élogos

COMUNICACIÓN DE APLICACIONES EN RED

élogos

ÍNDICE

COMUNICACIÓN DE APLICACIONES EN RED

1. Protocolo TCP y UDP

TCP

TCP es un Protocolo de flujo de bytes, fiable, orientado a conexión y con control de flujo.

Fiable: se encarga de comprobar que los datos llegan al destino. Si no, los vuelve a transmitir. Orientado a conexión: antes de enviar se establece conexión y hasta que no está establecida no se envían datos. (En UDP e IP no sucede así).

Flujo de Bytes: TCP maneja los datos como un stream. El stream se parte en trozos (paquetes) los envía (no necesariamente tienen que llegar en el mismo orden). El software TCP del receptor los recompone y los coloca en orden.

Posibilidad de recuperar los paquetes perdidos (IP no detecta paquetes perdidos).

Confirmación y control de flujo: TCP envía las confirmaciones al emisor.
Si no hay confirmación no sigue enviando los datos. Así se consigue el control de flujo, que no nos envíen más datos de los que somos capaces de procesar.

Entrega a la aplicación correcta: cuando llega un paquete mira las cabeceras para ver a qué aplicación va el paquete.

UDP

UDP es un Protocolo de trasporte de datagramas, no fiable sin conexión y sin control de flujo. La unidad de trasporte se llama datagrama y son independientes.

- No fiable: envía un paquete IP y si no llega se pierde.
- Sin conexión
- No control de flujo
- Se utiliza cuando queremos hacer una consulta rápida a una base de datos (no de gestión) por ejemplo un servidor de DNS.
- UDP es mucho más rápido.
- Se usa también en aplicaciones de audio o vídeo. Si algún datagrama se pierde se produce una interferencia

Los números de puertos

Los números de puerto se representan con 16 bits (2 bytes).

2 elevado a 16 = 65.535 puertos distintos

Nos van a permitir identificar una aplicación dentro de un Host. Junto con la dirección IP trasferimos el puerto. Se dividen en 2 grupos.

Well Known Ports (0..1023) Anonymous Ports (1024..65535)

Los números de puerto están estandarizados (IANA)

21 - FTP
80 - http
25 - SMTP
23 - TELNET
110 - POP3
53 - DNS

El cliente coge un puerto cualquiera que esté libre. El sistema nos da uno aleatorio. Tanto para TCP como UDP están estandarizados los mismos puertos.

¿Dónde aparecen publicados los puertos y servicios?

UNIX - /etc/services
WINDOWS - /Windows/services
En Unix los primeros 1024 puertos sólo puede usarlos el superusuario
No se permite al usuario instalar servicios en los puertos bajos (caballos de Troya).

Ejemplo

Enviar una petición a un Servidor DNS.
Cliente IP: 120.181.31.4
Servidor IP:80.41.12.3
Puerto: 53 (DNS)
Dir. IP Origen: 120.181.31.4
Dir. IP Destino: 80.41.12.3
Puerto Origen:1300
Puerto Destino:53
Pregunta:www.Microsoft.com
Dir. IP Origen: 120.181.31.4
Dir. IP Destino: 80.41.12.3
Puerto Origen:1300
Puerto Destino:53
Respuesta: 181.12.131.4

Tipo de servidores TCP/UDP

Hay 2 tipos de servidores:

Servidores concurrentes: Normal en TCP y
Servidores iterativos: Típico de UDP.

Iterativos

Típico de UDP (atiende peticiones sueltas).

Consiste en hacer un bucle que atiende peticiones de clientes. Coge la petición, la resuelve y manda la respuesta al cliente. Después atiende la siguiente petición.

Son fáciles de implementar.

Concurrentes
Normal en TCP

Cuando se conecta el cliente, creamos un nuevo hilo (thread) que se encarga de atender a ese cliente, cuando el cliente se desconecta el hilo muere.

