



Bloque III: El nivel de transporte

Tema 5: UDP y TCP



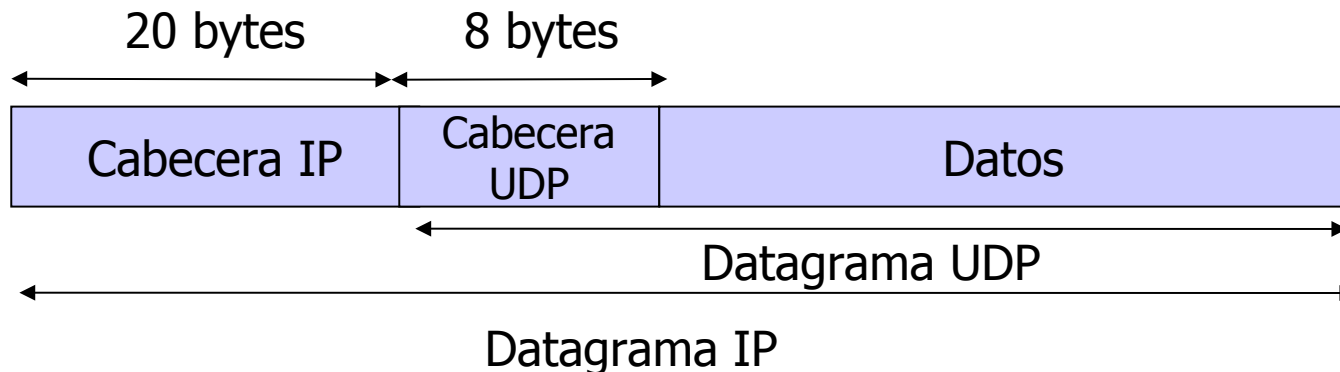
Índice

- Bloque III: El nivel de transporte
 - Tema 5: UDP y TCP
 - UDP
 - Cabecera UDP
 - TCP
 - Cabecera TCP
 - Conexiones TCP
 - Estados TCP
 - Segmentos de Reset
- **Lecturas recomendadas:**
 - Capítulo 3, secciones 3.1, 3.2, 3.3, 3.5.1, 3.5.2 y 3.5.6, de “Redes de Computadores: Un enfoque descendente”. James F. Kurose, Keith W. Ross. Addison Wesley.
 - Capítulo 13 de “TCP/IP Illustrated, Volume 1: The Protocols”, W. Richard Stevens, Addison Wesley.



UDP

- User Datagram Protocol – Especificado en el RFC 768.
- 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 comprobación de errores, pero no realiza:
 - Control de flujo
 - Control de congestión
 - Retransmisión de datos perdidos
 - Conexión/desconexión





UDP

- Se utiliza principalmente en los siguientes casos:
 - Cuando el medio de transmisión es altamente fiable y sin congestion (LANs). Por ejemplo: DNS, NFS.
 - Cuando la aplicación es en tiempo real y no se pueden esperar los ACKs. Por ejemplo, videoconferencia, voz sobre IP.
 - Cuando los mensajes se producen regularmente y no importa si se pierde alguno. Por ejemplo: NTP, SNMP.
 - Si se envía tráfico broadcast o multicast.



Cabecera UDP

0	16	31
Nº de puerto origen		Nº de puerto destino
Longitud UDP		Checksum UDP

- Los números 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 con la información de la cabecera IP.
- Checksum: se calcula sobre la cabecera UDP y los datos UDP.
 - Antes era opcional. Ahora es obligatorio por defecto (RFC 1122).



