Bloque III: El nivel de transporte

Tema 6: Conexiones TCP

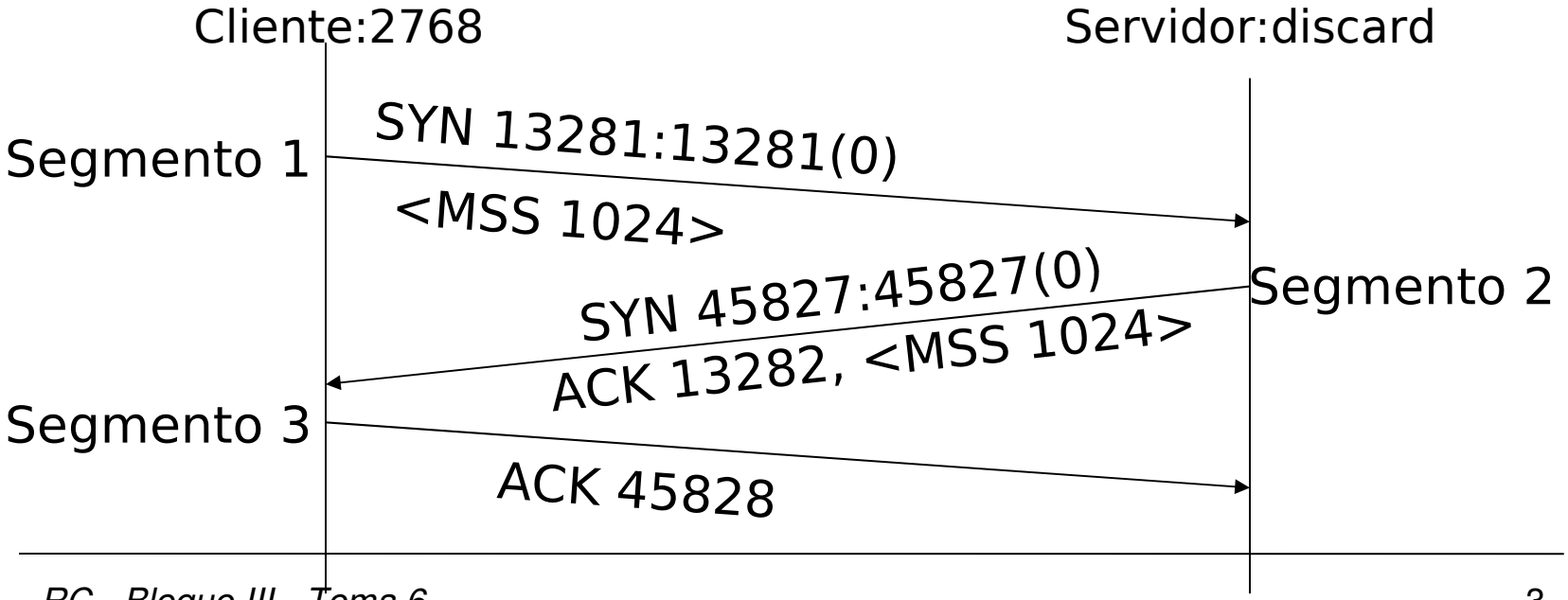
Índice

- Bloque III: El nivel de transporte
  - Tema 6: Conexiones TCP
    - Establecimiento de conexión
    - Finalización de conexión
    - Diagrama de estados
    - Segmentos de reset
    - Establecimiento y cierre simultáneos

- **Referencias**
  - Capítulo 3 de “Redes de Computadores: Un enfoque descendente basdado en Internet”. James F. Kurose, Keith W. Ross. Addison Wesley, 2ª edición. 2003.
  - Capítulo 18 de “TCP/IP Illustrated, Volume 1: The Protocols”, W. Richard Stevens, Addison Wesley, 1994.

Conexión TCP: Establecimiento

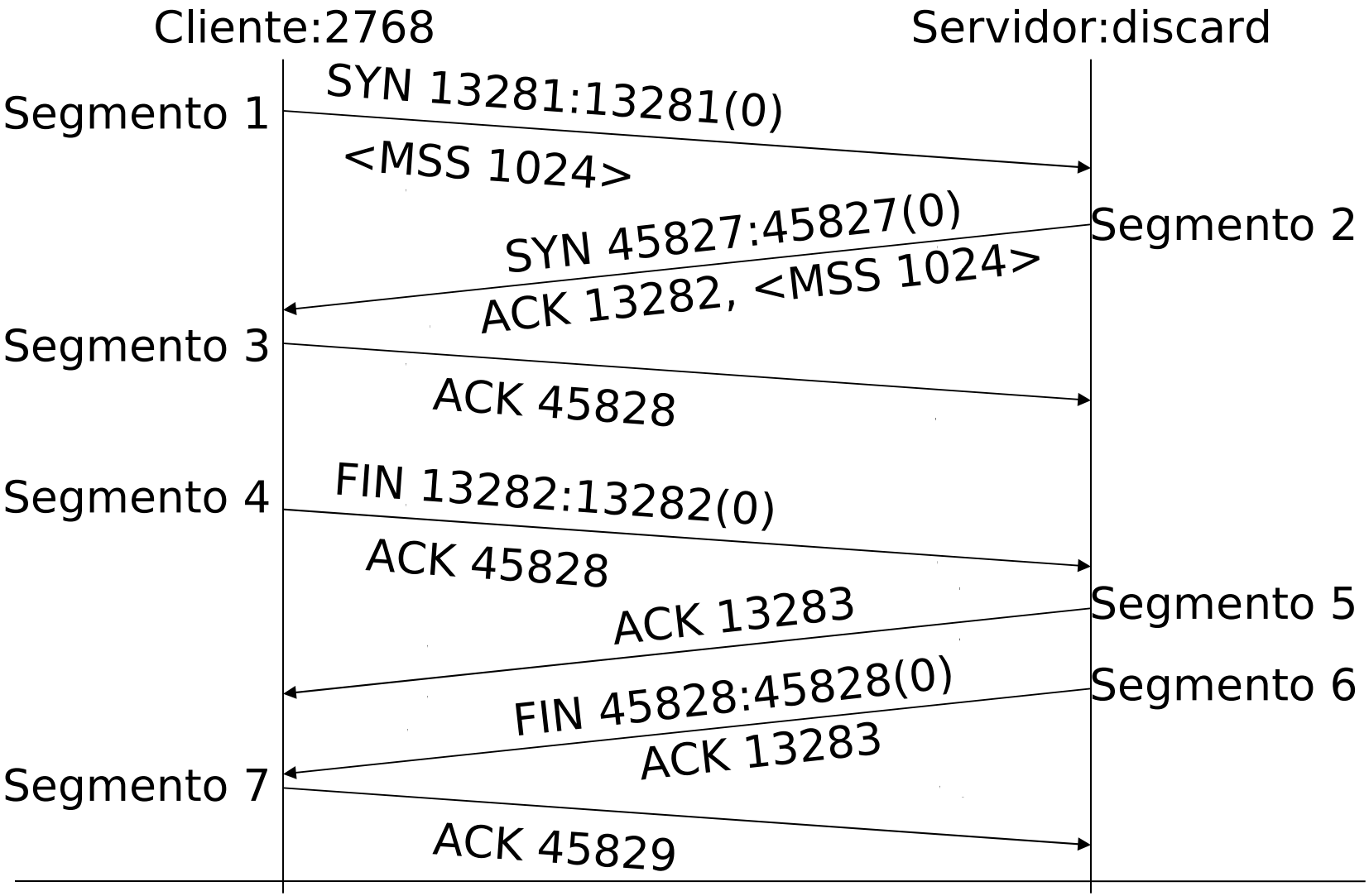
- Las conexiones las inicia, normalmente, el cliente (apertura activa) ➔ El servidor hace una apertura pasiva.
- Protocolo de establecimiento de conexión:
  - El emisor (cliente) envía un segmento SYN especificando el nº de puerto del servidor al que quiere conectarse y el nº de secuencia inicial (segmento 1).
  - El servidor responde con su propio segmento SYN que contiene el nº de secuencia inicial del servidor (segmento 2). También asiente (ACK) el SYN del cliente + 1 (los mensajes SYN consume un nº de secuencia).
  - El cliente asiente el SYN del servidor con un nº de ack igual al ISN del servidor +1 (segmento 3).



