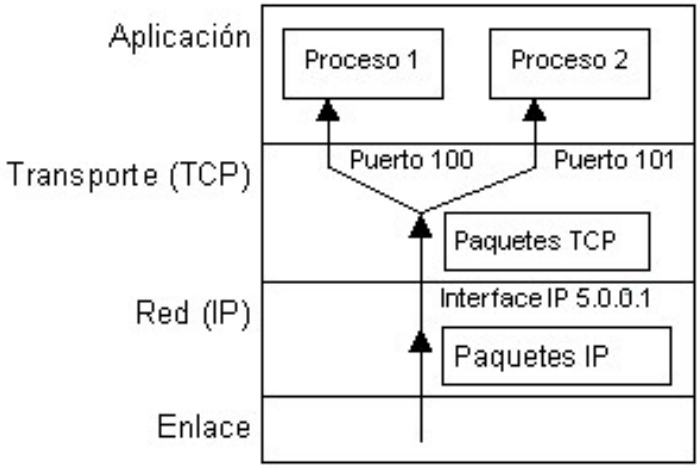
IP_origen:puerto_origen -> IP_destino:puerto_destino

IP	TCP	
5.0.0.1 -> 5.0.0.2	100 -> 200	DATOS

IP	TCP	
5.0.0.2 -> 5.0.0.1	200 -> 100	DATOS

6.1 Funcionalidades del nivel de transporte

6.1.2 Multiplexión de conexiones



La capa de transporte soporta múltiples conexiones entre un par de equipos empleando el número de puerto para separar los paquetes IP de cada conexión.

6.2 Protocolo de Datagramas de Usuario (UDP)

6.2.1 Funcionalidades

El protocolo UDP (User Datagram Protocol) está definido en RFC 768

Las características principales de este protocolo son:

Sin conexión. No emplea ninguna sincronización origen – destino.

Trabaja con paquetes o datagramas enteros, no con bytes individuales como TCP. Una aplicación que emplea el protocolo UDP intercambia información en forma de bloques de bytes, de forma que por cada bloque de bytes enviado desde la capa de aplicación a la capa de transporte, se envía un paquete UDP.

No es fiable. No emplea control del flujo ni ordena los paquetes.

Su gran ventaja es que provoca **poca carga adicional en la red**, ya que es sencillo y emplea cabeceras muy simples.

Un paquete **UDP puede ser fragmentando por el protocolo IP** para ser enviado fragmentado en varios paquetes IP si resulta necesario.

Un paquete UDP admite utilizar como dirección IP de destino la **dirección de broadcast** de la red IP ya que no emplea conexiones

6.2 Protocolo de Datagramas de Usuario (UDP)

6.2.1 Funcionalidades

Formato del paquete UDP

Puerto fuente y puerto destino. Valores de 16 bits correspondientes a los puertos de nivel de transporte.
Longitud. Número total de bytes en el paquete UDP original (incluye cabecera y datos), antes de ser fragmentado en paquetes IP.
SVT. Suma de verificación, aplicada a la cabecera y datos UDP, además de a algún campo de la cabecera IP.

6.2.2 Aplicaciones

Transmisión de datos en LAN's fiables. Por ejemplo con TFTP.

Operaciones de sondeo. Protocolos DNS, SNMP y NTP, servicios echo y daytime.

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.1 Funcionalidades

El protocolo TCP (Transmision Control Protocol) está definido en el documento RFC 793

Las características principales de este protocolo son:

Trabaja con un flujo de bytes. El nivel de aplicación entrega o recibe desde el nivel de transporte bytes individuales. TCP agrupa esos bytes en paquetes de tamaño adecuado para mejorar el rendimiento y evitar a la vez la fragmentación a nivel IP.

Transmisión orientada a conexión. Se requiere una secuencia de conexión previa al envío - recepción de datos entre los extremos de la comunicación, y una desconexión final.

