Bloque III: El nivel de transporte

Tema 7: Intercambio de datos TCP

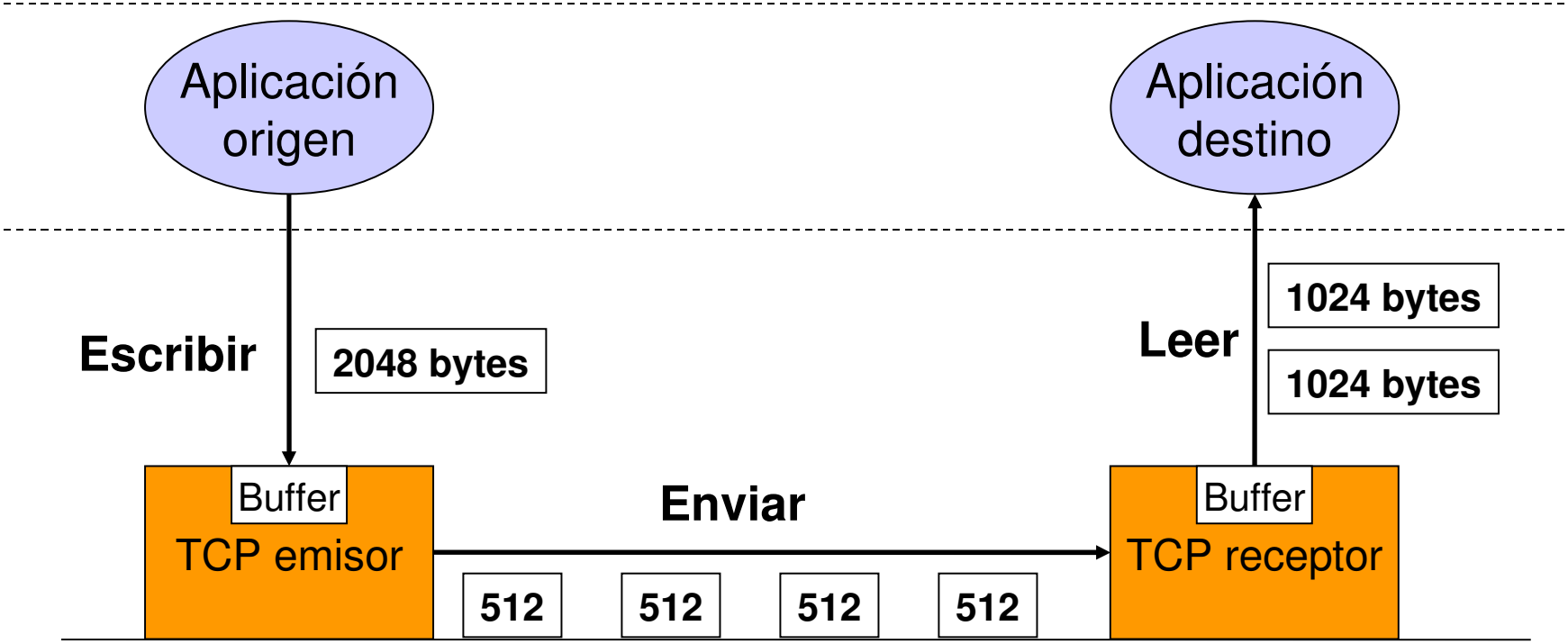
Índice

- Bloque III: El nivel de transporte
  - Tema 7: Intercambio de datos TCP
    - Flujo de datos interactivo
      - ACKs retardados
      - Algoritmo de Nagle
      - Modo de urgencia
    - Flujo de datos no interactivo
      - Control de flujo
      - Control de congestión

- **Referencias**
  - Capítulo 3 de “Redes de Computadores: Un enfoque descendente basdado en Internet”. James F. Kurose, Keith W. Ross. Addison Wesley, 2ª edición. 2003.
  - Capítulos 19 y 20 de “TCP/IP Illustrated, Volume 1: The Protocols”, W. Richard Stevens, Addison Wesley, 1994.

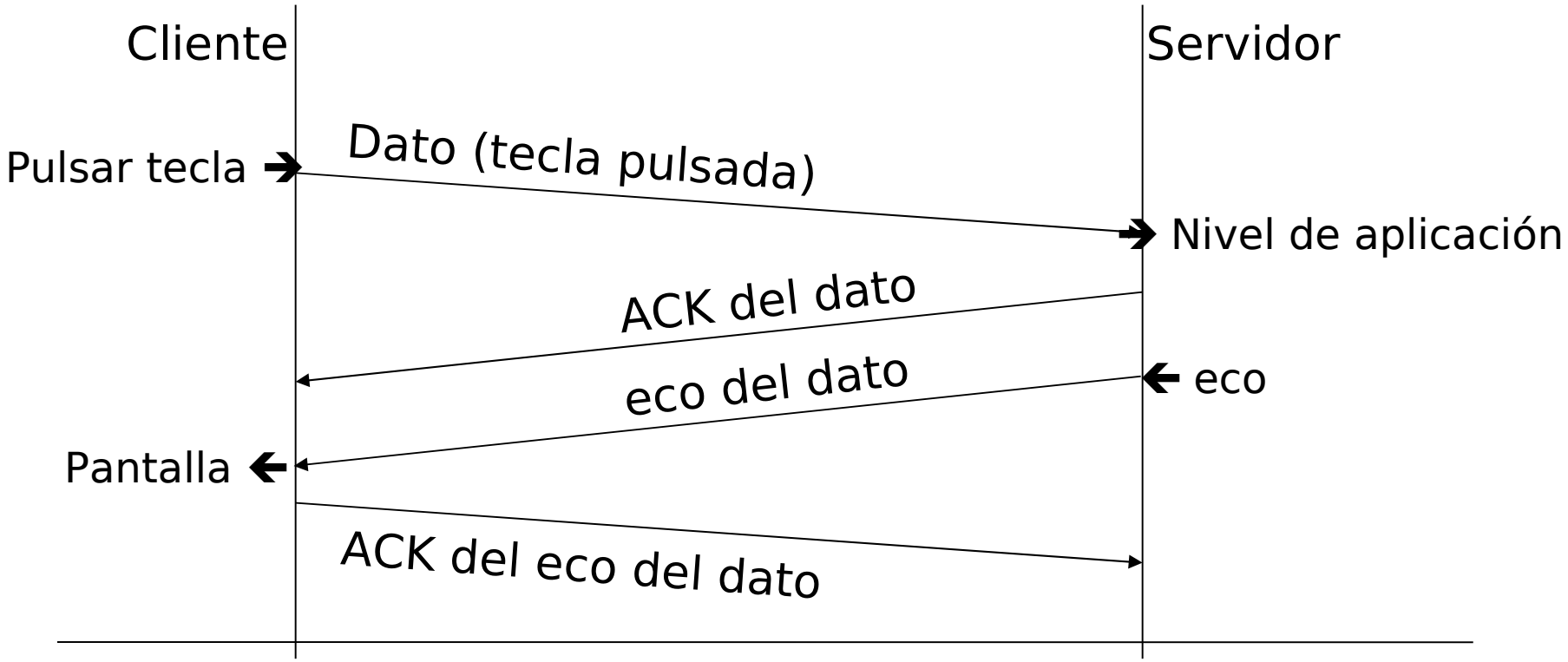
TCP: Intercambio de datos

- En TCP se consideran dos tipos de tráfico de datos:
  - **Interactivo**: gran número de segmentos de pequeño tamaño (menos de 10 bytes). Por ejemplo: telnet, rlogin.
  - **No Interactivo**: segmentos de gran tamaño, normalmente el máximo permitido por las limitaciones de la red. Por ejemplo: FTP, e-mail.