Conexión TCP: Finalización

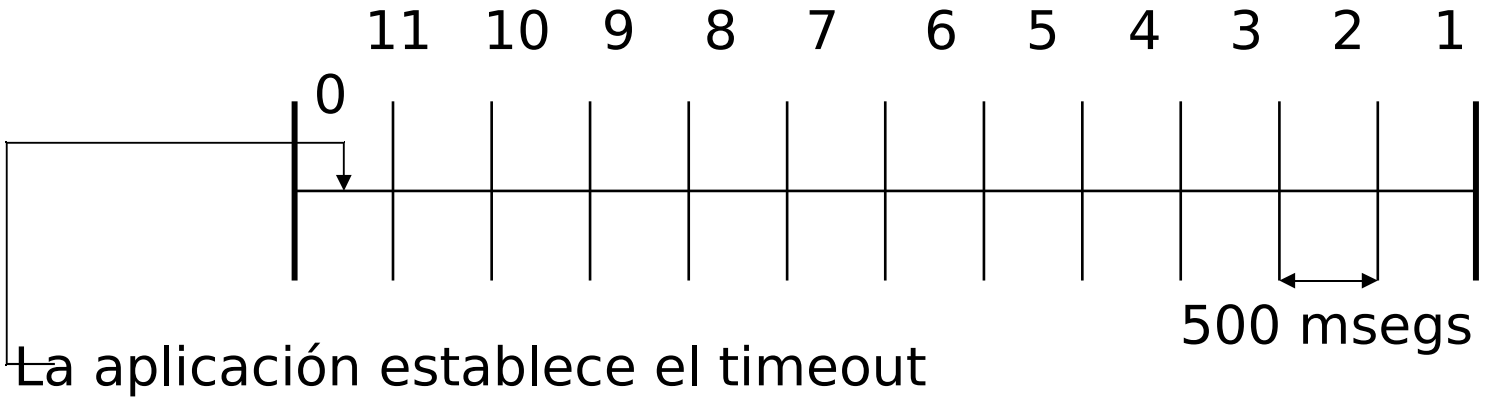
- Se intercambian 4 segmentos para cerrar una conexión.
  - Una conexión TCP es full-duplex y cada dirección se cierra independientemente.
  - Cada extremo envía un FIN cuando a finalizado el envío de datos ➔ El otro extremo puede continuar enviando datos (half-close).
- El extremo que envía el primer 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 nº de secuencia + 1 (los mensajes FIN consumen un nº de secuencia).
  - A continuación, el servidor envía un FIN (segmento 6).
  - El cliente confirma la recepción del FIN, con un ACK del nº de secuencia recibido + 1 (segmento 7).

Conexión TCP: Finalización



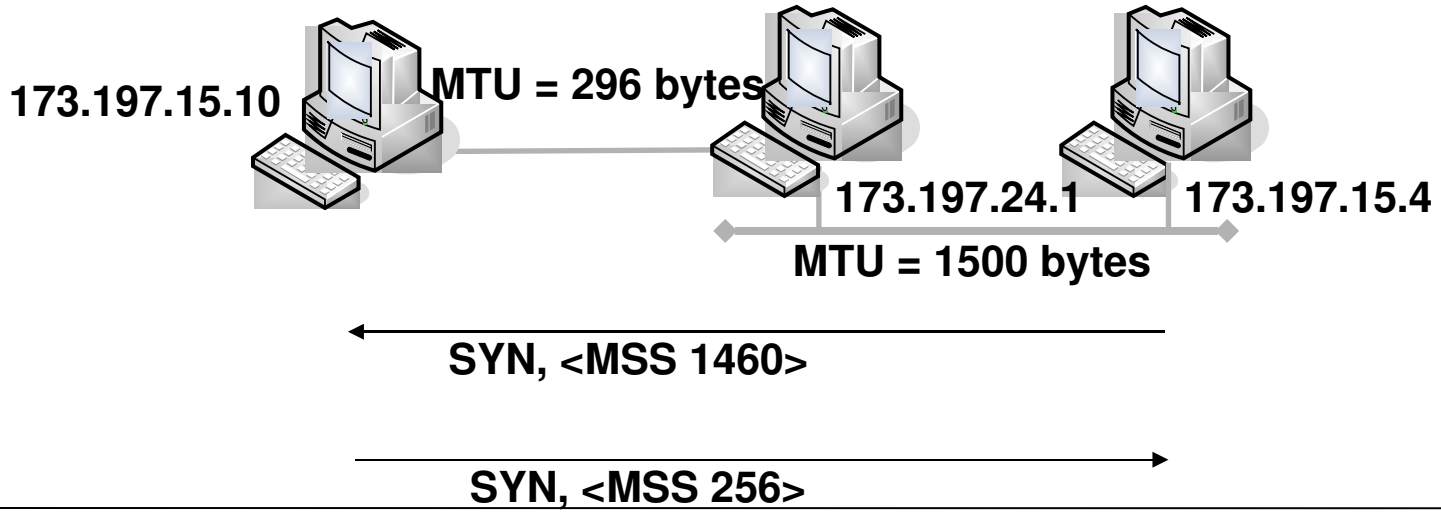
Conexión TCP: Timeout

- Tiempo que ha de transcurrir para que un cliente tratando de establecer una conexión indique que dicha conexión no puede ser establecida.
  - No se especifica el motivo en la aplicaciones estándares.
- Este timeout varía de unas implementaciones a otras.
- 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.
- Funciona en base a ticks de 500 mseg.