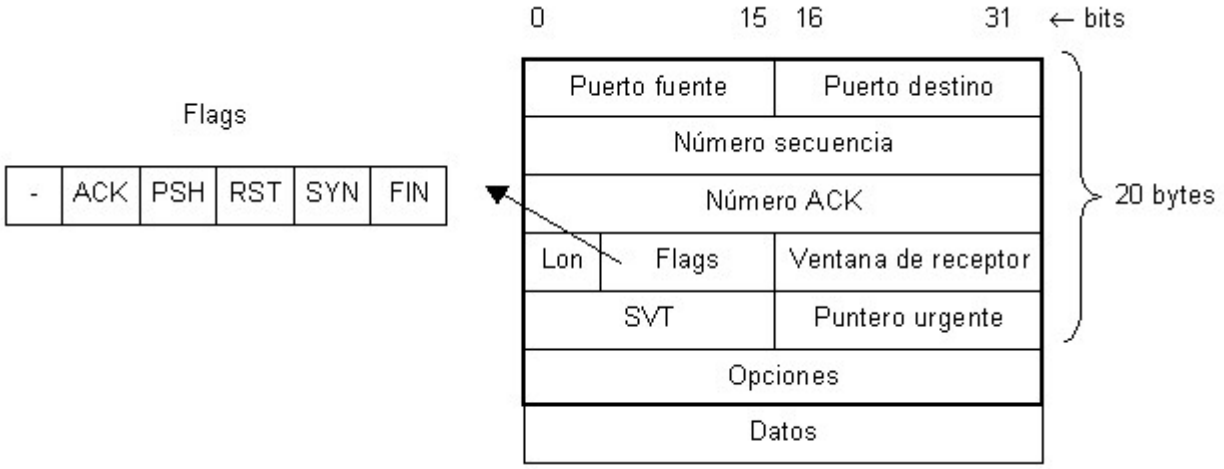
Fiable. Emplea control de flujo mediante ventana deslizante de envío continuo y asentimientos positivos o ACKs para confirmar las tramas válidas recibidas. La ventana deslizante se aplica a los bytes: se numeran y confirman bytes y no paquetes.

Flujo de bytes ordenado. Aunque IP trabaja con datagramas, un receptor TCP ordena los paquetes que recibe para entregar los bytes al nivel superior en orden.

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.1 Funcionalidades

Formato del paquete TCP



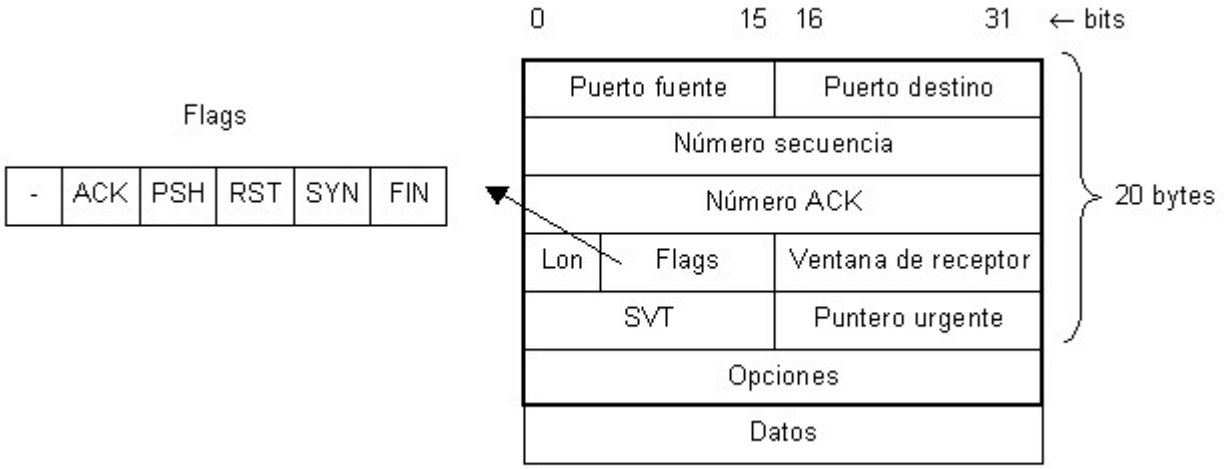
Puerto fuente y puerto destino. Valores de 16 bits correspondientes a los identificadores de los puertos de nivel de transporte.