El Protocolo TCP

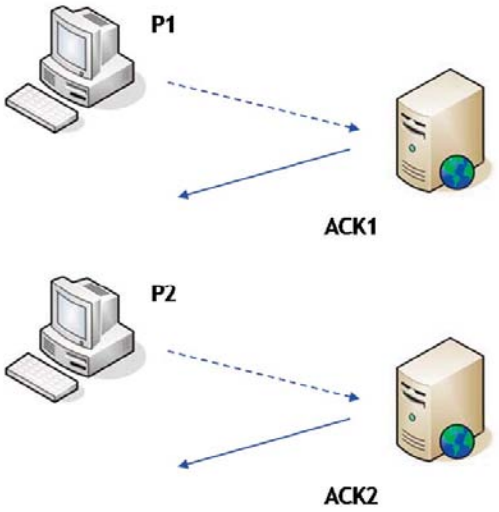
En TCP las conexiones siempre son full-duplex > ambos interlocutores pueden leer o escribir al mismo tiempo. La orden primeramente irá al buffer de TCP.
En TCP se envían y reciben bytes

TCP se encarga de la confirmación de que los datos hayan llegado. Para ello TCP utiliza un sistema de ACK (ACKnowledge).

Una vez que hemos recibido datos se envía un ACK (es un paquete adicional que se envía). No existe el concepto de NAK > Si llega mal o no llega aviso al cliente (emisor).
Si no llega confirmación de una trama, se volverá a enviar pasado un tiempo (t. máximo).

Sistema de parada y espera

El sistema de parada y espera es bastante lento, hay que esperar a que llegue el ACK.



Sistema de ventana deslizante

El sistema de parada deslizante.

Stall (Parada)

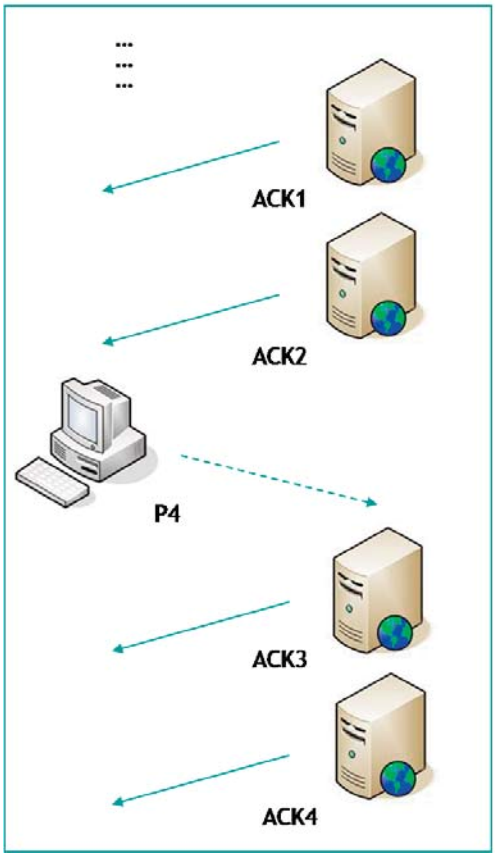
Podemos enviar varias tramas seguidas sin necesidad de tener que parar para cada una de ellas en espera de cada uno de sus ACK.

El problema es que podemos saturar al receptor. El receptor (TCP del receptor) irá cogiendo datos y se irá llenando el Buffer. Puede ser que no seamos capaces de procesar rápidamente los datos del Buffer y se saturará.

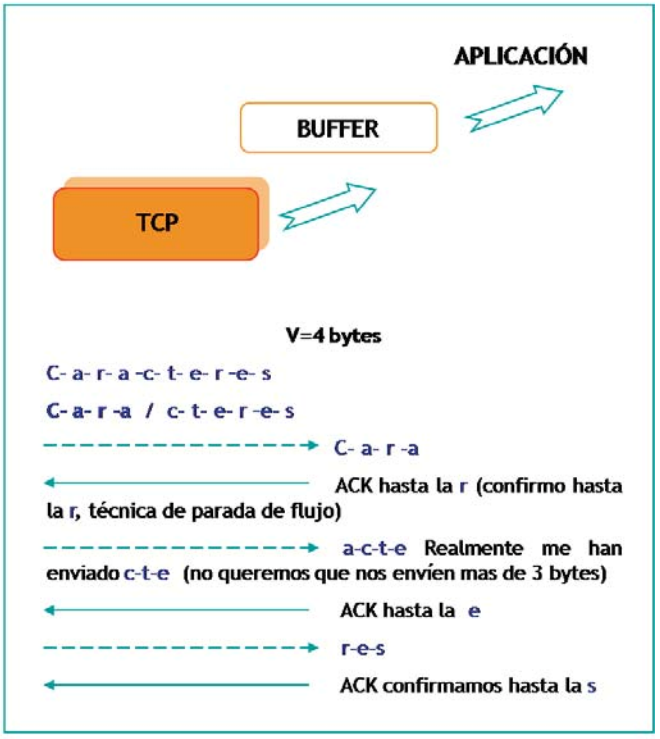
Ponemos un límite máximo de datos que podemos enviar al receptor sin que éste nos mande confirmación (ventana deslizante).