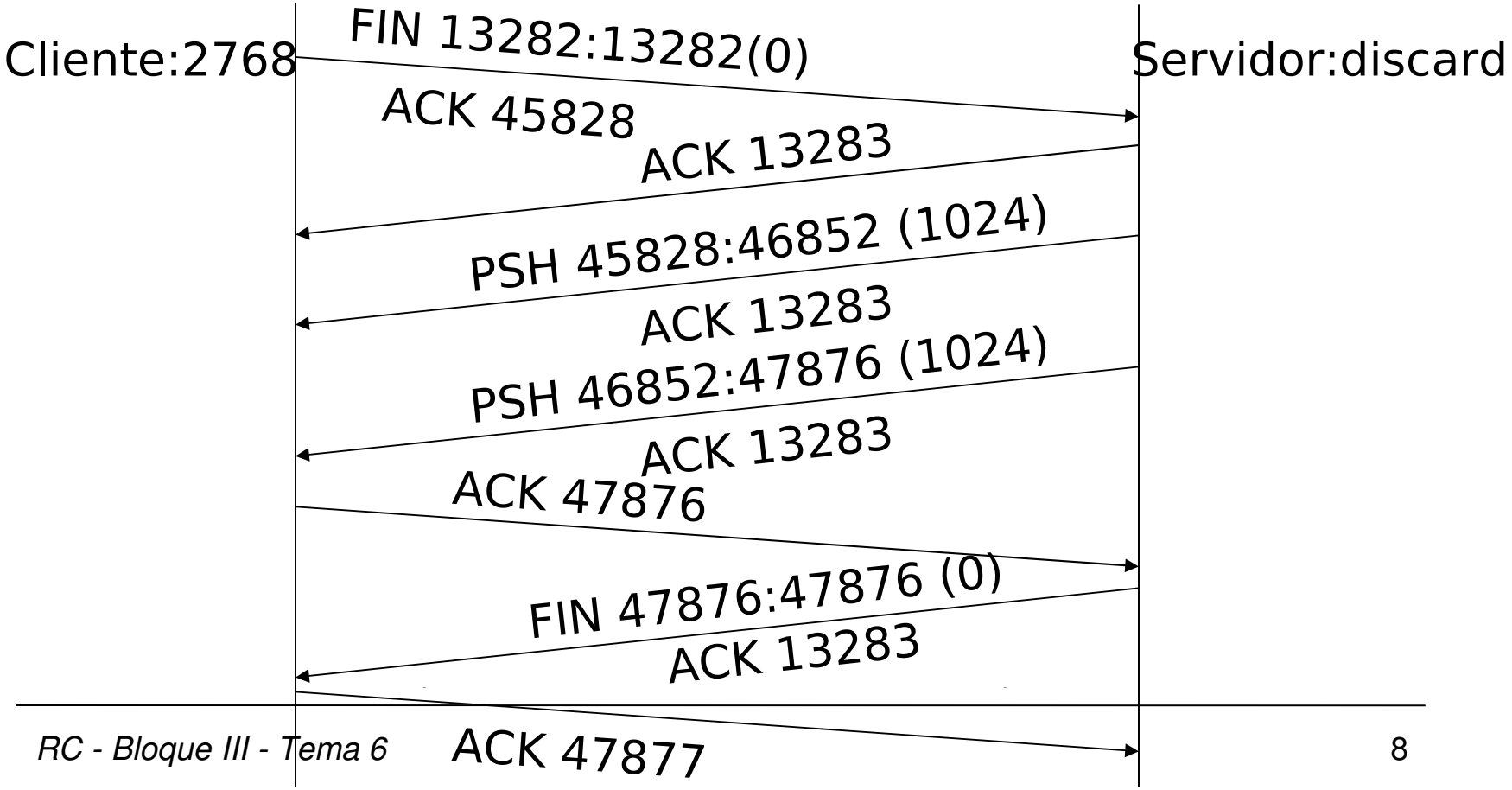
Conexión TCP: MSS

- 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 (MTU – 20 – 20 = MSS)
- En general es preferible un MSS grande que amortice el coste de cabeceras. Pero, obviamente, interesa evitar fragmentación.
- No se realiza una negociación del MSS, el tamaño de segmento será el menor de los dos.