Número de secuencia. Número de secuencia de numeración del primer byte del campo de datos del paquete.

Número de ACK. Número de la siguiente secuencia de numeración de los bytes del campo de datos que se espera recibir en un próximo paquete.

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.1 Funcionalidades



Flags. Campo con bits con significado propio, donde los más relevantes son:

ACK. Cuando toma el valor 1 indica que el número de ACK es válido y debe interpretarse, es decir, el paquete tiene información de asentimiento.

PSH (push). Cuando toma el valor 1 indica que la capa de transporte debe pasar los datos a la capa de aplicación sin esperar a recibir más datos.

RST *(*reset*).* Indica un rechazo de la conexión. Se usa cuando ha habido un problema en la secuencia de bytes, cuando falla un intento de iniciar conexión o para rechazar paquetes no válidos.

SYN *(*synchronice*).* Se utiliza para solicitar establecimiento de una conexión.

FIN. Se utiliza para solicitar la liberación de una conexión.

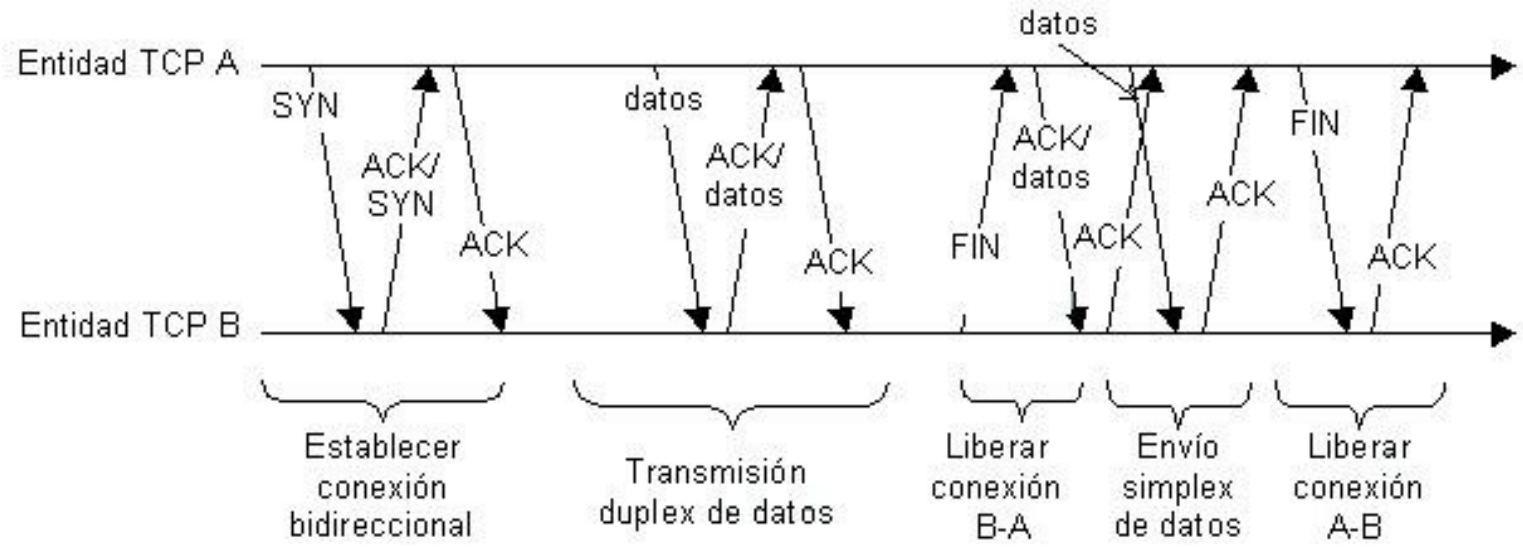
Ventana. Sirve para informar sobre el número de bytes que el emisor del paquete es capaz de recibir en su buffer de recepción. Si vale 0 indica que no puede recibir datos (aunque sí puede interpretar los paquetes con flags ACK, RST, FIN...).