Se utiliza como una técnica de control de flujo. Si no le enviamos un ACK le estamos parando el envío de datos.

RECUADRO 1

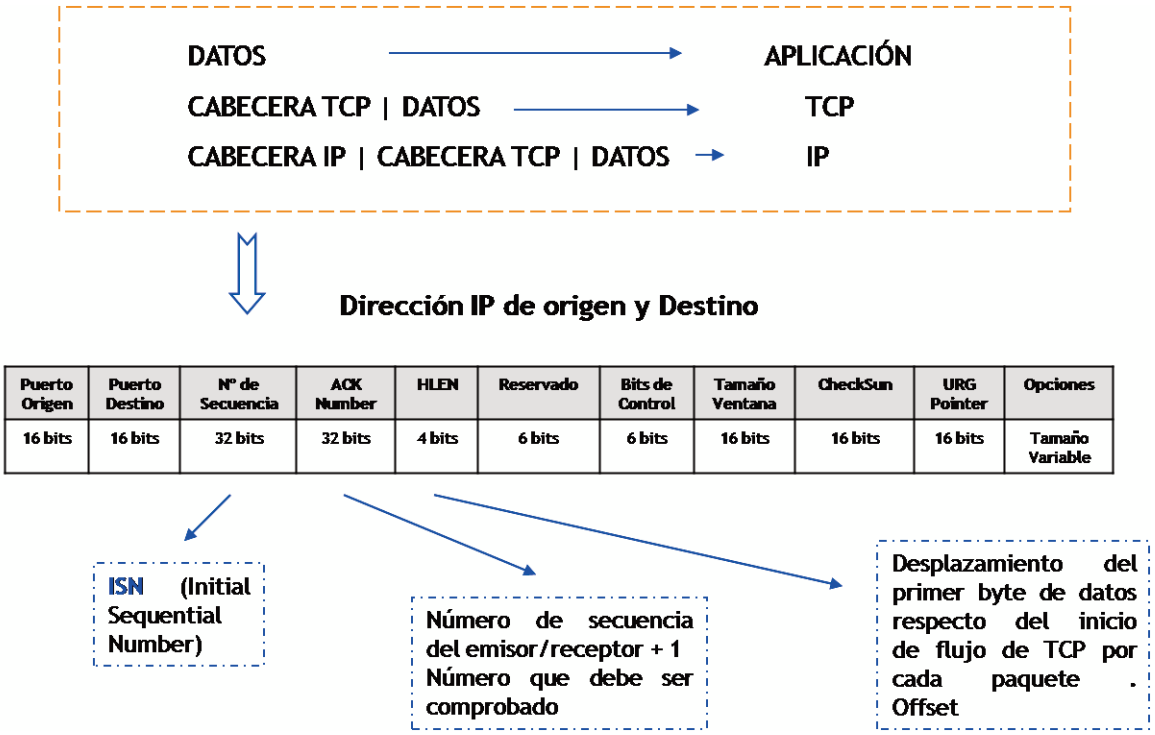


RECUADRO 2



Formato de la cabecera TCP

Podemos ver aquí el formato de la cabecera TCP



2. Establecimiento y cierre de la conexión TCP

Establecimiento de la conexión

Algoritmo Three Steps Handshake

El servidor elige un puerto y se lo pide a TCP, para instalarse.