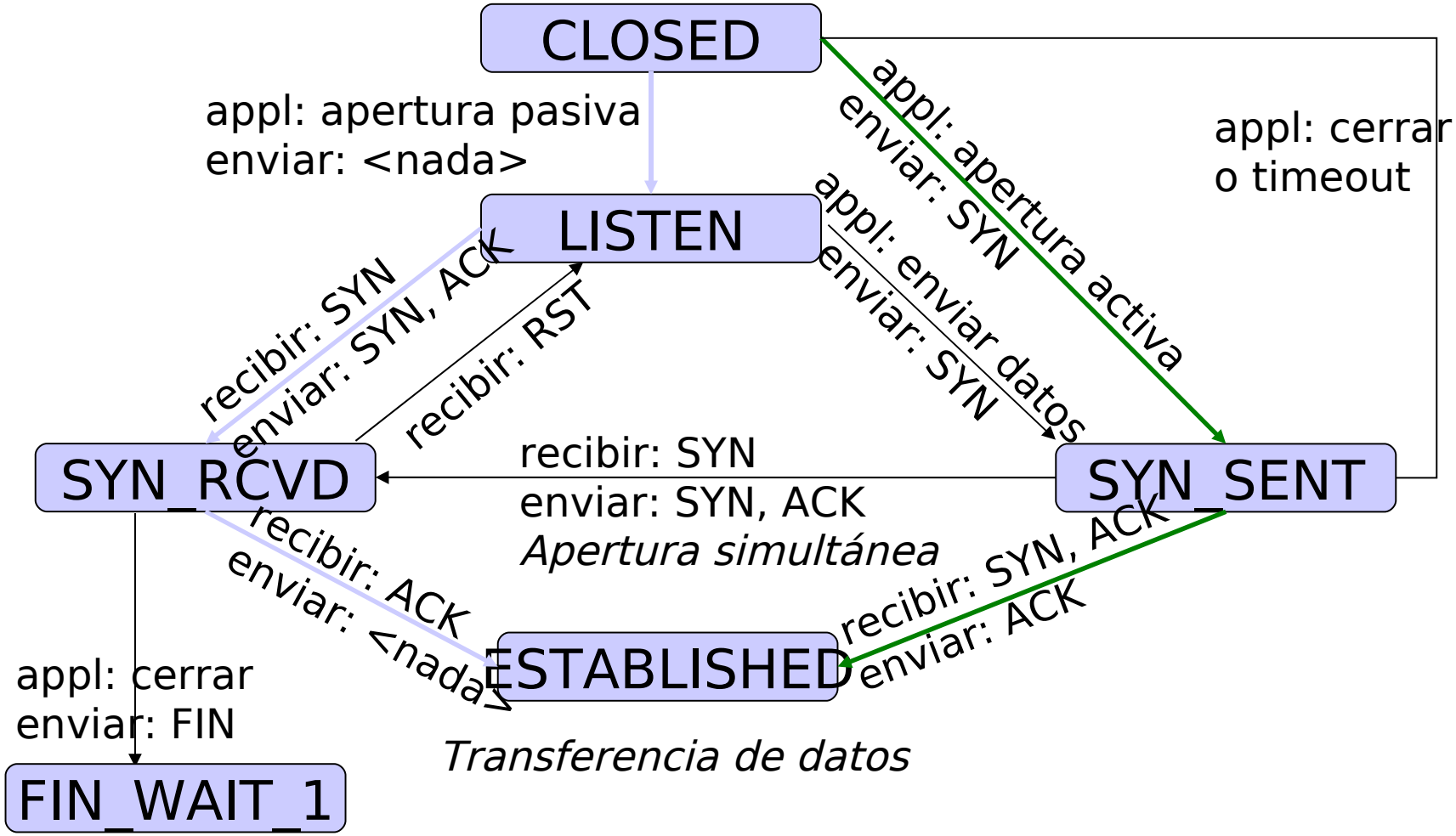


Conexión TCP: Half-close

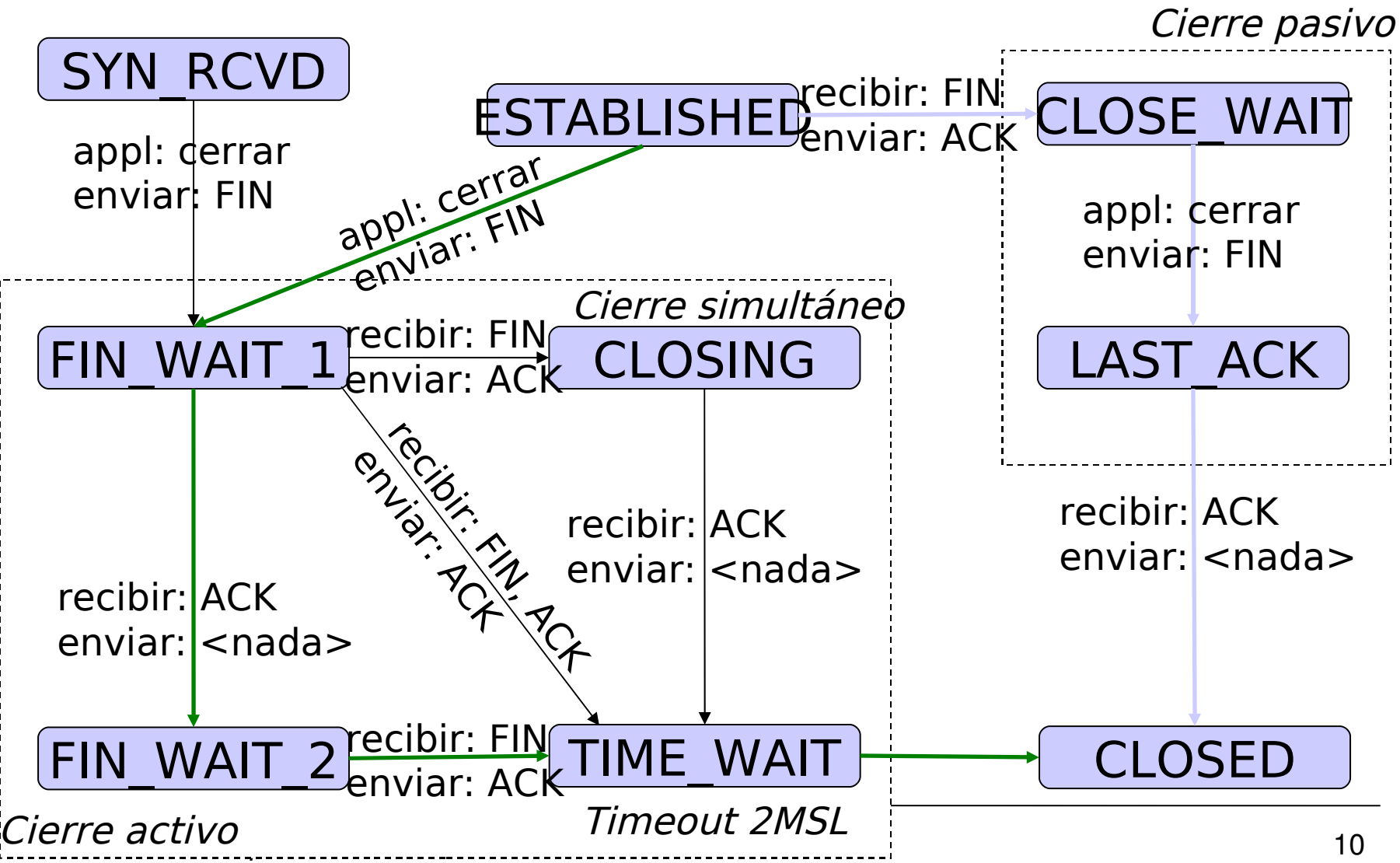
- Una parte termina la conexión (da por finalizado su emisión de datos) mientras todavía está recibiendo datos (half-close).
  - No puede enviar más datos, pero si ACKs, ...
- Pocas aplicaciones se beneficien de esta utilidad:
  - rsh servidor sort < ficheroEntrada.txt