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.2 Gestión de la conexión

Secuencia de funcionamiento de TCP

- Establecimiento bidireccional de la conexión.
- Intercambio de datos.
- Liberación bidireccional de la conexión.

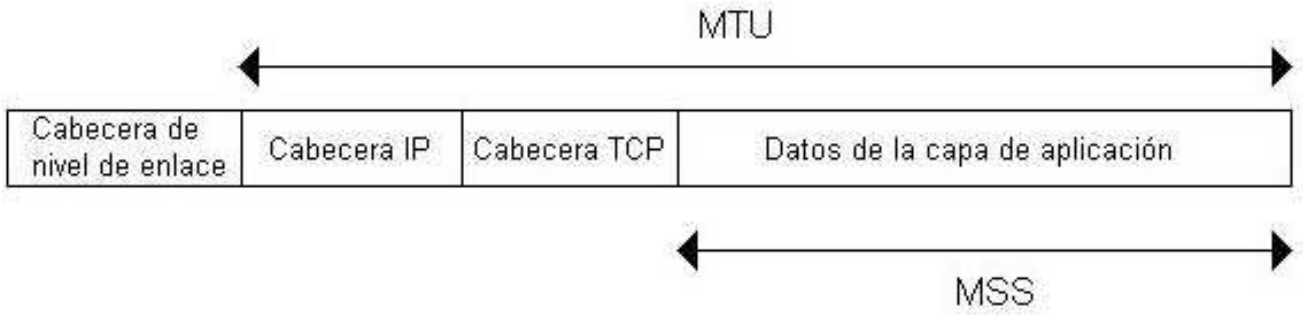


6.3 Protocolo de Control de la Transmisión (TCP)

6.3.2 Gestión de la conexión

MSS y norma RFC 1191

Se define el MSS (*Maximum Segment Size*) como la cantidad máxima de datos que puede incorporar un paquete (segmento) TCP. Este valor depende del MTU de la red donde se transmite el paquete TCP.



Para evitar la fragmentación IP, en el establecimiento de la conexión se negocia el valor del MSS. Este valor se intercambia en el campo de opciones de los paquetes SYN de establecimiento de conexión. Como MSS se establece el menor de los intercambiados por los extremos.

Si en una red intermedia entre origen y destino existe un MSS menor que el negociado, la norma RFC 1191 permite reducir el MSS. Para ello se activa el bit don't fragment en la cabecera IP de los paquetes TCP y se emplean los mensajes ICMP *Destination Unreachable* para configurar MSS menores en una conexión determinada.

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.3 Control del flujo de datos

Si la red no proporciona ningún mecanismo para controlar la congestión, éste ha de llevarse a cabo con los protocolos de la arquitectura de red.

El protocolo de la capa de transporte TCP es un protocolo que presenta las características de:

a) Protocolo fiable con confirmación de paquetes.
b) Transmisión orientada a conexión.
c) Control del flujo de bytes.