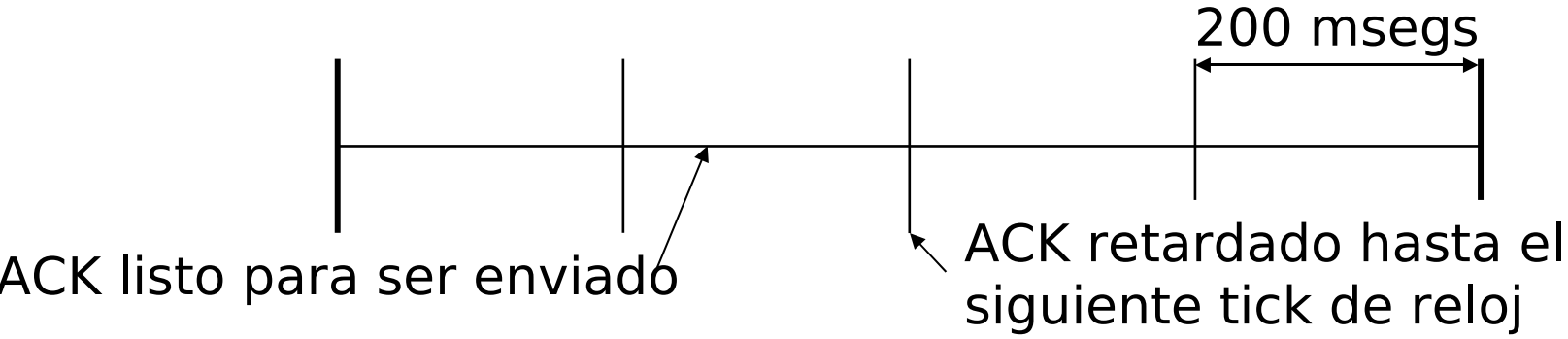
Flujo de datos interactivo

- Funcionamiento del telnet o rlogin:
  - Envío de la tecla pulsada por el cliente
  - ACK de la tecla pulsada por el cliente
  - Eco de la tecla desde el servidor
  - ACK del eco

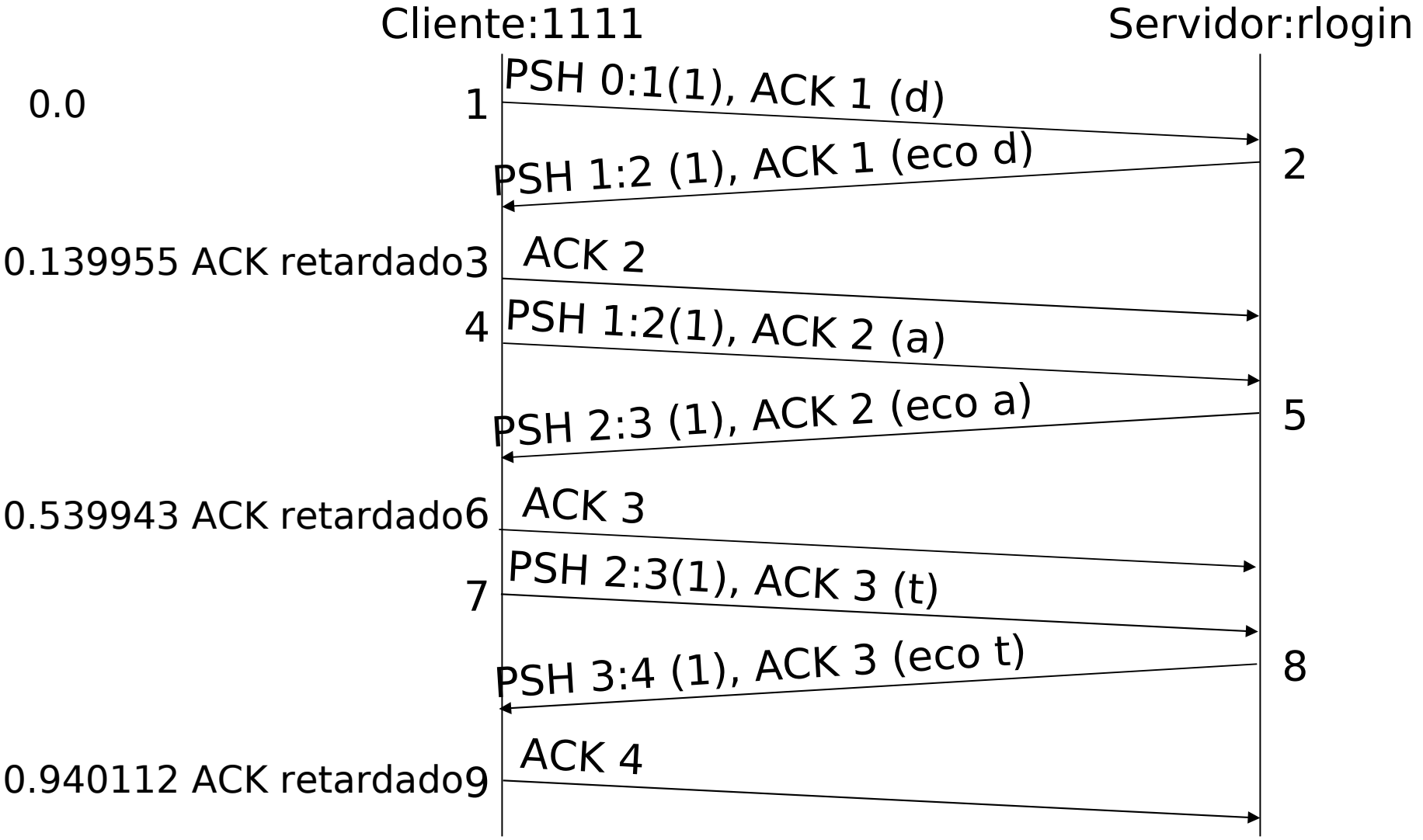


Flujo de datos interactivo

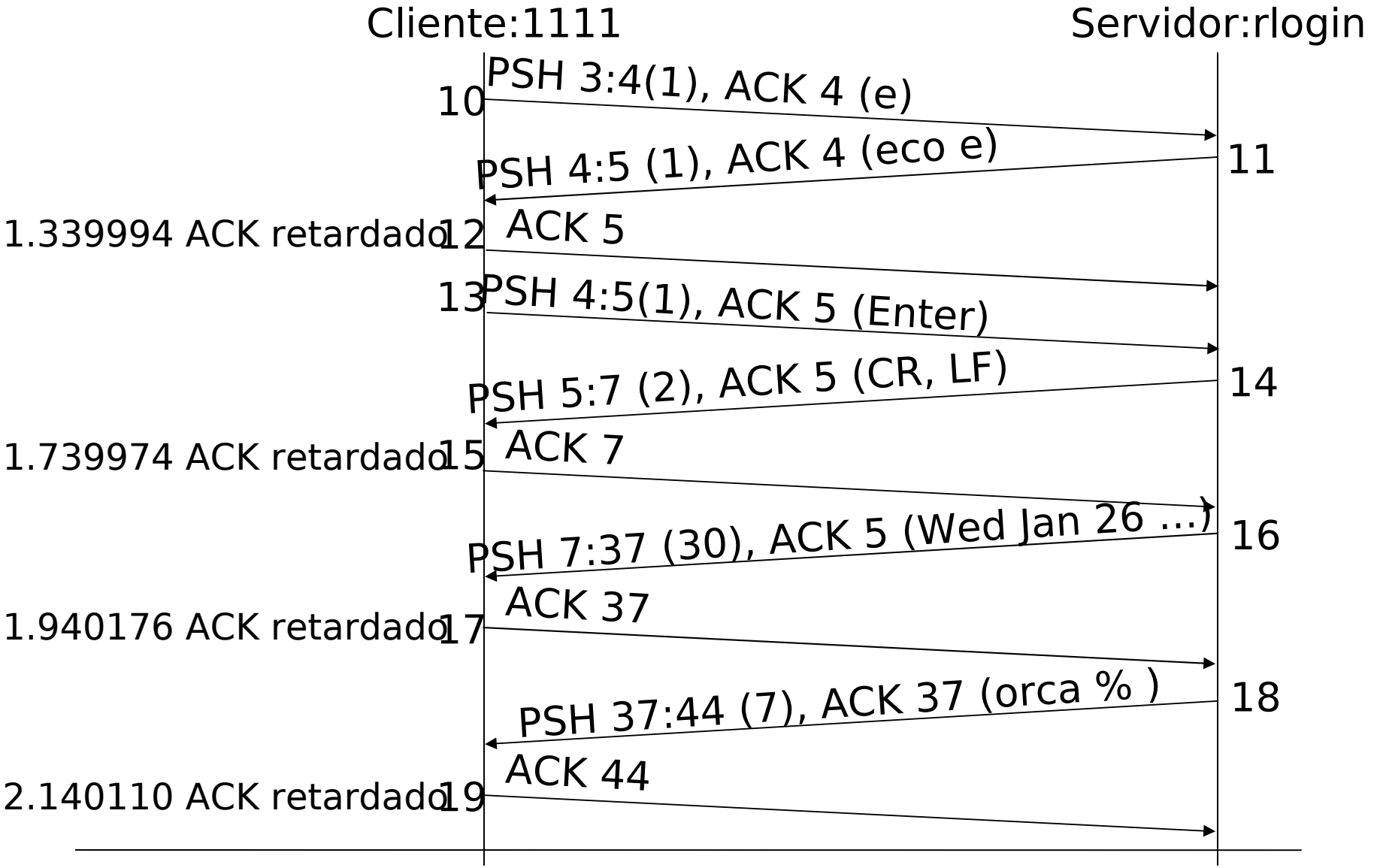
- ACKs retardados:
  - Objetivo: enviar el ACK + eco en un único datagrama.
- TCP no envía el ACK inmediatamente al recibir el dato, sino que retarda la salida del ACK esperando un tiempo para ver si hay datos para enviarlos con el propio ACK.
- El tiempo de espera es de 200 mseg:
  - No en valor absoluto, sino que se utiliza un reloj que da ticks cada 200 mseg.
- Tanto en el cliente como en el servidor se utilizan los acks retardados.