TCP

- Transmission Control Protocol – Especificado en los RFC 793, 1122, 1323, 2018 y 2581.
- TCP proporciona un servicio de envío de datagramas fiable y orientado a conexión:
 - Orientado a conexión: dos aplicaciones (cliente-servidor) deben establecer una conexión TCP entre ellos antes de comenzar el intercambio de datos.
 - Fiable: los datos se reciben correctamente y en orden.
 - Broadcasting y multicasting no son aplicables.
- Los paquetes TCP se denominan **segmentos**.
- TCP es full-duplex: la comunicación es bidireccional y simultánea.
- Funciones de TCP:
 - Establecer y terminar conexiones.
 - Gestionar los buffers y ejercer control de flujo de forma eficiente.
 - Multiplexar el nivel de aplicación (puertos) e intercambiar datos con las aplicaciones.
 - Controlar errores, retransmitir segmentos perdidos o erróneos y eliminar duplicados.
 - Efectuar control de congestión

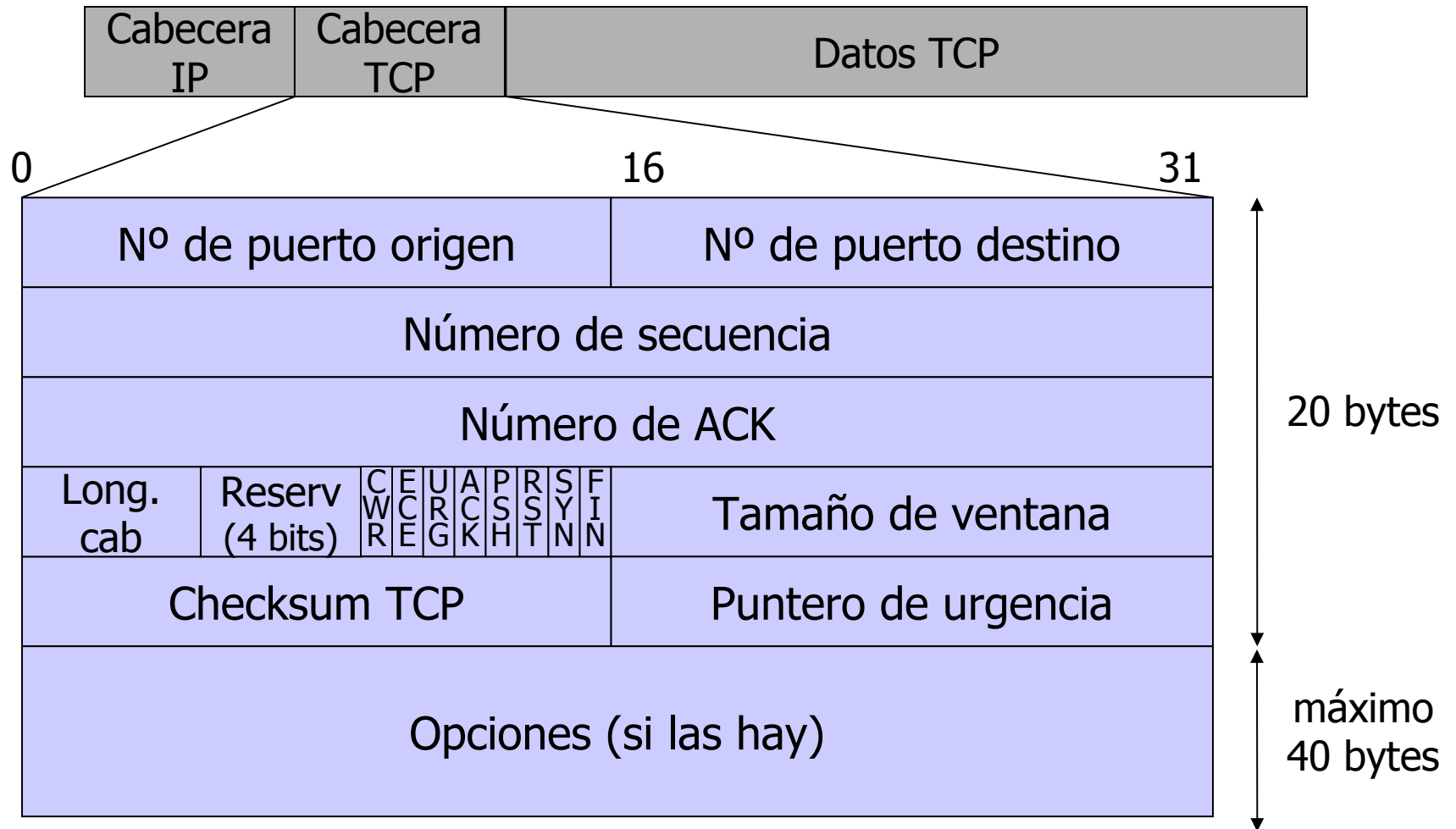


TCP

- 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 → retransmite 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 en 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 buffer de entrada, sin producirse desbordamientos.



Cabecera TCP





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 el 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 ISN (Initial Sequence Number) para esa conexión.
- **Número de ACK** (acknowledgment): 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 – opción de TCP).
- **Longitud de cabecera** (4 bits): tamaño de la cabecera incluyendo opciones.
 - Especifica el número de palabras de 32 bits
 - Valor máximo 60 bytes (15x4)



Cabecera TCP

- **Flags:**
 - CWR (Congestion Window Reduced): el emisor reduce su velocidad de transmisión.
 - ECE (ECN Echo): el emisor confirma la recepción de un paquete con el flag ECN (Explicit Congestion Notification – Cabecera IP) activado.