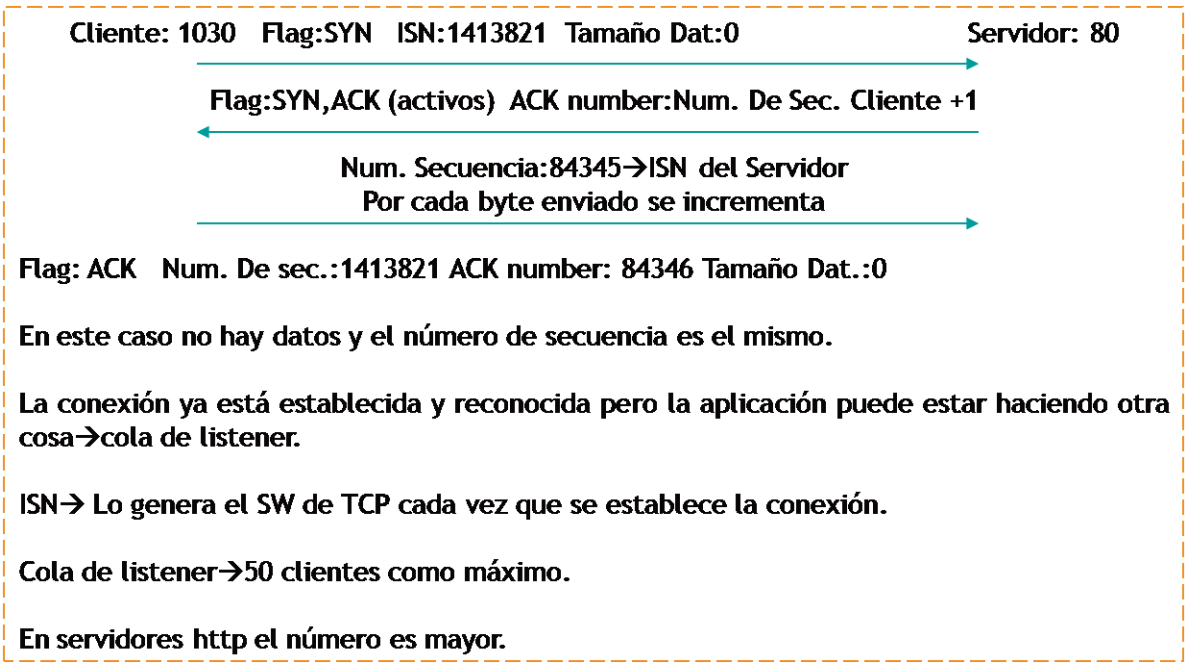
El hilo queda dormido esperando a que alguien se conecte > passive open

Después cuando el cliente se quiere conectar al servidor envía un segmento TCP formado exclusivamente por una cabecera TCP y dentro activa el flag de SYN y en el campo de número de secuencia el cliente pone un número a partir del cual quiere empezar a sincronizar los bytes de la conexión.

A ese número se le llama ISN (initial séquense number). No se empieza a contar desde 0.

Si abrimos una conexión, la cerramos y la volvemos a abrir rápidamente, en la segunda conexión pueden llegar segmentos de la primera conexión.

- Para evitar este efecto lateral, se lleva un contador interno que lleva el número de conexión para poder diferenciar a qué conexión pertenece el paquete.



Cierre de la conexión

Algoritmo Two Ways Handshake.

Durante el cierre se ve una doble conexión. Una de A > B y otra de B > A. Al cerrar no se distingue entre cliente y servidor.

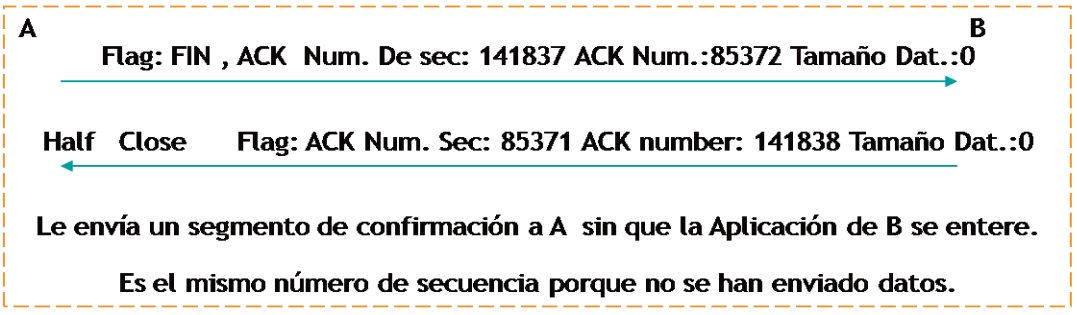
- Cuando cerramos de A - B, A ya no puede enviar datos a B pero B si puede enviar datos a A y ahí estamos en estado half close. Sí podemos enviar ACK's.
- Cuando se cierra de B - A ya no podemos enviar en ninguna dirección.