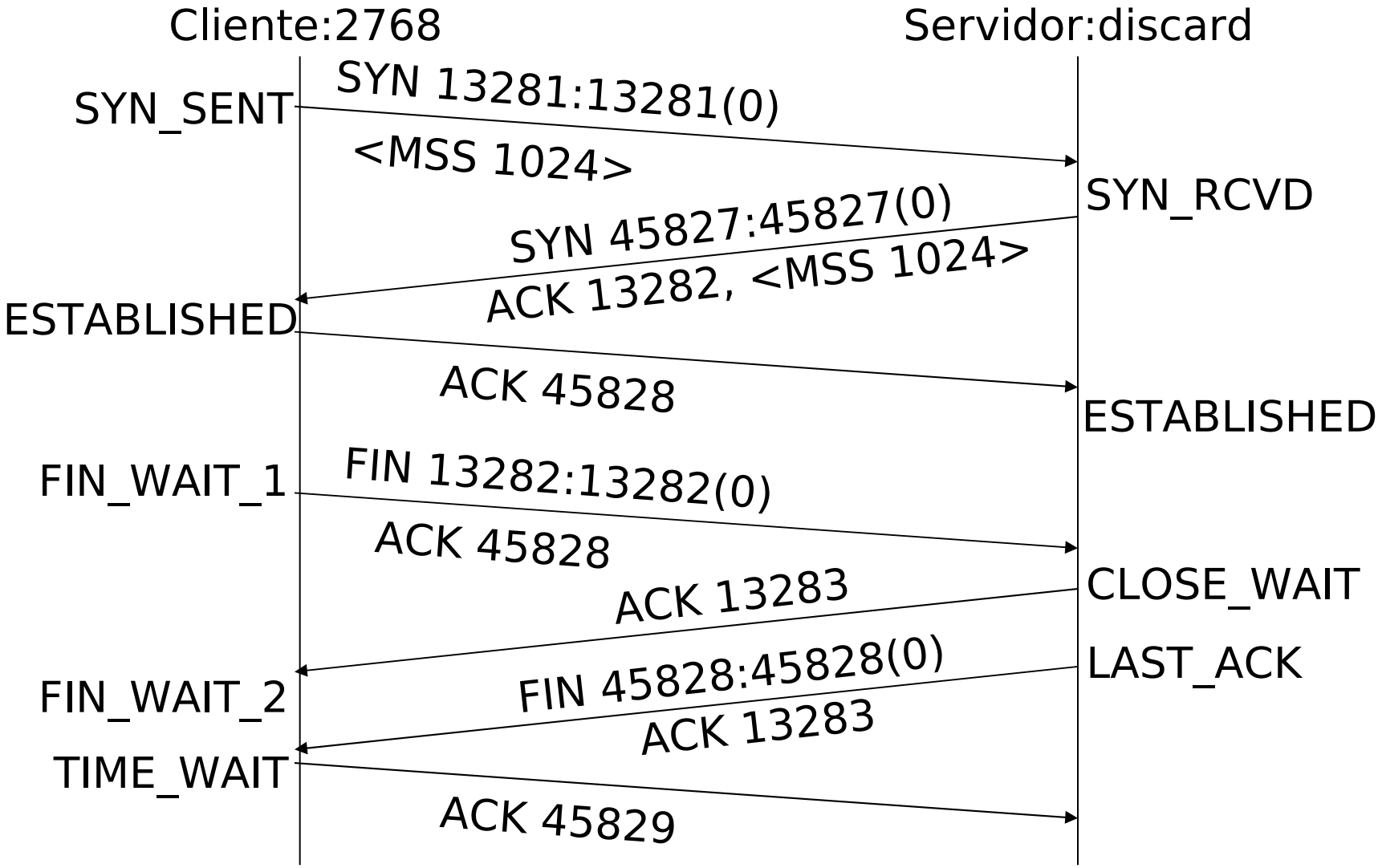
TCP: Diagrama de estados



TCP: Diagrama de estados



TCP: Diagrama de estados



TCP: Diagrama de estados

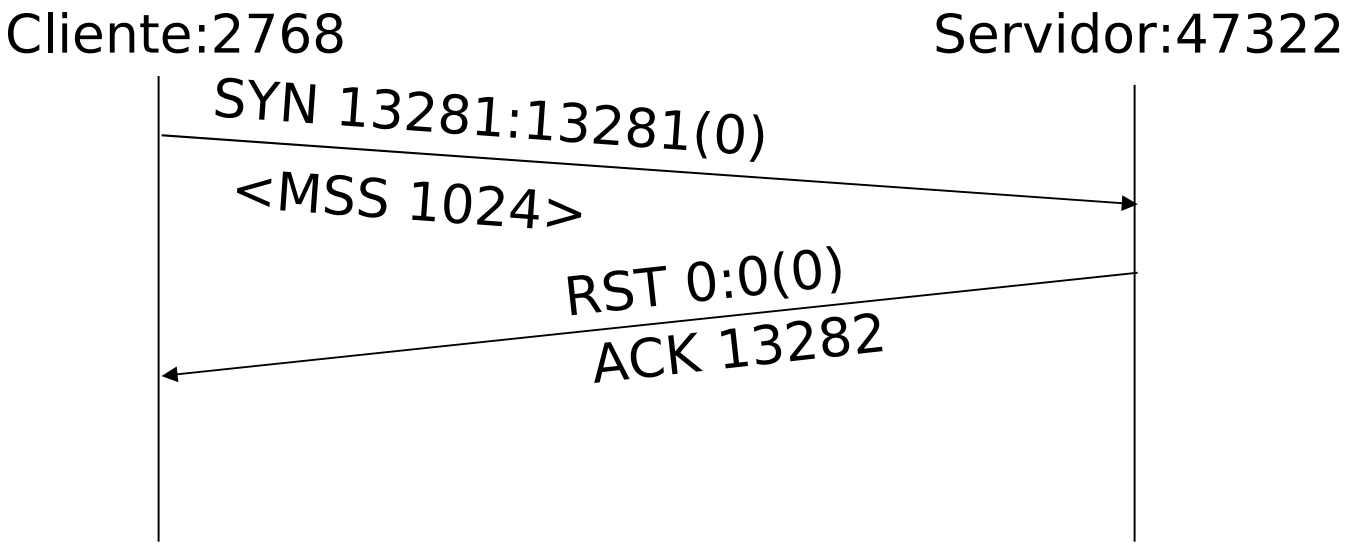
- Estado TIME_WAIT: estado de espera durante 2 MSL
  - MSL (Maximum Segment Lifetime): tiempo máximo que un segmento puede estar en la red antes de ser descartado.
  - Permite a TCP reenviar el ACK en caso de que se haya perdido (el otro extremo reenviará el FIN).
  - Mientras la conexión está en este 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.
- Quiet time: un host permanecerá durante MSL sin crear ninguna conexión después de reiniciarse.
- 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 suelen establecer un tiempo de espera (p.e. 10 minutos), 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:
  - Intento de conexión a un puerto no existente
  - Abortar una conexión
  - Respuesta ante conexiones semi-abiertas

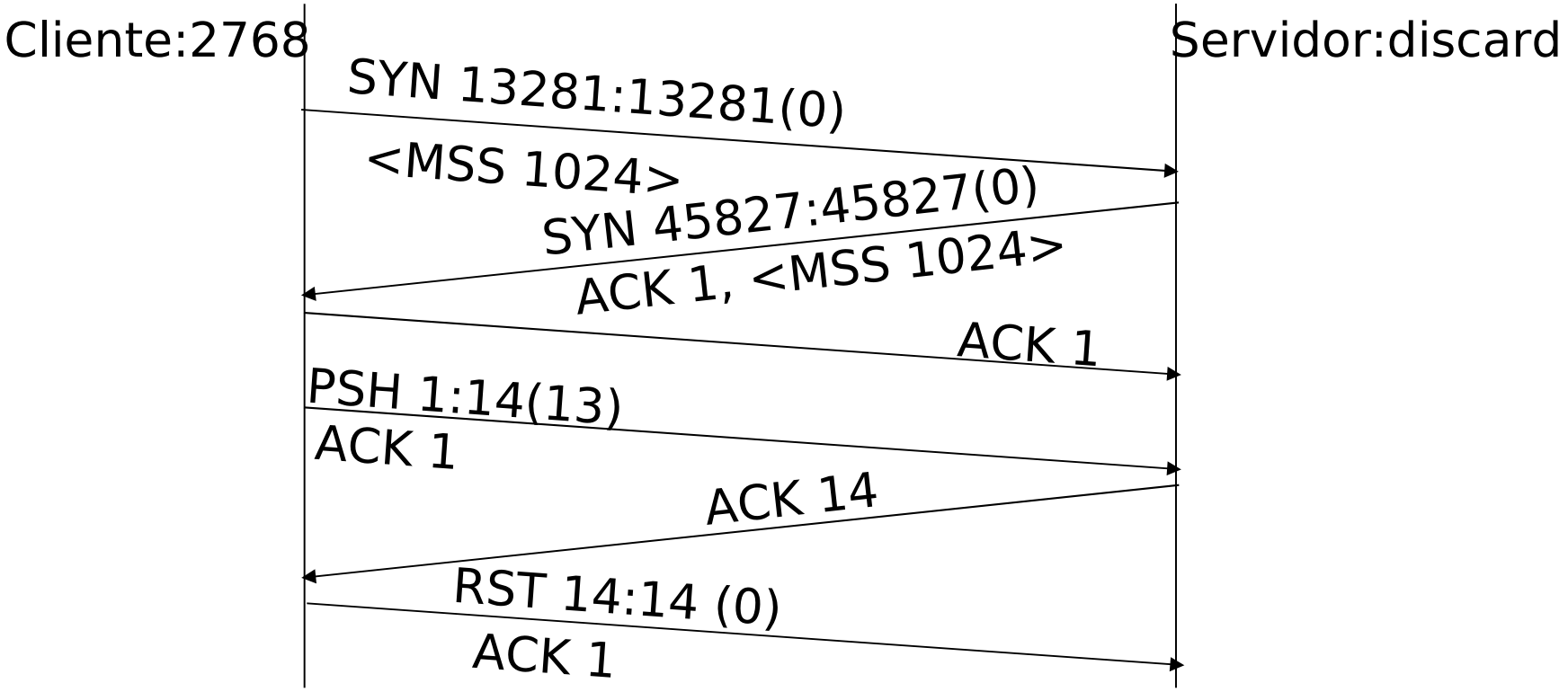
TCP: Segmentos de Reset

- Intento de conexión a un puerto no existente:
  - Si se trata de realizar una conexión a un puerto sin aplicación asociada ➔ No hay ningún proceso escuchando una posible llegada de conexiones.



