# UDP

User datagram protocol. UDP es un protocolo de nivel de transporte, orientado a datagramas, y simple. Cada bloque de datos generado por la capa de aplicación produce un único datagrama UDP. UDP no garantiza que el datagrama alcance su destino. UDP multiplexa los datos de las aplicaciones y efectúa una comporbación de errores, pero no realiza control de flujo, control de congestión, retransmisión de datos perdidos, conexiones ni desconexiones. Se utiliza principalmente en los siguientes casos:

- Cuando el medio de transmisión es altamente fiable y sin congestión (LANs).
- Cuando la aplicación es en tiempo real y no se pueden esperar los ACKs.
- Cuando los mensajes se producen regularmente y no importa si se pierde alguno.
- Si se envía tráfico de broadcast o multicast.

## Cabecera UDP

16 primeros bits: nº de puerto de origen (longitud UDP) 16 siguientes bits: nº de puerto destino, checksum UDP Los numeros de puerto identifican los procesos emisor y receptor, los números de puerto UDP son independientes de los de TCP. Longitud UDP = longitud de la cabecera UDP + longitud de datos. El valor mínimo es de 8 bytes. Es redundante la infromación de la cabecera IP. Checksum: se calcula sobre la cabecera UDP y los datos UDP. Antes era opcional, ahora es obligatorio por defecto.

# TCP

TCP o Transmission Control Protocol es un protocolo que proporciona un servicio de envío de datagramas fiable y orientado a conexión. Al ser orientado a conexión, dos aplicaciones, generalmente en un modelo cliente-servidor, deben establecer una conexión TCP entre ellas antes de iniciar el intercambio de datos. Su fiabilidad garantiza que los datos se reciben correctamente y en el orden adecuado. Cabe destacar que TCP no admite broadcasting ni multicasting. Los paquetes en TCP se denominan segmentos y la comunicación que permite es full-duplex, es decir, bidireccional y simultánea. Entre sus principales funciones se encuentran el establecimiento y la terminación de conexiones, la gestión de buffers y el control de flujo eficiente, la multiplexación del nivel de aplicación mediante puertos, el intercambio de datos con las aplicaciones, el control de errores mediante la retransmisión de segmentos perdidos o erróneos y la eliminación de duplicados, así como el control de congestión en la red. Para implementar la fiabilidad TCP implementa lo siguiente:

- Divide datos de la aplicación en segmentos con la longitud más adecuada para la aplicación.
- Asocia un temporizador con los segmentos que envía. Si no recibe el ACK del destino a tiempo remite el segmento.
- Mantiene un checksum en la cabecera TCP para comprobar el segmento recibido. No se envía ACK si el segmento es incorrecto.
- El receptor TCP reordena los segmentos, si es necesario, para pasarlos ordenados a la aplicación (los segmentos se pueden desordenar un la transmisión).
- Descarta segmentos que se hayan podido duplicar.
- Proporciona control de flujo: un receptor TCP sólo deja transmitir al otro extremo segmentos que pueden almacenarse en su bugger de entrada, sin producirse desbordamientos.

# Cabecera TCP

**Nº de puerto origen y destino** + dir. IP origen y destino de cabecera IP identifican unívocamente la conexión TCP. **Número de secuencia**: identifica ek nº de byte en el flujo de bytes TCP entre el emisor y el receptor que supone el primer byte de la sección de datos:

- Cuando se llega a $2^{32}$-1 se comienza de nuevo por 0.
- Cuando se establece una conexión, se pone a 1 el flag SYN, y la máquina selecciona un Initial Sequence Number, ISN, para esa conexión. **Número de ACK**: indica el siguiente número de secuencia que el emisor del ACK espera recibir.
- Es el nº de secuencia + 1 del último byte recibido satisfactoriamente.
- TCP proporciona una comunicación "full-duplex" al nivel de aplicación, cada extremo mantiene su nº de secuencia.
- No existen ACK's negativos, pero sí selectivos, SACK. **Longitud de cabecera**: tamaño de la cabecera incluyendo opciones.
- Especifica el número de palabras de 32 bits.
- Valor máximo 60 bytes. **Flags**
- Congestion Window Reduced, CWR: el emisor reduce su velocidad de transmisión.
- ECE: el emisor confirma la recepción de un paquete con el flag Explicit Congestion Notification, ECN, activado.
- URG: puntero de urgencia válido.
- ACK: número de ACK válido, siempre activado una vez establecida la conexión.
- PSH: el receptor debe pasar estos datos a la aplicación lo antes posible, es una implementación poco fiable y poco usada.
- RST: reinicia la conexión.
- SYN: sincronizar números de secuencia para iniciar una conexión.
- FIN: el emixor finaliza el envío de datos. **Tamaño de ventana**: indica el nº de bytes, comenzando por el valor del campo de nº de acknowledge, que el receptor puede aceptar.
- Utilizado para establecer control de flujo.
- Máximo de 65.535, pero existe una opción de factor de escala para incrementar ese valor. **Checksum**: sobre todo el segmento TCP
- Es obligatorio, debe calcularlo el emisor y comprobarlo el receptor.
- El cálculo es similar al checksum de UDP. **Puntero de urgencia**: válido si el flag URG es 1.
- Indica un offset a añadir al nº de secuencia.
- Se utiliza para transmitir datos urgentes. **Opciones**: la más común es la opción de máximo tamaño de segmento (Maximum Segment Size). **Datos**: información enviada (opcional).