 - URG: puntero de urgencia válido (poco usado).
 - 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 (implementación poco fiable y poco usada).
 - RST: reinicializar la conexión (reset).
 - SYN: sincronizar números de secuencia para iniciar una conexión.
 - FIN: el emisor finaliza el envío de datos.



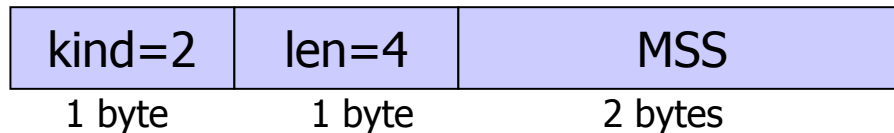
Cabecera TCP

- **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 65.535, pero existe una opción de factor de escala para incrementar este valor.
- **Checksum:** sobre todo el segmento TCP (cabecera + datos).
 - 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)

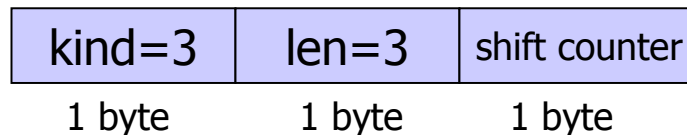


Cabecera TCP: Opciones

- Maximum Segment Size (RFC 793): tamaño máximo de datos que puede enviar un extremo de la conexión.



- Window Scale Factor (RFC 1323): permite ampliar el tamaño del buffer hasta casi 1 GB.
 - Cuenta desplazamientos (máx. 14).



- Se puede desactivar con:
echo 0 /proc/sys/net/ipv4/tcp_window_scaling



Conexiones TCP: Establecimiento

- Las conexiones las inicia, normalmente, el cliente (**apertura activa**) → El servidor hace una **apertura pasiva**.
- Protocolo de establecimiento de conexión (**Three-Way Handshake**):
 - El emisor (cliente) envía un segmento SYN indicando el nº de secuencia inicial (segmento 1).
 - El servidor responde con su propio segmento SYN que contiene el nº de secuencia inicial del servidor (segmento 2). 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 (segmento 3).
- **Número de secuencia inicial (ISN):**
 - Cada extremo selecciona su ISN al establecerse la conexión.
 - Se obtiene (pseudo)aleatoriamente.
 - Objetivo: evitar que segmentos “antiguos” (de otra conexión igual) se confundan con los actuales → Reencarnación.