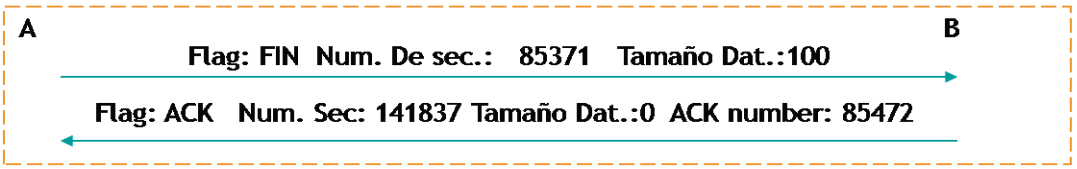
A cierra B

TCP envía los datos que tiene el Buffer y luego envía el flag de fin en el último segmento TCP o bien envía flag de fin en uno independiente.

El SW de TCP de B recibe este segmento e informa a la aplicación de que se va a cerrar la conexión, pero la forma de informar es que la aplicación pide datos a TCP no al revés pero la aplicación puede tardar en responder.

Más tarde la aplicación de B verá que no hay más datos que leer y pedirá a su SW de TCP cerrar la conexión. El SW de TCP enviará otro segmento con el flag de FIN y datos que quedasen en el buffer de B.





Timeout

Es el tiempo que ha de transcurrir para que un cliente tratando de establecer una conexión indique que dicha conexión no puede ser establecida.

1. No se especifica el motivo en las aplicaciones estándares.

2. Este timeout varía de unas implementaciones a otras.

3. En el caso del UNIX BSD y derivados, este timeout vale 75 segundos en los que se suelen mandar entre 3 y 5 paquetes de establecimiento.

4. Funciona en base a ticks de 500 mseg.

Segmentos de Reset

- Un segmento es de Reset cuando se activa en la cabecera TCP el flag RST.
- Se activa el bit de Reset en una conexión TCP cuando el paquete que ha llegado no parece, en principio, estar relacionado con la conexión a la que está referido el paquete.
- Las causas de generar un paquete con este bit para una conexión TCP pueden ser varias.

a) Intento de conexión a un puerto no existente.
b) Abortar una conexión.
c) Respuesta ante conexiones semi-abiertas

La mayoría de los servidores TCP son concurrentes:

- Un servidor recibe una nueva petición de conexión.
- El servidor acepta la conexión e invoca a un nuevo proceso (fork o threads) para gestionar esa conexión.
- Estado de las conexiones TCP: netstat –n
- Demultiplexación en base a dir. IP de destino, puerto de destino, dir. IP de origen y puerto de origen: Sólo el proceso en estado LISTEN recibirá los segmentos SYN.

  Sólo los procesos en ESTABLISHED reciben datos (nunca el del estado LISTEN).
- Cola de entrada de peticiones de conexión:

  Todo punto final de conexión (socket) en escucha tiene una cola de longitud fija de conexiones que han sido aceptadas por TCP y no por la aplicación.

  La aplicación especifica un límite (backlog) a esta cola (entre 0 y 5).

Cuando llega un segmento SYN pidiendo una nueva conexión, TCP aplica un algoritmo para decidir en función del backlog y del nº de conexiones encoladas si acepta o no la nueva.

Si se puede aceptar la nueva conexión _ TCP lo hace.

- La aplicación del servidor no "ve" la nueva conexión hasta que no se recibe el tercer segmento del establecimiento de la conexión.
- Si el cliente comenzara a enviar datos antes de que esta notificación se produjera, estos datos se encolarían en un buffer de entrada.

Si no hay sitio en la cola para la nueva conexión, se rechaza sin enviar nada de vuelta al cliente. En el cliente se producirá un timeout