ACKs retardados



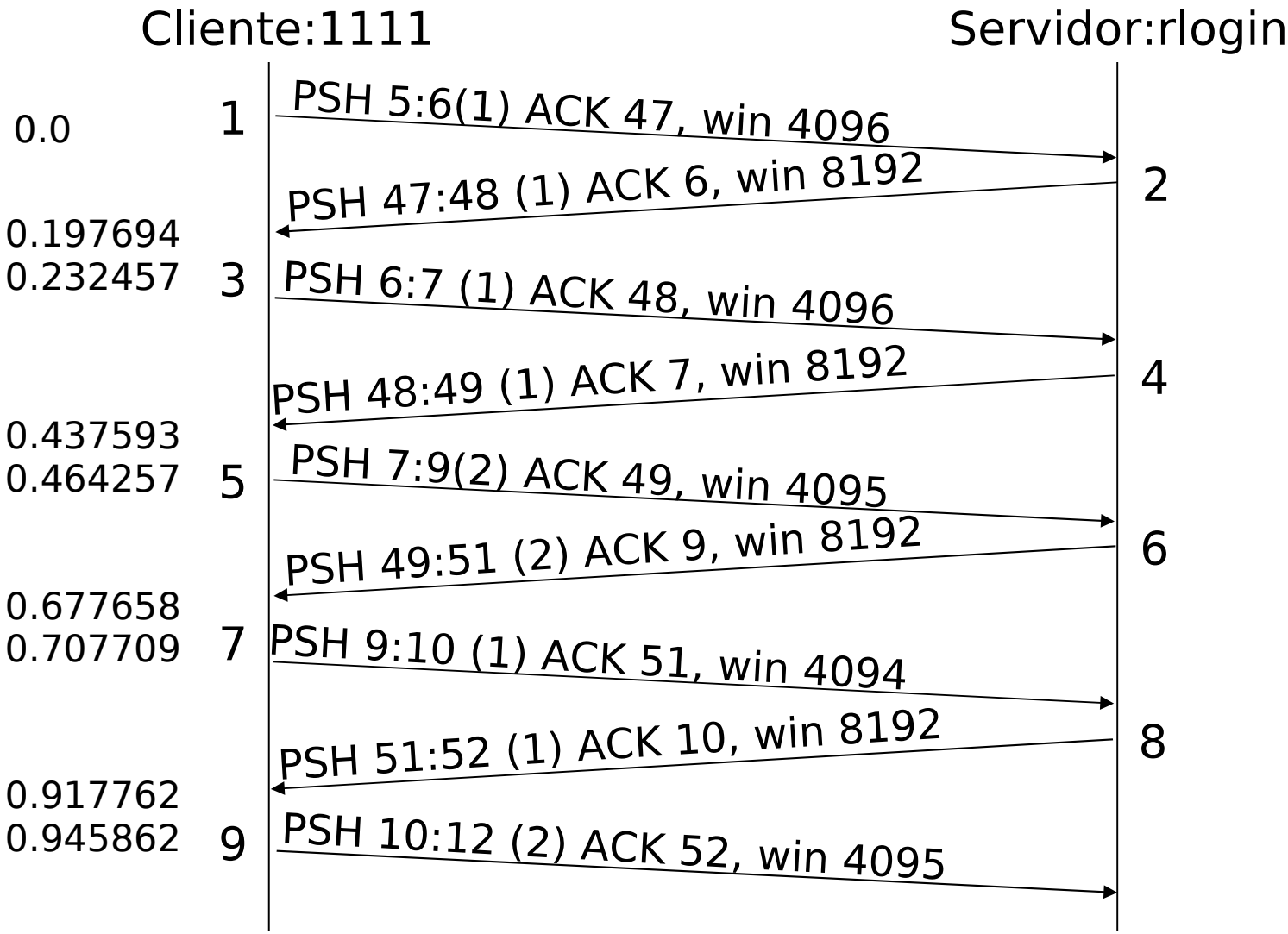
ACKs retardados



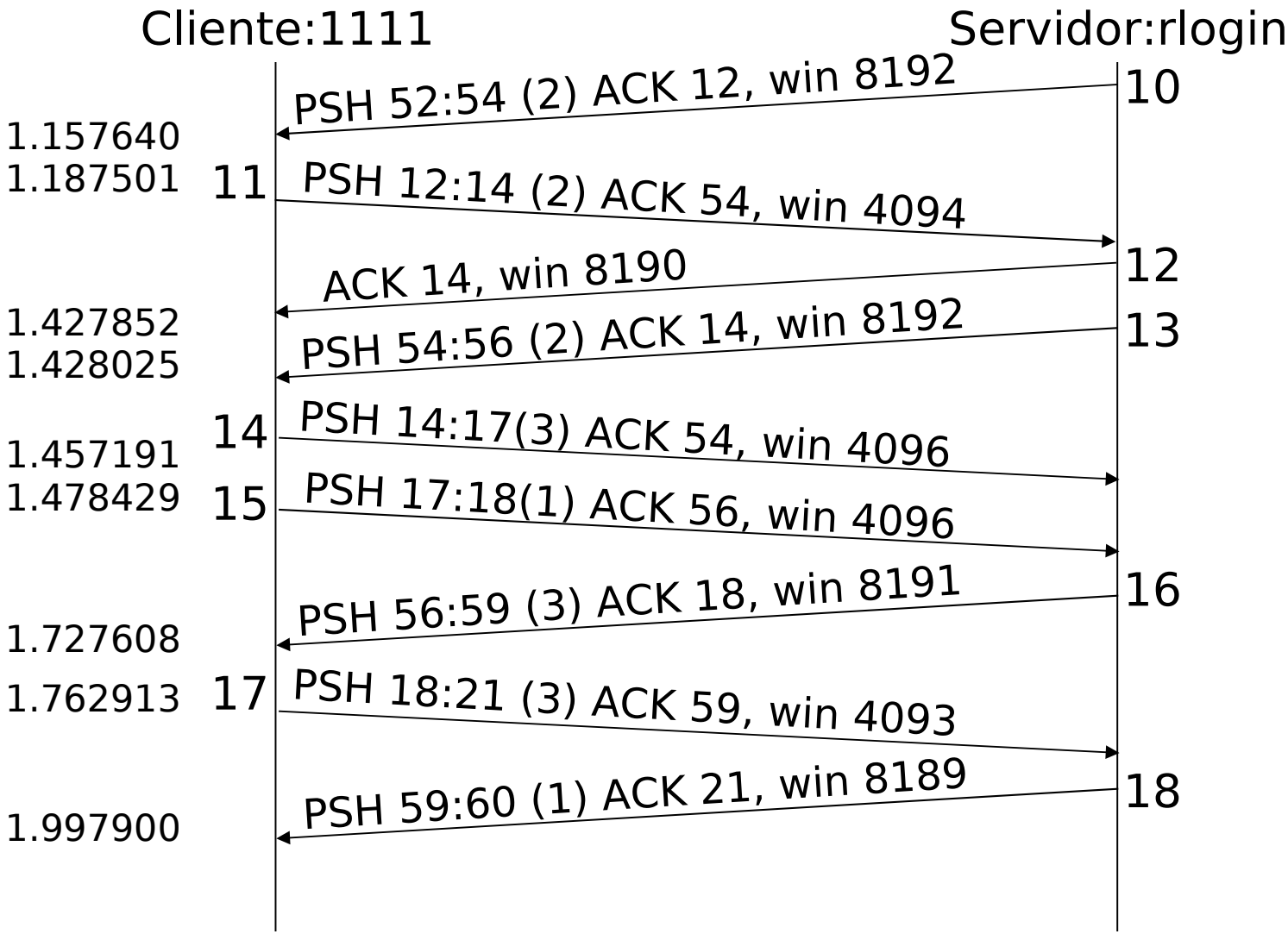
Flujo de datos interactivo

- El tráfico interactivo genera gran cantidad de paquetes de tamaño muy pequeño (1 byte datos + 20 cab. TCP + 20 cab. IP = 41 bytes) = tinygrams.
  - En las redes de área local no presenta ningún problema
  - En las WAN supone una gran sobrecarga para la red.
- Algoritmo de Nagle:
  - Pretende resolver este problema, y se aplica en una conexión TCP de tráfico interactivo en una red de área extensa.
  - Enunciado: "*Una conexión TCP puede tener un único segmento pequeño que no haya sido confirmado. No se pueden enviar otros segmentos hasta recibir un ACK. En cambio, esos datos se almacenan y son enviados por TCP al llegar el ACK*."
  - Es auto-ajustable: cuanto más rápido lleguen los ACKs ➔ más rápido se enviarán los datos.