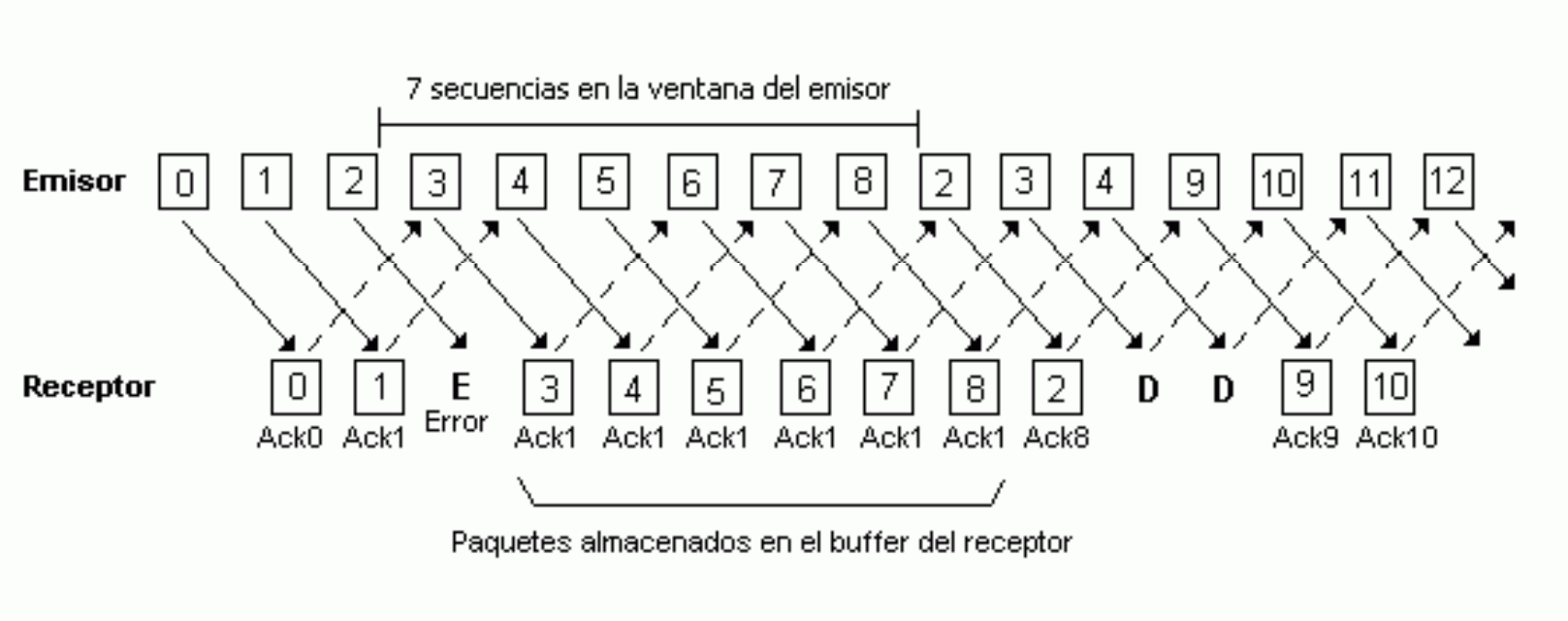
El control del flujo de bytes permite un control de la congestión, adaptándose TCP al retardo en el envío de la información en la red.

TCP emplea un algoritmo de ventana deslizante para el control del flujo.

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.3 Control del flujo de datos

Funcionamiento de un protocolo de ventana deslizante. Caso W_e=7 y W_r=7



6.3 Protocolo de Control de la Transmisión (TCP)

6.3.3 Control del flujo de datos

TCP emplea números de secuencia de bytes y tamaños de ventana en bytes

El control del flujo se realiza variando el tamaño de la ventana del receptor (campo window en la cabecera TCP):

a) Si la ventana del receptor aumenta, el emisor puede enviar más información sin esperar a recibir ACK (aumenta ventana del emisor).

b) Si la ventana del receptor disminuye, el emisor envía menos información sin esperar a recibir ACK (disminuye ventana del emisor). Caso límite: window=0.

Pérdida de segmentos. Reenvío de la información.