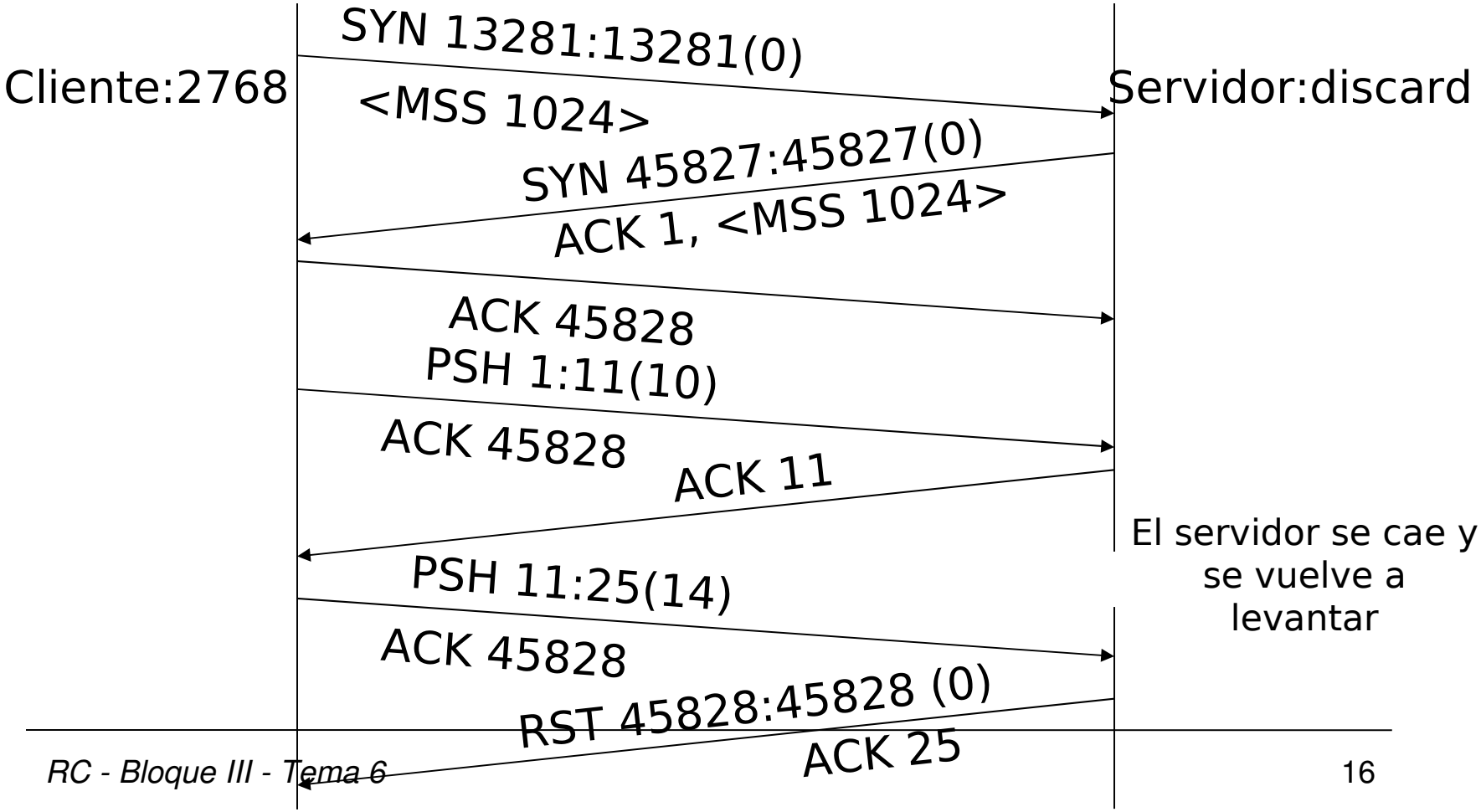
TCP: Segmentos de Reset

- Abortar conexión
  - Existe la posibilidad de acabar mediante un paquete con el bit de RST (terminación anormal) ➔ Cualquier dato esperando ser enviado será descartado automáticamente.



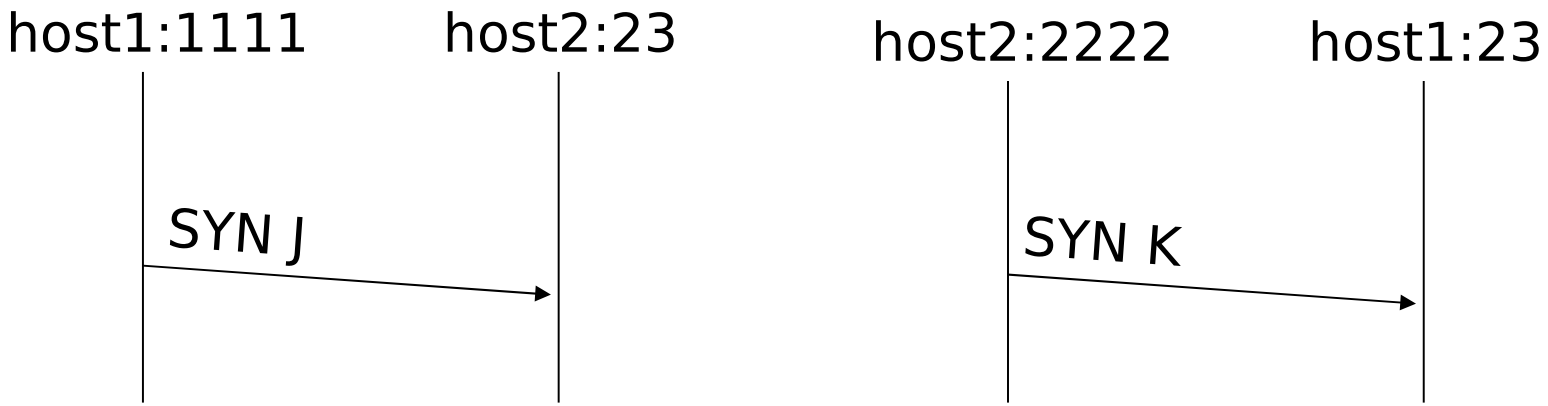
TCP: Segmentos de Reset

- Respuesta ante conexiones semi-abiertas
  - Sucede cuando el servidor se cae y se vuelve a levantar, después de lo cual recibe datos del cliente.
  - La respuesta en este caso es un segmento de RESET