Algoritmo de Nagle



Algoritmo de Nagle



Algoritmo de Nagle

- Se usan 9 segmentos para enviar 16 bytes de datos.
- Segmentos 14 y 15 consecutivos, ¿cumplen el algoritmo de Nagle?
  - Si, se corresponden con los ACKs 12 y 13, respectivamente, que se han recibido casi en el mismo instante (0.2 msegs de diferencia).
- Segmento 12 es un ACK retardado (servidor ➔ cliente).
- ACK = TCP (nivel de transporte)
- Win = nivel de aplicación

- En algunos casos no interesa el algoritmo de Nagle.
  - Por ejemplo, al utilizar las XWindows es necesario enviar mensajes pequeños (movimientos del ratón) sin retardos para conseguir una interacción real.
  - En este caso, el algoritmo de Nagle puede introducir retardos adicionales al ejecutar una aplicación interactiva a través de una WAN, al generar eventos (pulsaciones, movimientos ratón, …) que producen datos multi-byte.

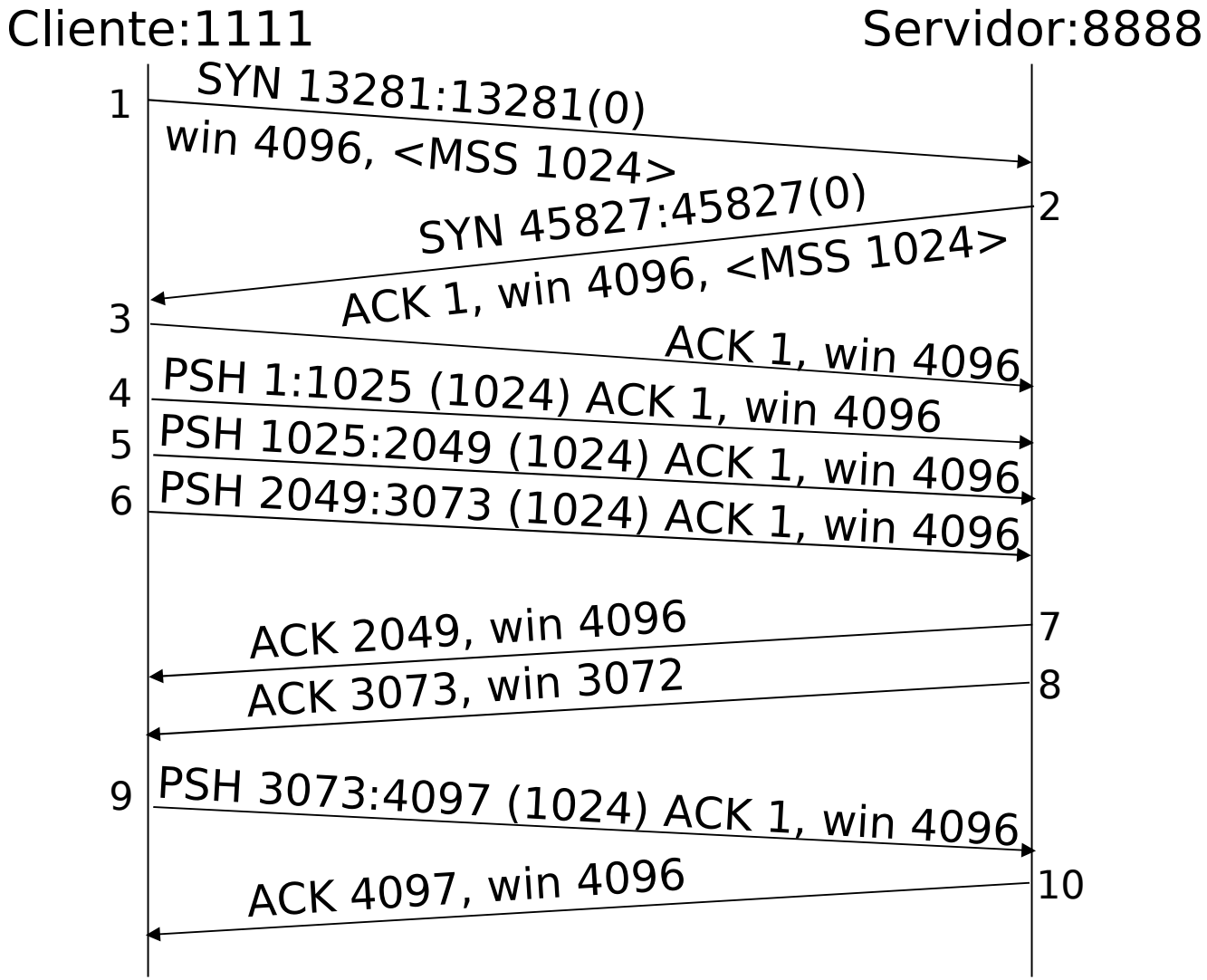
Flag URG

- Modo de urgencia: permite a un extremo indicar que se han introducido datos urgentes en el flujo normal de datos.
  - Se activa el flag de Urgencia (URG)
  - Puntero de urgencia: offset a sumar al número de secuencia de la cabecera TCP para obtener el número de secuencia del último byte de los datos urgentes.
- Se utiliza en aplicaciones como telnet y rlogin cuando se pulsa la tecla de interrupción. O en el FTP cuando se aborta la transferencia de un fichero.