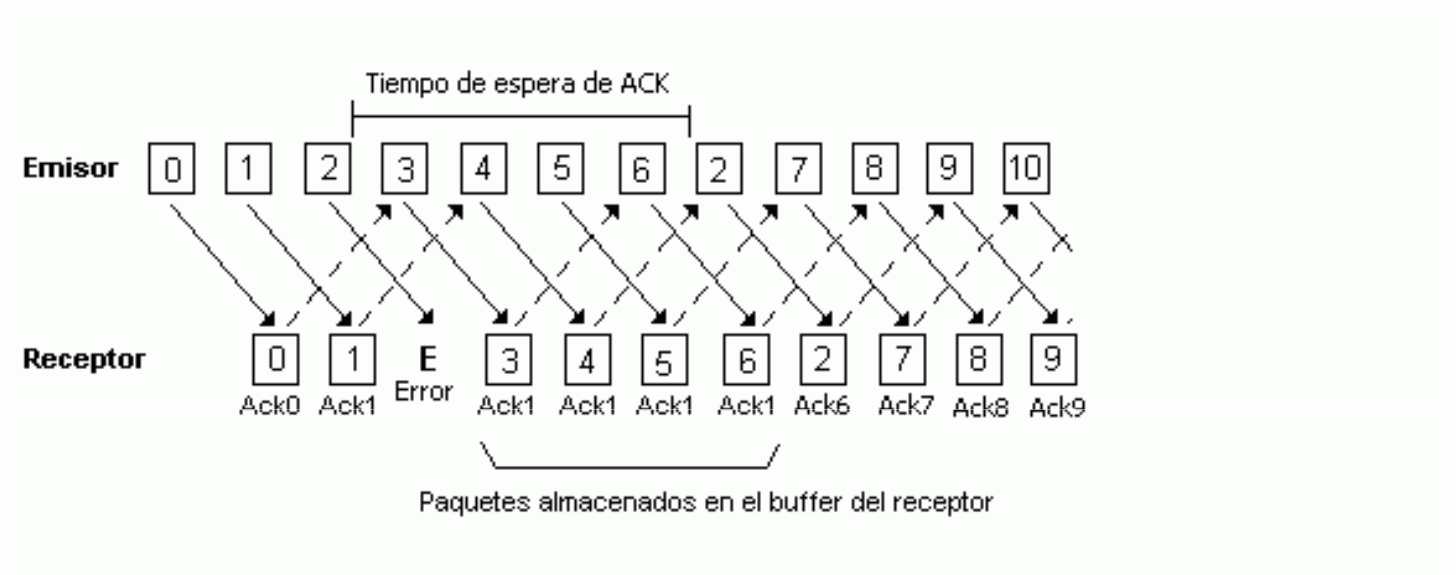
Segmento: paquete TCP con datos.

Cada vez que TCP recibe un ACK, la ventana del emisor permite enviar un nuevo fragmento.

Si un segmento no llega al receptor o llega con errores, el receptor no enviará ACK. Los siguientes segmentos que envíe el emisor (hasta su tamaño de ventana máximo) se almacenarán en el buffer del receptor pero éste enviará ACK de la secuencia previa al paquete erróneo.

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.3 Control del flujo de datos



El emisor tiene especificado un tiempo de espera de ACK para cada segmento. Si el ACK no llega se procede con el reenvío del primer segmento sin ACK en la ventana del emisor.

Para evitar reenvíos inútiles se espera al ACK del reenvío, así se comprobará que hay que continuar con otro segmento distinto del siguiente en espera.

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.3 Control del flujo de datos

Cálculo del tiempo de espera de ACK. Algoritmo de Karn.

El tiempo de espera de un ACK (Timeout) debe ser calculado de forma que:

- Sea lo suficientemente grande para evitar que los retardos en la red no provoquen reenvíos innecesarios por retardos en el envío del ACK.
- Sea lo suficientemente pequeño para que no haya periodos de inactividad en el envío de datos en la red.

El valor del timeout se calcula de forma dinámica durante el funcionamiento de TCP a partir del RTT (Round Trip Time) o tiempo de ida y vuelta. Este RTT se calcula como el tiempo transcurrido desde el envío de un segmento y la llegada de su ACK.

El timeout se calcula como Timeout=β*RTT. El RTT se actualiza en cada envío de segmento, por lo que el timeout se adapta a los retardos en la red. El factor β se establece entre 1 y 2, de forma que se consiga un reenvío adecuado. (La especificación original recomienda el valor de 2).