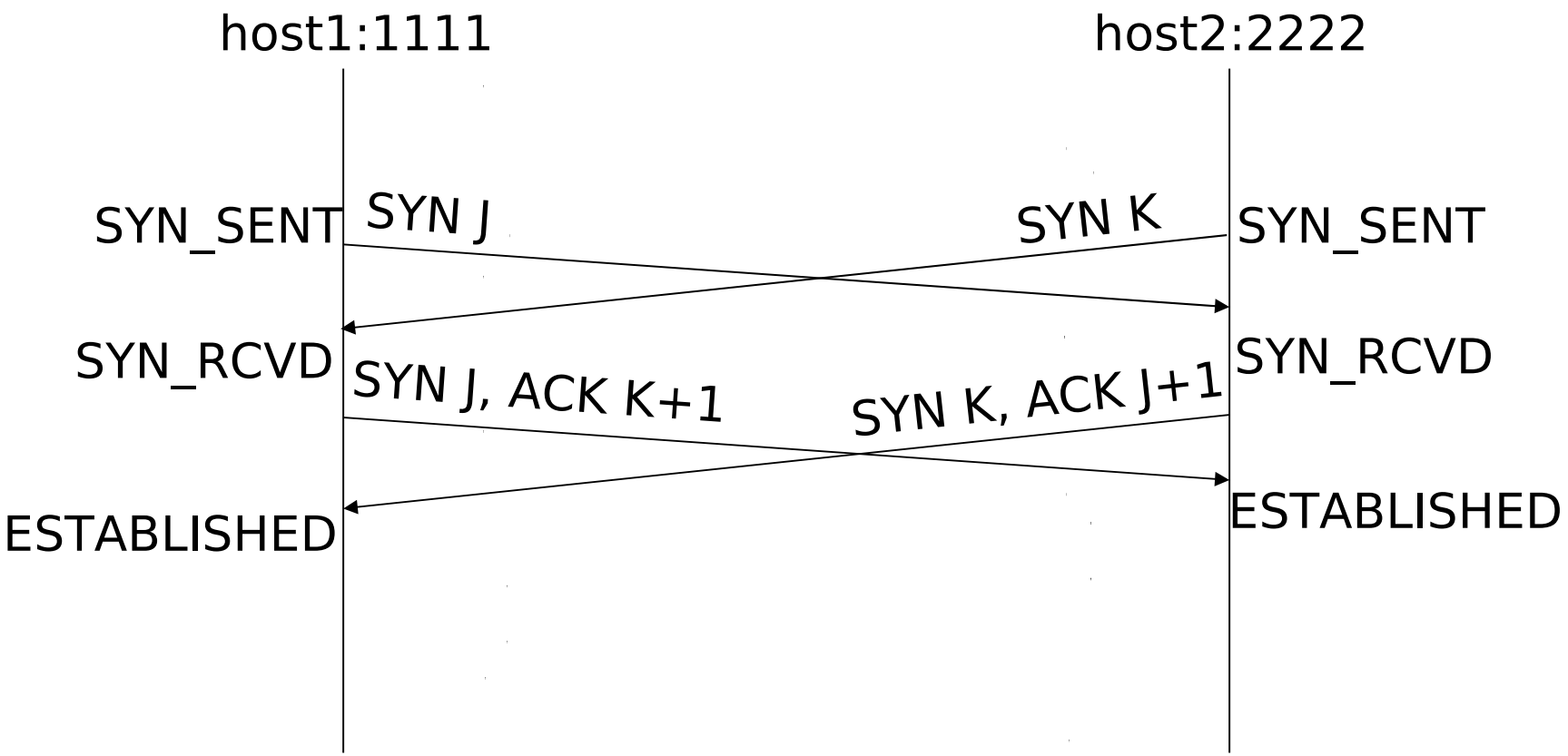


TCP: Establecimiento simultáneo

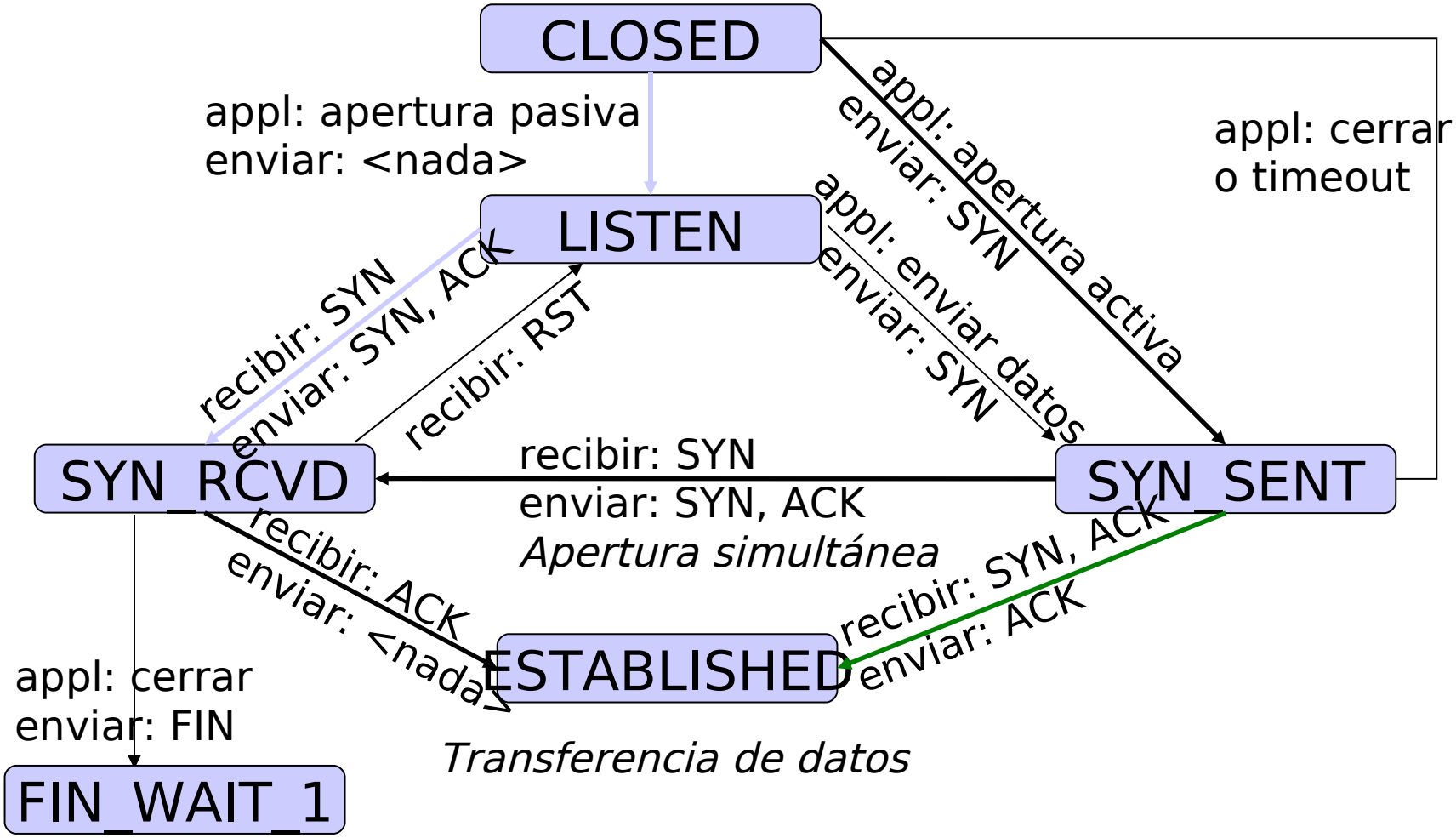
- La posibilidad de establecimiento de conexión simultáneo es mínima aunque posible:
  - Dos aplicaciones en dos hosts se envían mensajes de conexión simultáneamente.
  - No existe una figura clara de cliente y servidor, ahora hay dos “clien-servidor”.
- TCP está diseñado para manejar correctamente esta posibilidad.
- Este tipo de apertura requiere el envío de 4 segmentos (uno más de lo habitual) y sigue algunas variantes sobre el diagrama de transición de estados habitual.
- ¿Esto es una apertura simultánea?
  - host1% telnet host2 host2% telnet host1