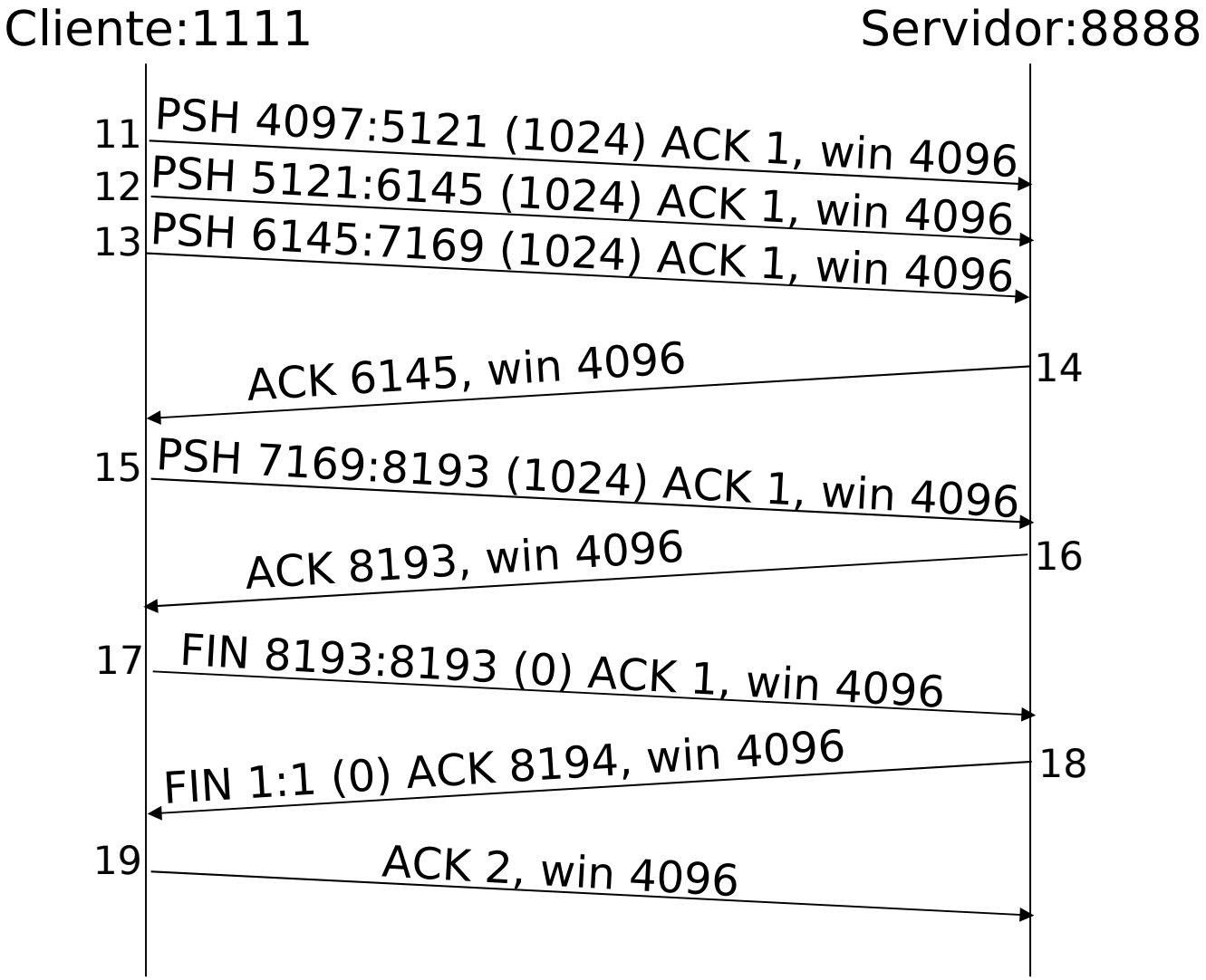
Flujo de datos no interactivo

- También llamado flujo de datos en masa (*bulk data flow*).
- En este tipo de tráfico se generan pocos segmentos, pero de gran tamaño.
- El principal problema a resolver en este tipo de tráfico es el **control de flujo**:
  - Evitar que un emisor rápido sature a un receptor lento.
- TCP utiliza una ventana deslizante: permite al emisor enviar múltiples paquetes antes de parar y esperar por el ACK, lo que da una mayor rapidez a este tipo de tráfico.
- Con un protocolo de ventana deslizante no es necesario confirmar todos los paquetes recibidos, sino que se pueden confirmar varios paquetes simultáneamente.

Control de flujo



Control de flujo

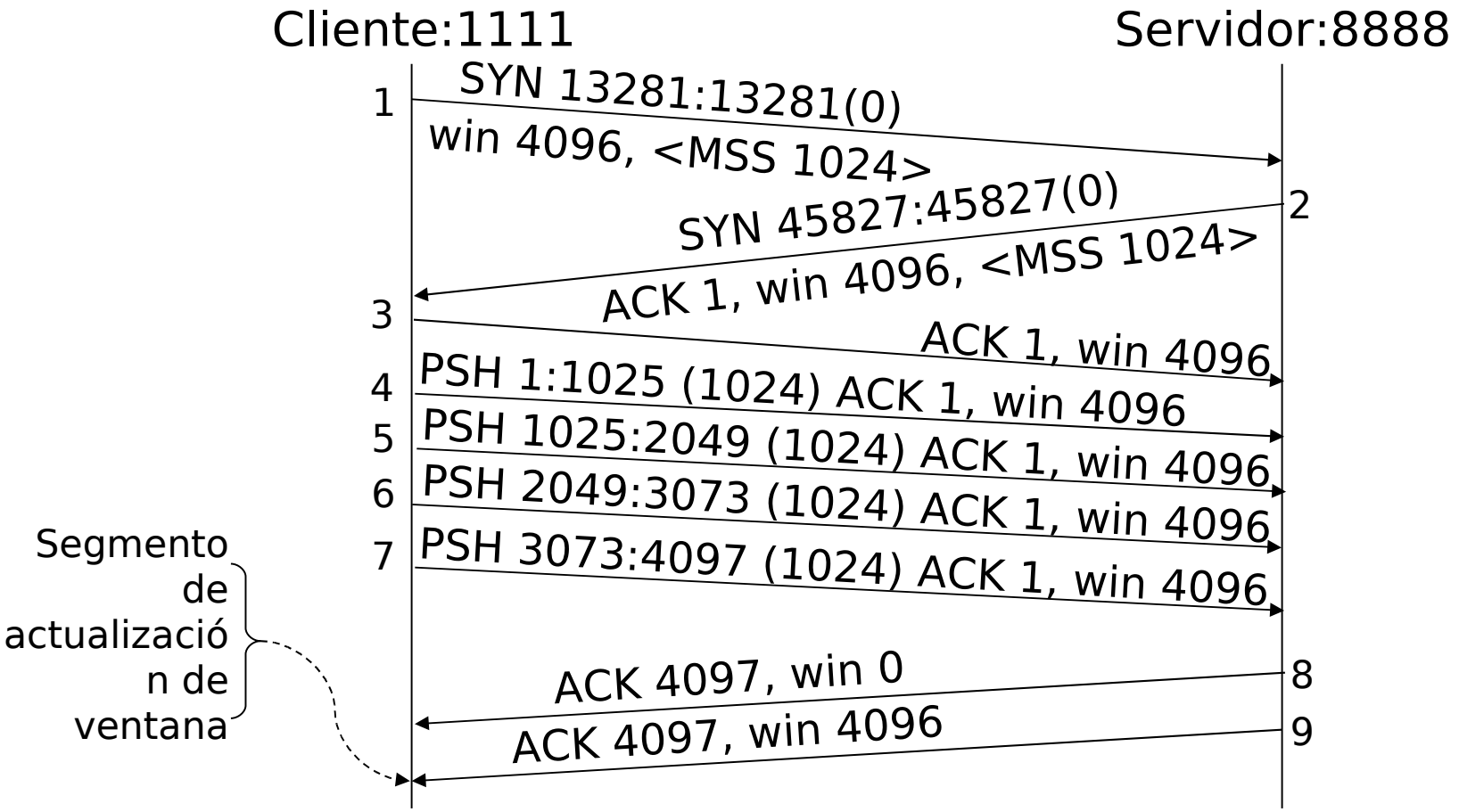


Control de flujo

- El envío de los mensajes no es determinista, depende de múltiples factores: la carga de la red, la carga del receptor y emisor, …
- TCP utiliza ACKs acumulativos:
  - ACK 2049: confirma que se ha recibido correctamente hasta el 2048, y que el siguiente byte que se espera recibir es el 2049.
  - Segmentos 7, 14 y 16 son ACKs acumulativos.
- Flag PSH (PUSH): notificación del emisor al receptor para enviar todos los datos recibidos a la aplicación.
  - Datos recibidos en el segmento PSH + datos almacenados en el buffer
  - Permite al emisor notificar al receptor que puede pasar los datos al receptor, que no es necesario esperar por más datos.
  - No se puede especificar a nivel de aplicación. Es incluido automáticamente por TCP en cada grupo de datos transmitido.