Conexiones TCP: Finalización

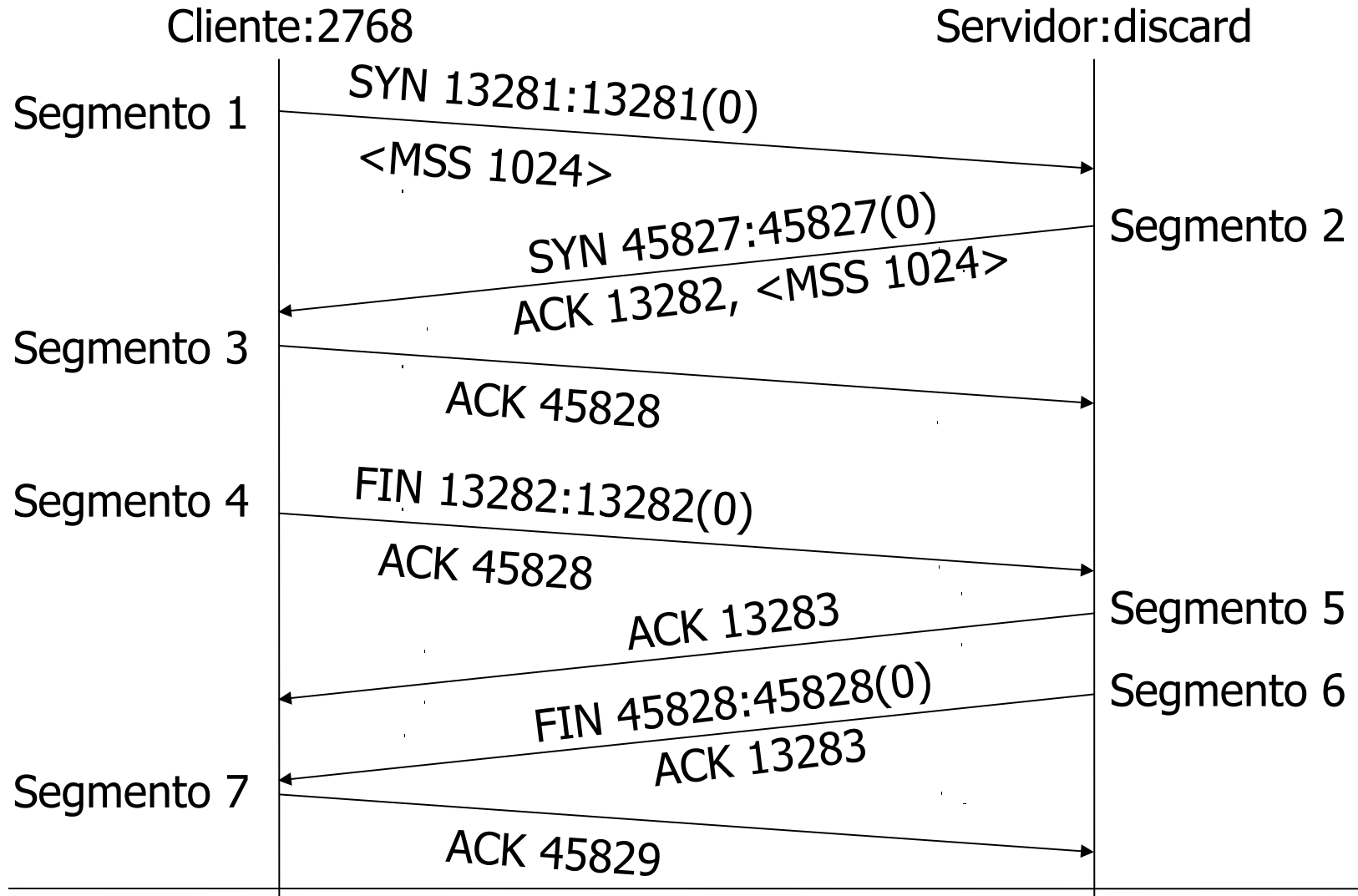
- Se intercambian 4 segmentos para cerrar una conexión.
 - Una conexión TCP es full-duplex y cada dirección se cierra independientemente.
 - Cada extremo envía un FIN cuando ha finalizado el envío de datos → El otro extremo puede continuar enviando datos (half-close).
- El extremo que envía el primer FIN realiza el cierre activo, y el otro extremo el cierre pasivo.
 - Cualquiera de los dos extremos puede iniciar el cierre.
- Protocolo de finalización de conexión:
 - El cliente finaliza la aplicación → El cliente TCP envía un FIN (segmento 4) con el número de secuencia correspondiente (cierre del flujo de datos cliente a servidor).
 - El servidor responde con un ACK (segmento 5) del n° de secuencia + 1 (los mensajes FIN consumen un n° de secuencia).
 - A continuación, el servidor envía un FIN (segmento 6).
 - El cliente confirma la recepción del FIN, con un ACK del n° de secuencia recibido + 1 (segmento 7).