TCP: Establecimiento simultáneo

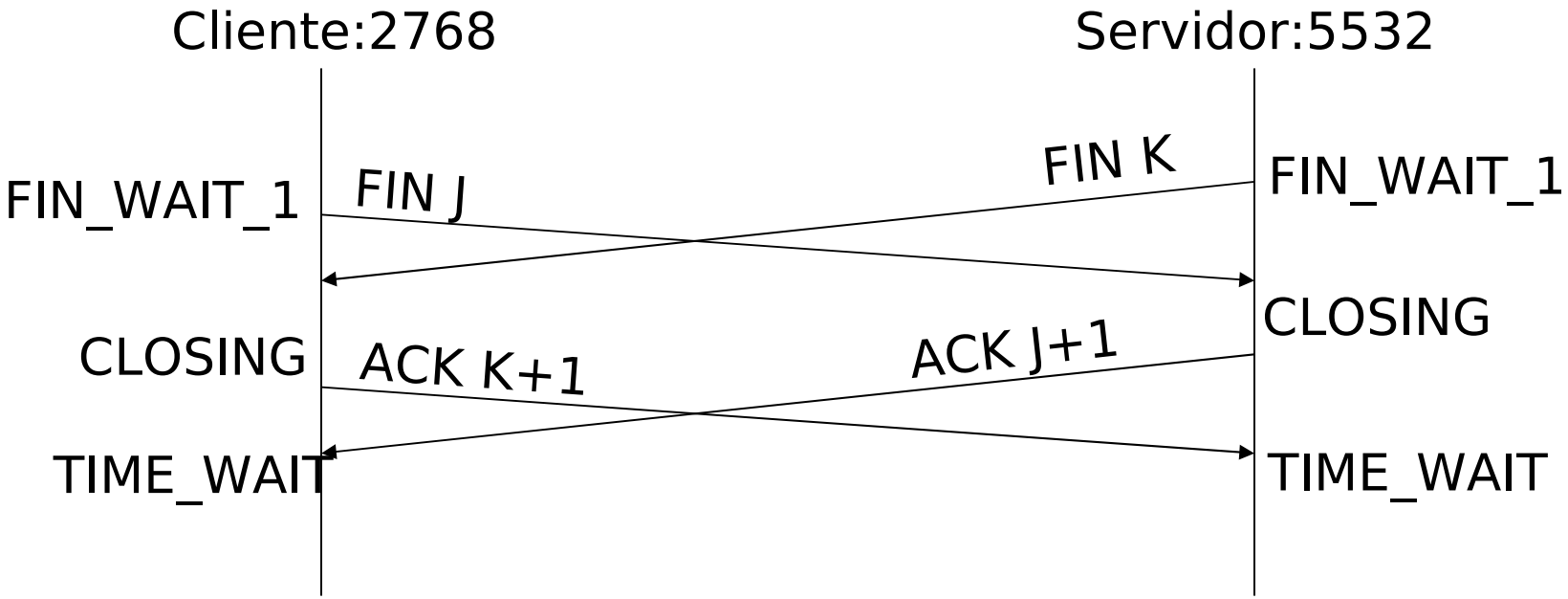


TCP: Establecimiento simultáneo

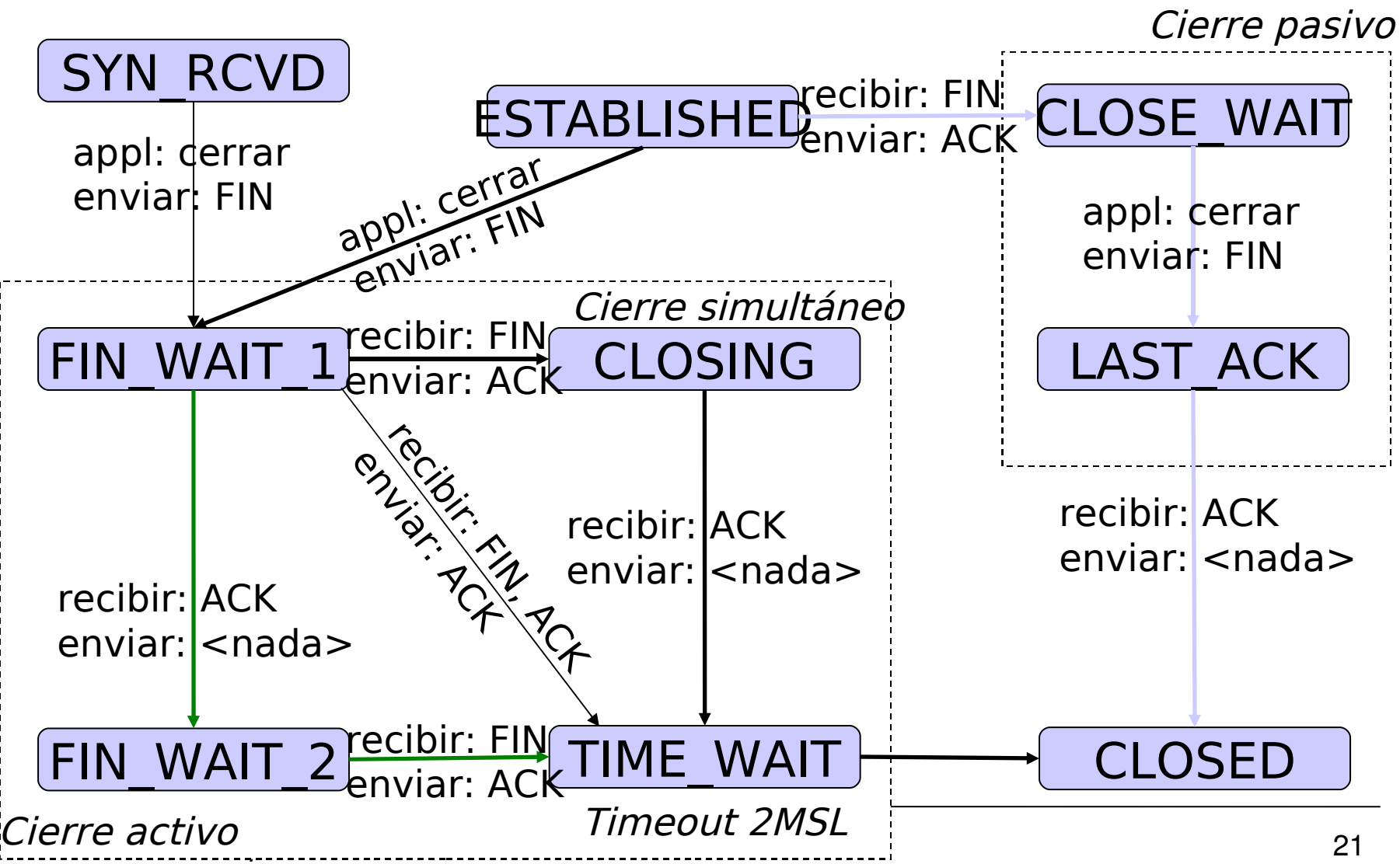


TCP: Cierre simultáneo

- TCP contempla la posibilidad de dos cierres simultáneos (simultaneous close).
- En este caso el número de segmentos intercambiados es el mismo que en un cierre normal TCP.



TCP: Cierre simultáneo



TCP: Diseño de servidores

- 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.