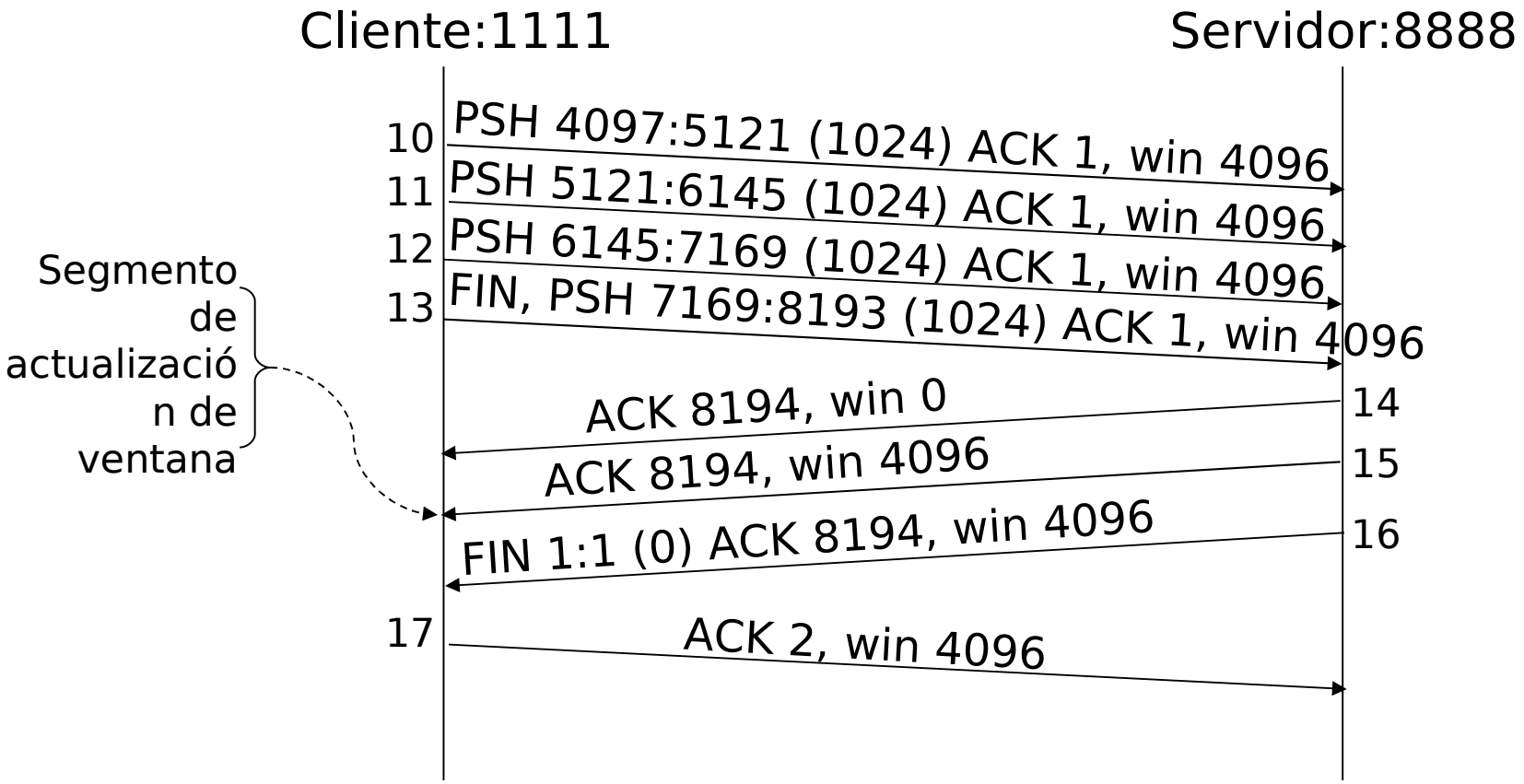
Control de flujo

- Emisor rápido, receptor lento



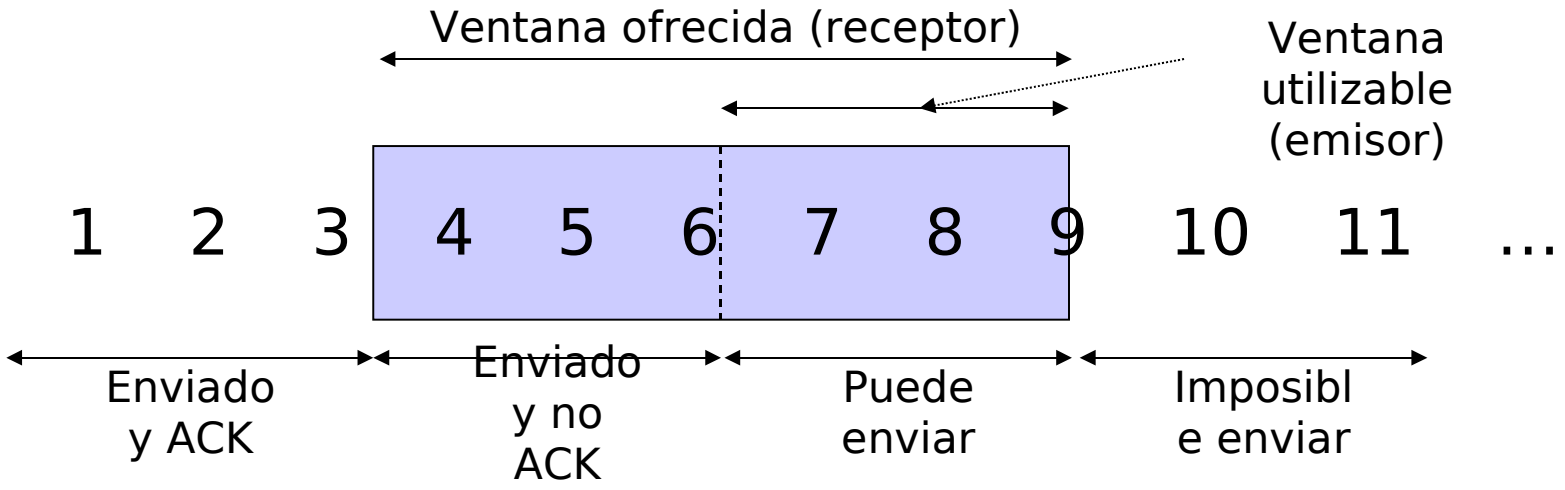
Control de flujo

- Emisor rápido, receptor lento (continuación)



Control de flujo

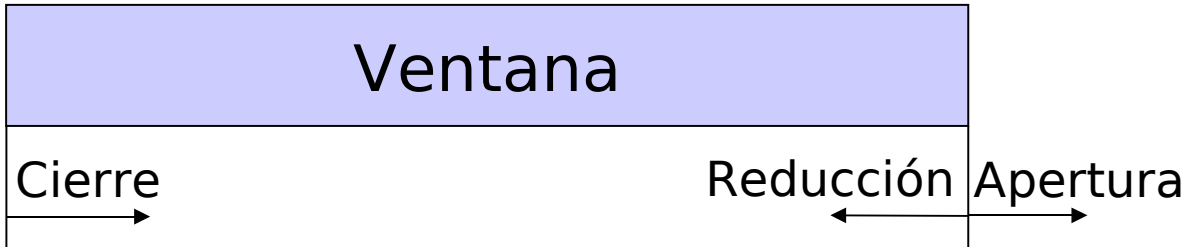
- Ventana deslizante
  - Funcionamiento:



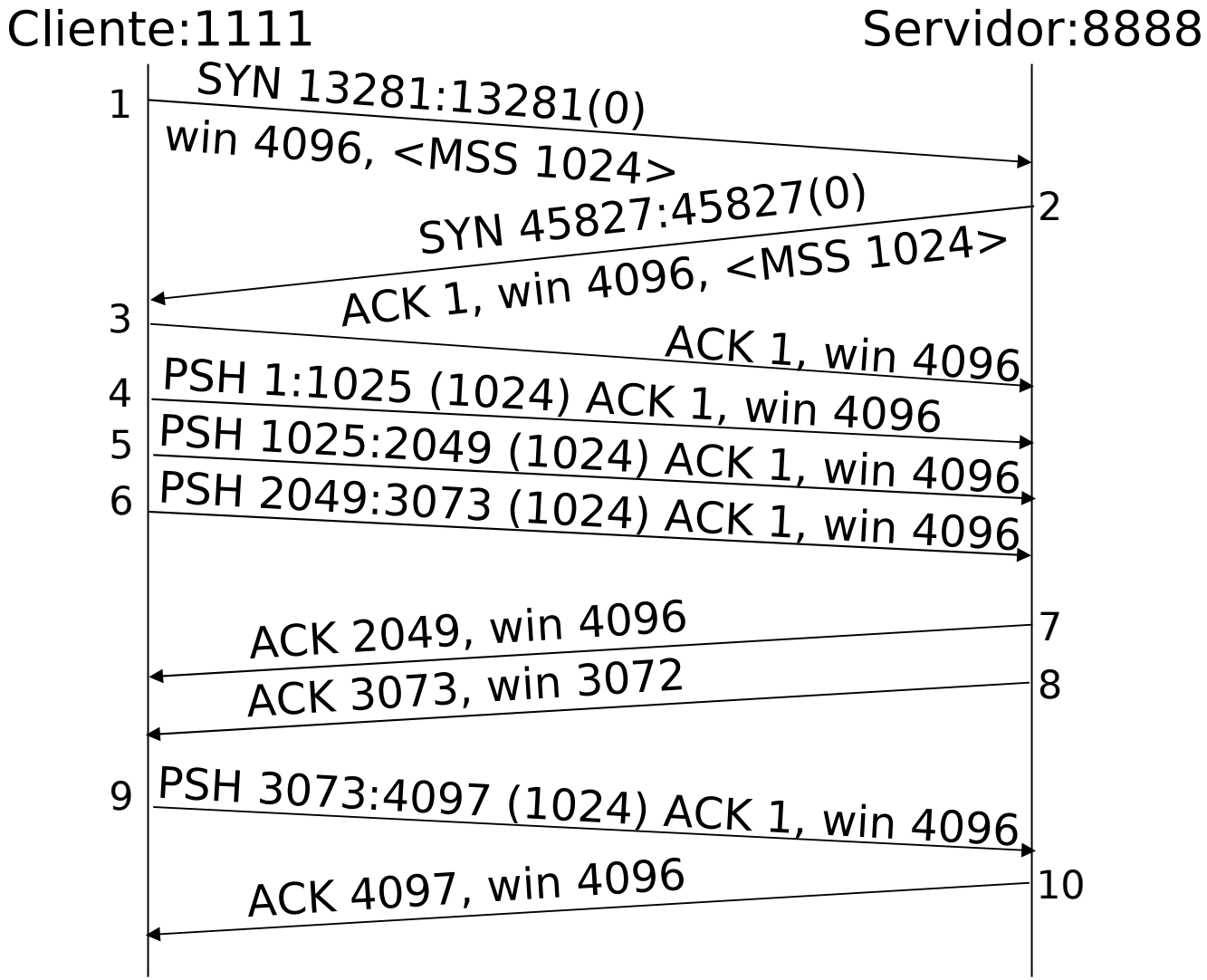
  - **Ventana ofrecida** (receptor): número de bytes que indica el receptor que puede recibir, en el momento de enviar el paquete.
  - **Ventana utilizable** (emisor): número de bytes que se pueden enviar inmediatamente.