## TCP: Establecimiento

Las conexiones las inicia, normalmente, el cliente. El servidor hace una apertura pasiva. Protocolo de establecimiento de conexión (**Three-Way Handshake**):

- El emisor envía un segmento SYN indicando el nº de secuencia inicial.
- El servidor responde con su propio segmento SYN que contiene el nº de secuencia inicial del servidor. También confirma (ACK) el SYN del cliente + 1 (los mensajes SYN consumen un nº de secuencia).
- El cliente confirma el SYN del servidor con un nº de ACK igual al ISN del servidor + 1. **Número de secuencia inicial, ISN**:
- Cada extremo sececciona su ISN al establecerse la conexión.

- Se obtiene pseudoaleatoriamente.
- Objeto: evitar que segmentos "antiguos", de otra conexión igual, se confundan con los actuales.

## TCP: finalización

Se intercambian 4 segmentos para cerrar una conexión.

- Una conexión TCP es full-duplex y cada dirección se cierra independientemente.
- Cada extremo envía un FIN cuando ha realizado el envío de datos. El otro extremo puede continuar enviando datos. El extremo que envía el primer FIN realiza el cierre activo, y el otro extremo el cierre pasivo, cualquiera de los dos extremos puede empezar el cierre. Protocolo de finalización de conexión:
- El cliente finaliza la aplicación, el cliente TCP envía un FIN con el número de secuencia correspondiente.
- El servidor responde con un ACK del nº de secuencia + 1.
- A continución, el servidor envía un FIN.
- El cliente confirma la recepción del FIN, con un ACK del nº de secuencia recibido + 1.

## TCP: MSS

Maximum Transmission Unit: número máximo de bytes de datos que puede enviar el nivel de enlace.
Maximum Segment Size: indica el número máximo de bytes de datos que le conviene recibir a cada extremo, para evitar la fragmentación IP. Cuando se establece una conexión TCP, cada extremo anuncia el MSS que espera recibir:

- La opción MSS sólo aparece en un segmento SYN.
- Si no se declara, toma un valor por defecto.
- MSS no incluye las longitudes de cabecera IP y TCP. En general es preferible un MSS grande que amortice el coste de cabeceras. Pero también interesa evitar la fragmentación. No se realiza una negociación del MSS, el tamaño de segmento será el menor de los dos.

## TCP: estados

Estado **TIME_WAIT**:

- TCP espera 2 veces el tiempo máximo de vida de un paquete en la red, por si se ha perdido el último ACK.
- Variable /proc/sys/net/ipv4/tcp_fin_timeout (segundos)
- Permite a TCP reenviar el ACK en caso de que se haya perdido, el otro extremo reenviará el FIN
- Mientras la conexión está en este estado, no se puede reutilizar el par de sockets de esa conexión, cualquier segmento retrasado recibido es descartado para garantizar que no aparecen reencarnaciones de segmentos en futuras conexiones. Estado **FIN_WAIT_2**:
- Permanecerá en este estado hasta recibir el FIN del otro estremo.
- El otro extremo está en el estado CLOSE_WAIT y debe esperar a que se cierre la aplicación.
- Para evitar una espera infinita, las implementaciones establecen un tiempo de espera, tras el cual pasa directamente al estado CLOSED.

## TCP: Segmentos de Reset

Un segmento 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 relacioando con la conexión a la que está referido el paquete. Las causas de generar un paquete con ese bit para una conexión TCP pueden ser varias.

- Intento de conexión a un puerto no existente.
- Respuesta ante conexiones semi-abiertas.