Este mecanismo presenta un problema: ¿ que ocurre si un ACK llega demasiado tarde ?

Al reenviar el paquete y llegar el ACK del primero enviado, el RTT se actualiza al nuevo valor. Este será demasiado pequeño, y se producirán reenvíos inútiles, afectando a la fluidez de la comunicación.

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.3 Control del flujo de datos

La situación anterior se resuelve con el algoritmo de Karn.

Cuando se produce un reenvío, el valor del timeout se incrementa en función del último timeout calculado: nuevo_Timeout=γ*Timeout. γ toma el valor de 2 para evitar inestabilidades.

El timeout se volverá a calcular en función del RTT cuando se envíe un nuevo segmento que no haya sido reenviado.

Control de la congestión en TCP. RFC 2581

La congestión en una red es una situación de retardo elevado en el envío de información, debido a la sobrecarga de encaminamiento en los routers de una red.

Cuando en una red TCP/IP se produce una situación de congestión, TCP reacciona reenviando datos debido a la expiración de los timeouts. El reenvío genera más tráfico y por tanto más congestión, alcanzando la red un estado de bloqueo denominado colapso de congestión.

Para reducir la congestión, TCP debe reducir la tasa de envío de datos, es decir reducir su ventana de emisor.

TCP dispone de una serie de mecanismos para reducir su tasa de envío de datos cuando los retardos son elevados, descritos en el documento RFC 2581.

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.3 Control del flujo de datos

Prevención de la congestión por decremento multiplicativo

Esta técnica se fundamenta en la definición en el emisor de una nueva ventana denominada ventana de congestión, un valor en bytes al igual que la ventana del emisor.

En todo momento, la ventana del emisor se calcula como el valor mínimo de dos valores: la ventana de congestión y la ventana que informa el receptor.

TCP supone que la expiración del timeout de un segmento es debido a la congestión, y actualiza los siguientes valores: Con cada expiración de timeout para un segmento, reduce el tamaño de la ventana de congestión a la mitad, y multiplica por dos el timeout de los paquetes en la ventana del emisor. Esto provoca que conforme expiran temporizadores, el emisor envía cada vez menos datos.

Recuperación de una situación de congestión. Algoritmo de inicio lento.

Una vez que se evita la congestión y comienzan a llegar ACK's, el timeout vuelve a decrementarse y la ventana de congestión debería aumentar.

Sin embargo, si la recuperación es a la misma velocidad que la reducción del envío de datos, se puede producir un efecto "ola" de congestión periódica, la red queda oscilando entre congestión – no congestión.

Para evitar esto, la recuperación se realiza más lentamente. Para ello, el valor de la ventana de congestión se incrementa en un tamaño de MSS bytes, cada vez que el emisor recibe un ACK.

6.3 Protocolo de Control de la Transmisión (TCP)

6.3.3 Control del flujo de datos

El problema de los paquetes pequeños. Algoritmo de Nagle. RFC 896.

Si una aplicación envía a la capa TCP información en bloques de pocos bytes (Telnet envía un carácter (byte) al equipo remoto y espera un eco del carácter para enviar el siguiente), puede producirse una situación de desaprovechamiento del medio físico.

El emisor enviará bloques de un byte al receptor y esté hará ACK's de un byte. De esta forma el envío de información se ralentiza, sobre todo si el RTT es alto en la red.

Para evitar que el intercambio de datos sea byte a byte, el algoritmo de Nagle hace que TCP agrupe los bytes enviados por la aplicación en un segmento TCP. El primer byte será enviado y TCP almacena los bytes que lleguen del nivel superior hasta la llegada del ACK. A continuación enviará todos los bytes acumulados en otro segmento, y acumulará los siguientes hasta la llegada del ACK. Si se alcanza el tamaño del MSS en el buffer, TCP envía el segmento sin esperar al ACK.