¿Qué segmento puedo eliminar de la finalización sin que afecte?



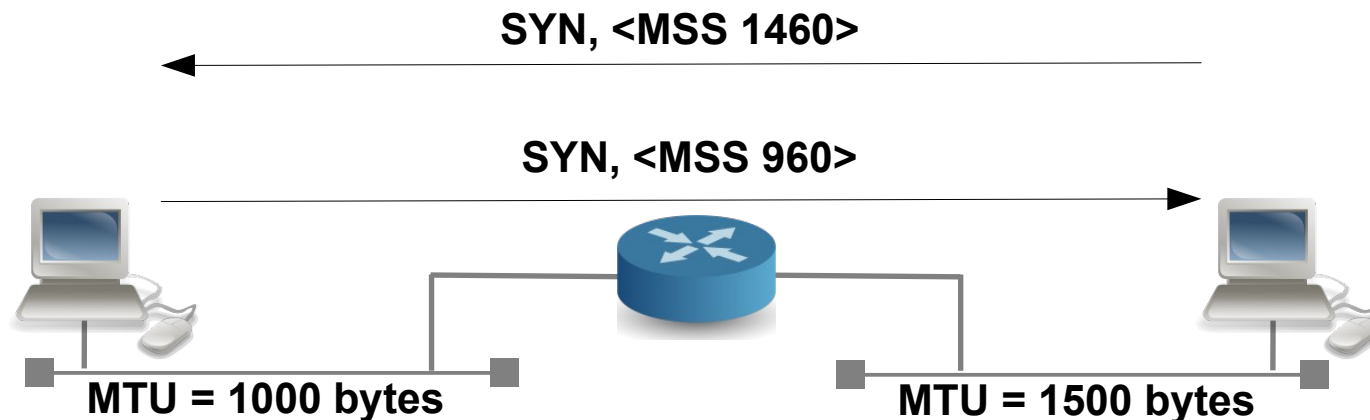
Conexiones TCP





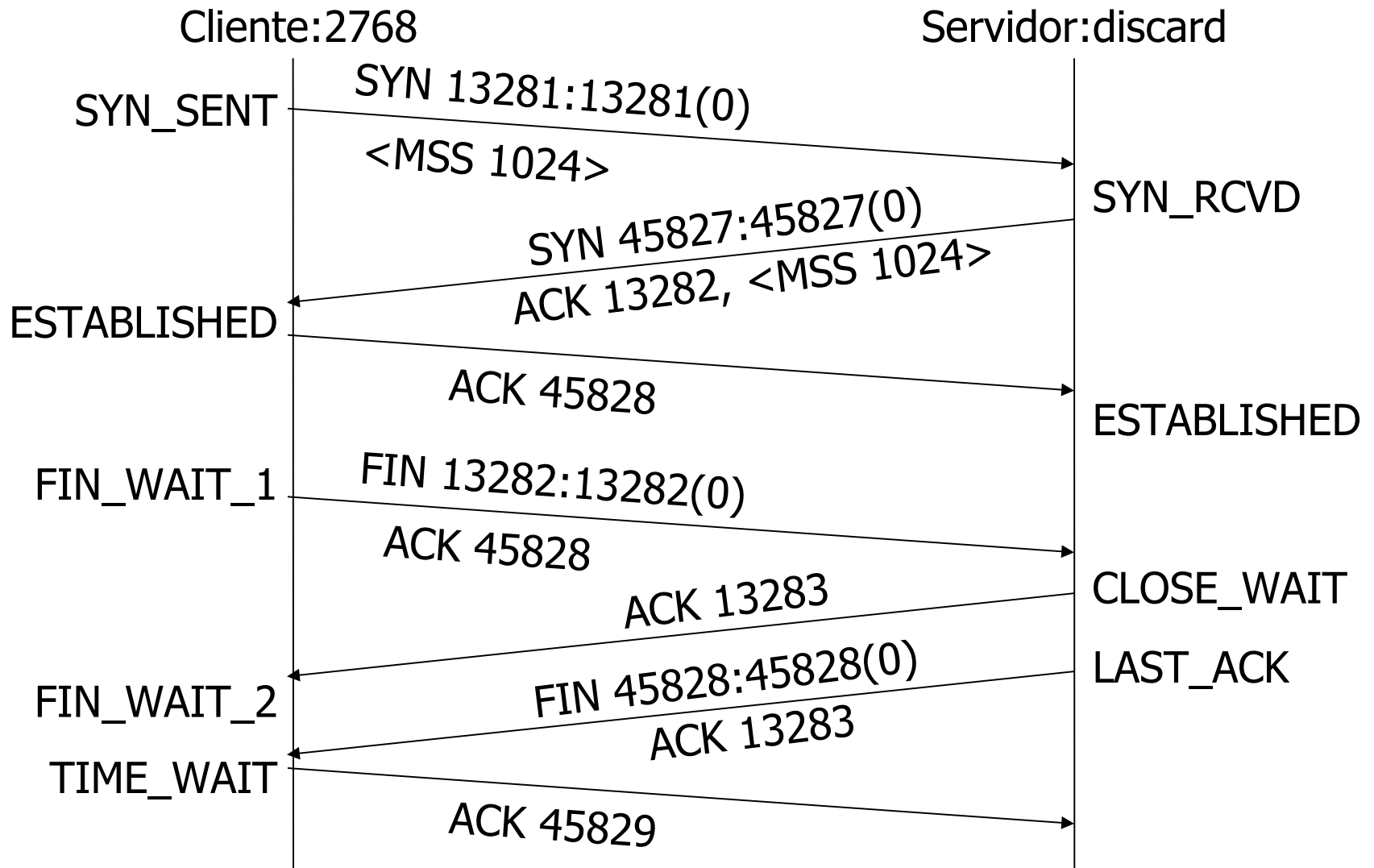
Conexiones TCP: MSS

- **Maximum Transmission Unit (MTU):** número máximo de bytes de datos que puede enviar el nivel de enlace.
- **Maximum Segment Size (MSS):** 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, se toma por defecto el valor 536 (datagrama IP de 576 bytes).
 - MSS no incluye las longitudes de cabecera IP y TCP ($MSS = MTU - 20 - 20$).
- 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.





Conexiones TCP: Estados





Conexiones TCP: Estados

- Estado **TIME_WAIT**:
 - TCP espera 2 veces el tiempo máximo de vida de un paquete en la red (Maximum Segment Lifetime – MSL), 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 pueden reutilizar el par de sockets de esa conexión → Cualquier segmento retrasado recibido es descartado → Garantiza que no aparecen reencarnaciones de segmentos en futuras conexiones.
- Estado **FIN_WAIT_2**:
 - Permanecerá en este estado hasta recibir el FIN del otro extremo.
 - 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 (misma variable que antes), tras el cual pasa directamente al estado `CLOSED`.



TCP: 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. Por ejemplo:
 - Intento de conexión a un puerto no existente
 - Respuesta ante conexiones semi-abiertas



TCP: Segmentos de Reset

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