Control de flujo

- Movimientos de la ventana:
  - Cierre (el eje izquierdo se desplaza a la derecha): se envían y se agradecen datos.
  - Apertura (el eje derecho se desplaza a la derecha): el proceso receptor lee los datos agradecidos, y libera espacio en el buffer.
  - Reducción (el eje derecho se desplaza a la izquierda): no se produce en TCP.

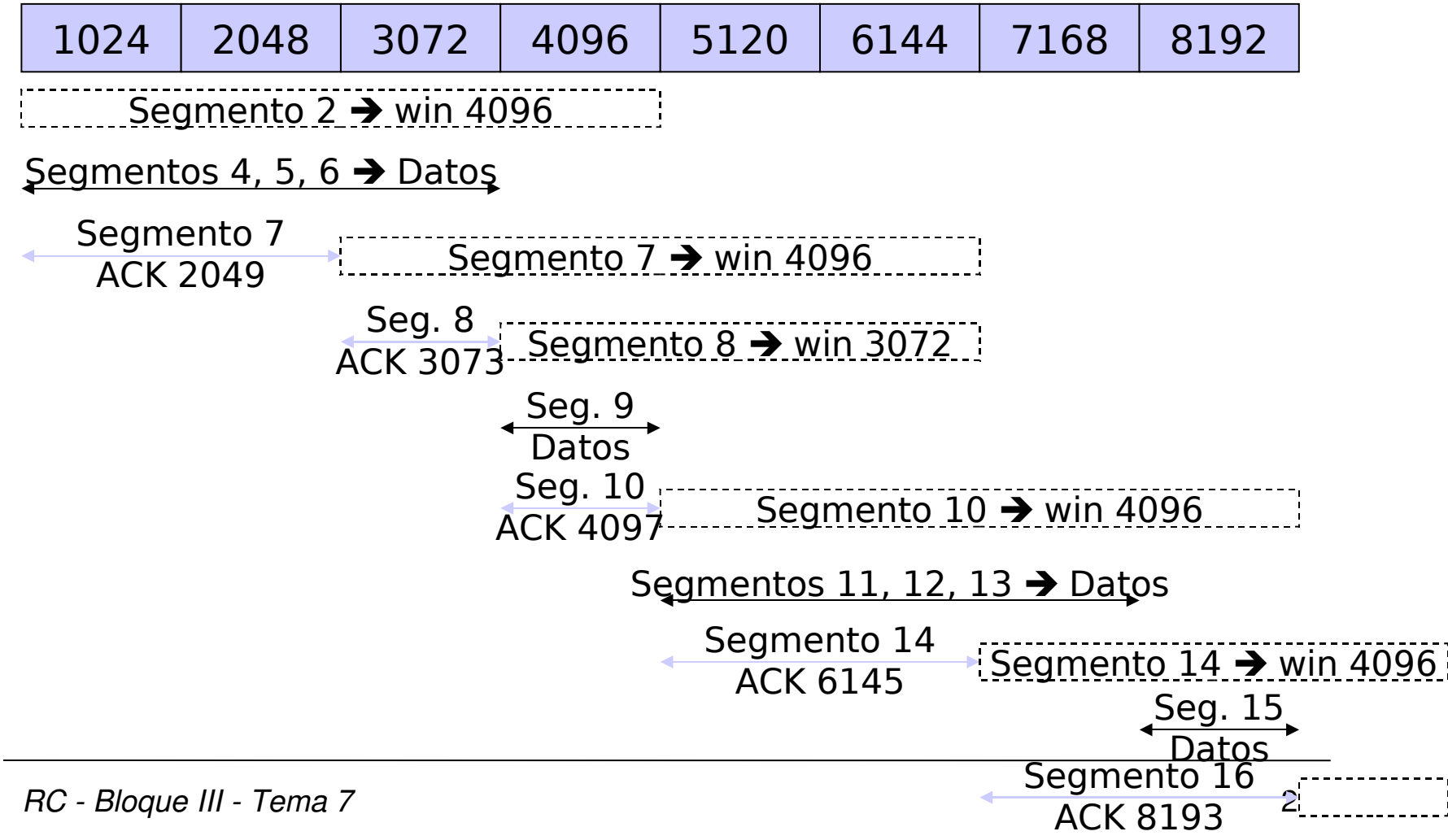


Control de flujo

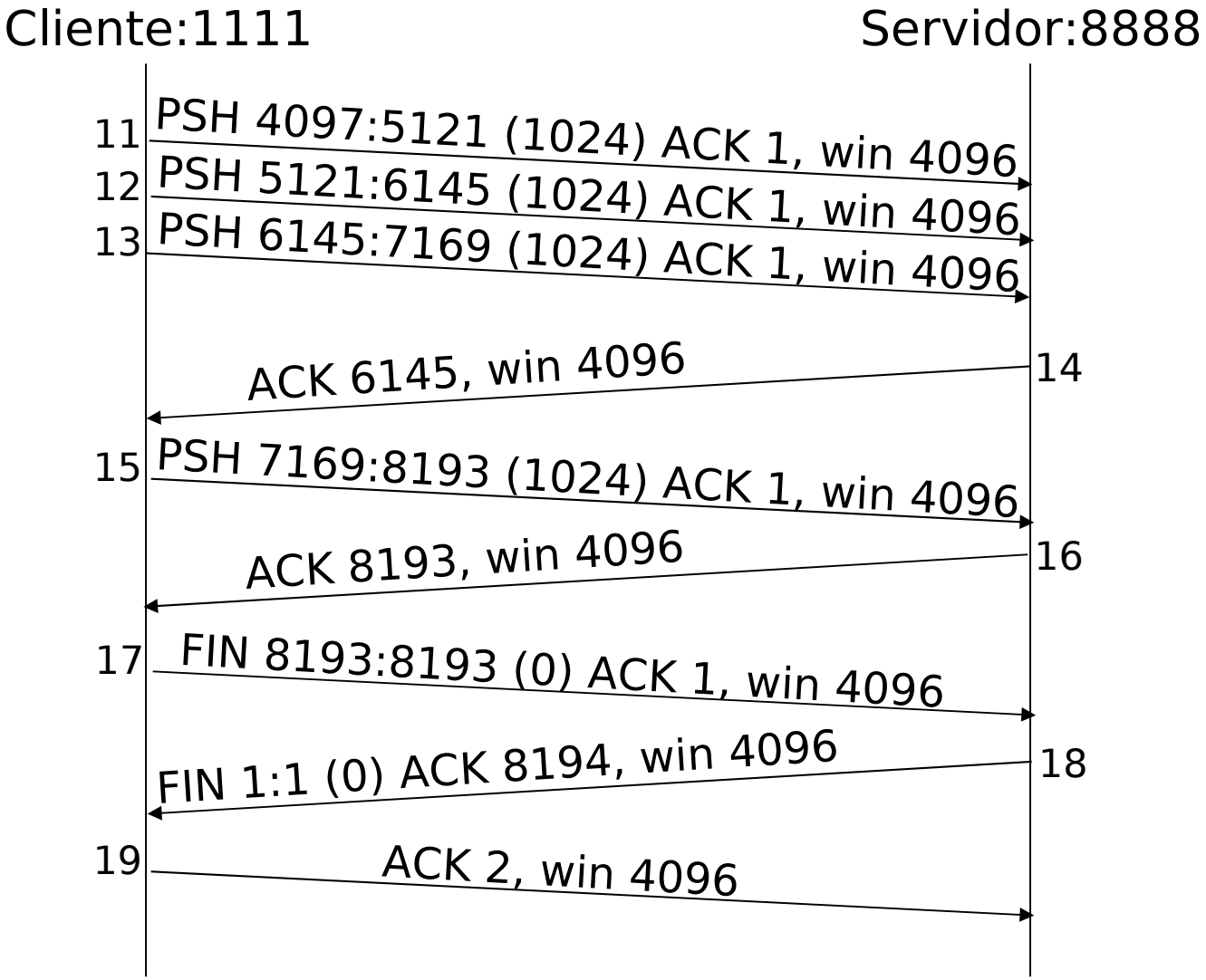


Control de flujo

- Movimientos de ventana sobre el ejemplo de control de flujo:



Control de flujo



Control de flujo

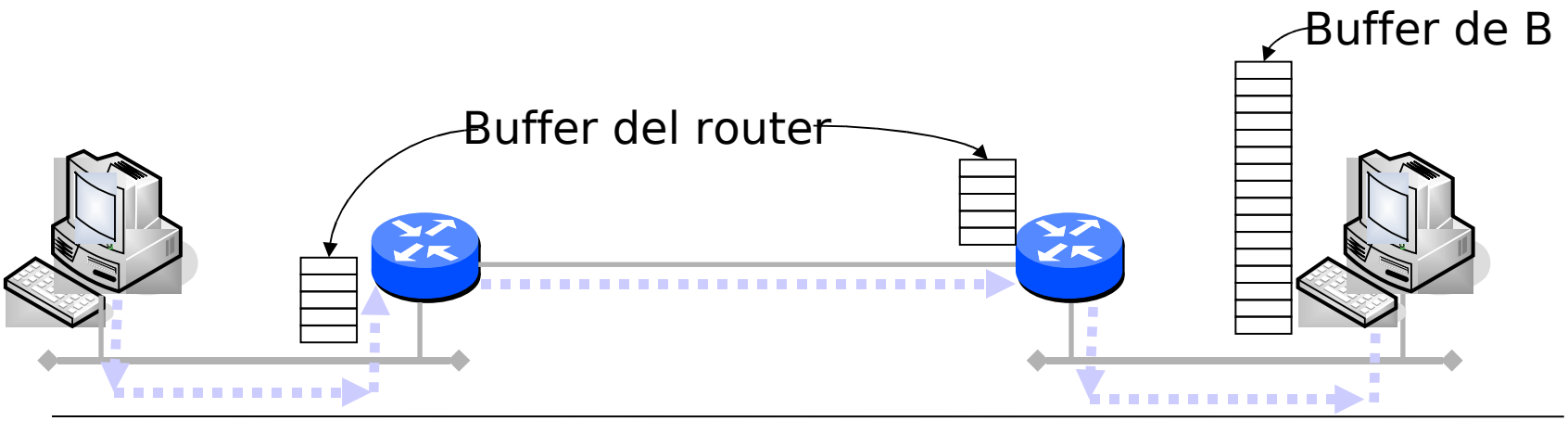
- Conclusiones sobre el ejemplo de ventana deslizante:
  - El emisor no tiene por que enviar una ventana completa de datos.
  - Cada ACK recibido desplaza la ventana hacia la derecha (el tamaño de ventana es relativo al ACK recibido).
  - El tamaño de ventana puede descender, pero el extremo de la derecha no debe desplazarse a la izquierda.
  - El receptor no tiene que esperar a que se llene la ventana para enviar un ACK.

Síndrome de la ventana tonta

- El control de flujo de TCP puede provocar que se intercambien pequeñas cantidades de datos, en lugar de esperar y enviar un segmento completo.
- Causas:
  - La aplicación receptora consume los datos muy lentamente
  - La aplicación emisora produce los datos muy lentamente
- En cualquier caso, los datos se envían con segmentos muy pequeños:
  - Uso ineficiente del ancho de banda
  - Incremento del procesamiento por parte de TCP
- **Solución de Clark**: no enviar notificaciones de ventana pequeñas (p.e. 1 byte).
- En cambio, cerrar la ventana completamente hasta que:
  - Hay espacio para un segmento entero (MSS)
  - Se ha liberado la mitad del espacio del buffer del receptor (para buffers muy pequeños)

Flujo de datos no interactivo

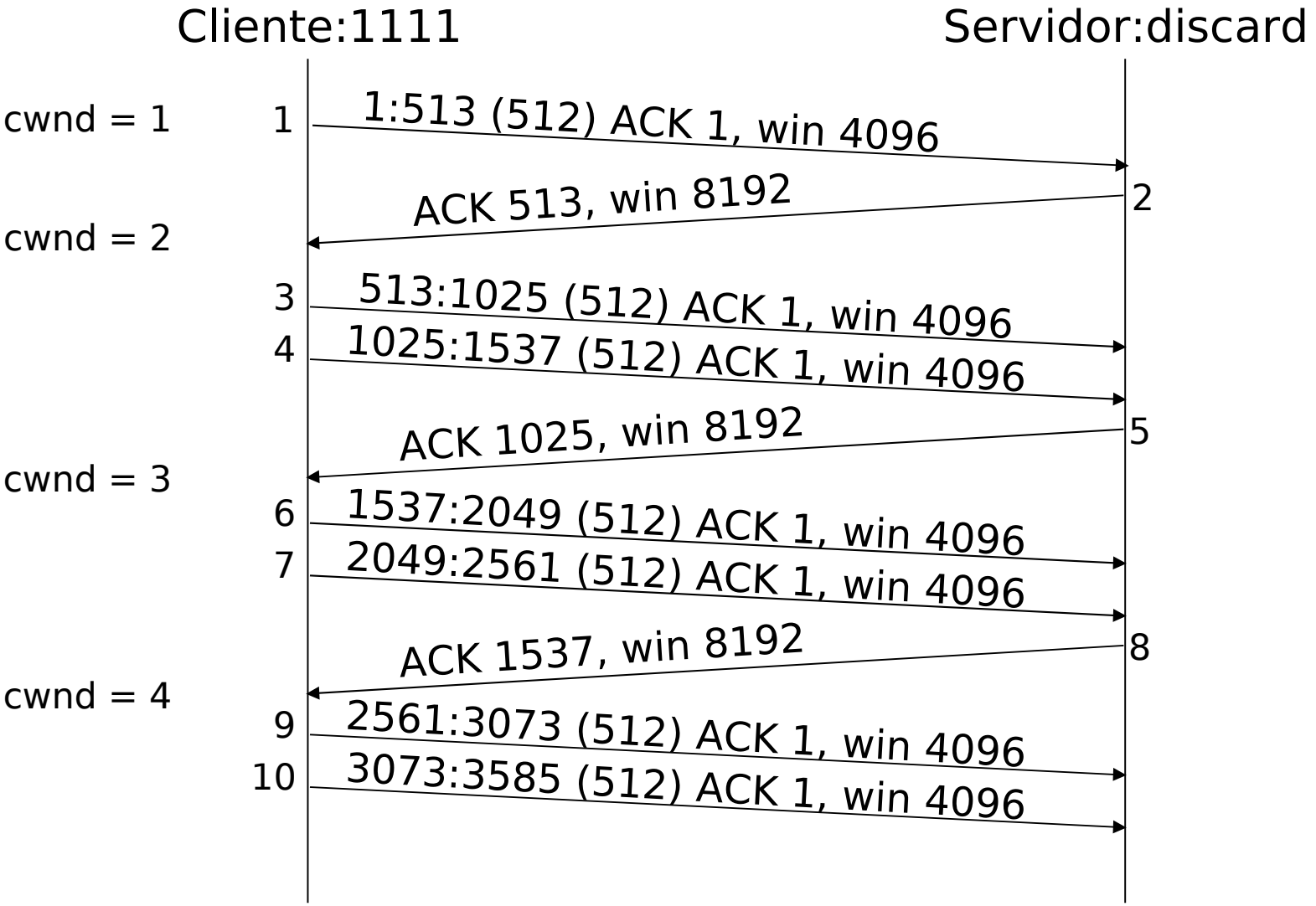
- El control de flujo evita que el emisor llegue a saturar al receptor.
- Esto es correcto en una LAN, sin embargo en un entorno con routers intermedios el control de flujo no es suficiente.
  - Los routers intermedios también se puede saturar.
- Solución: **Control de congestión** (algoritmo de inicio lento)



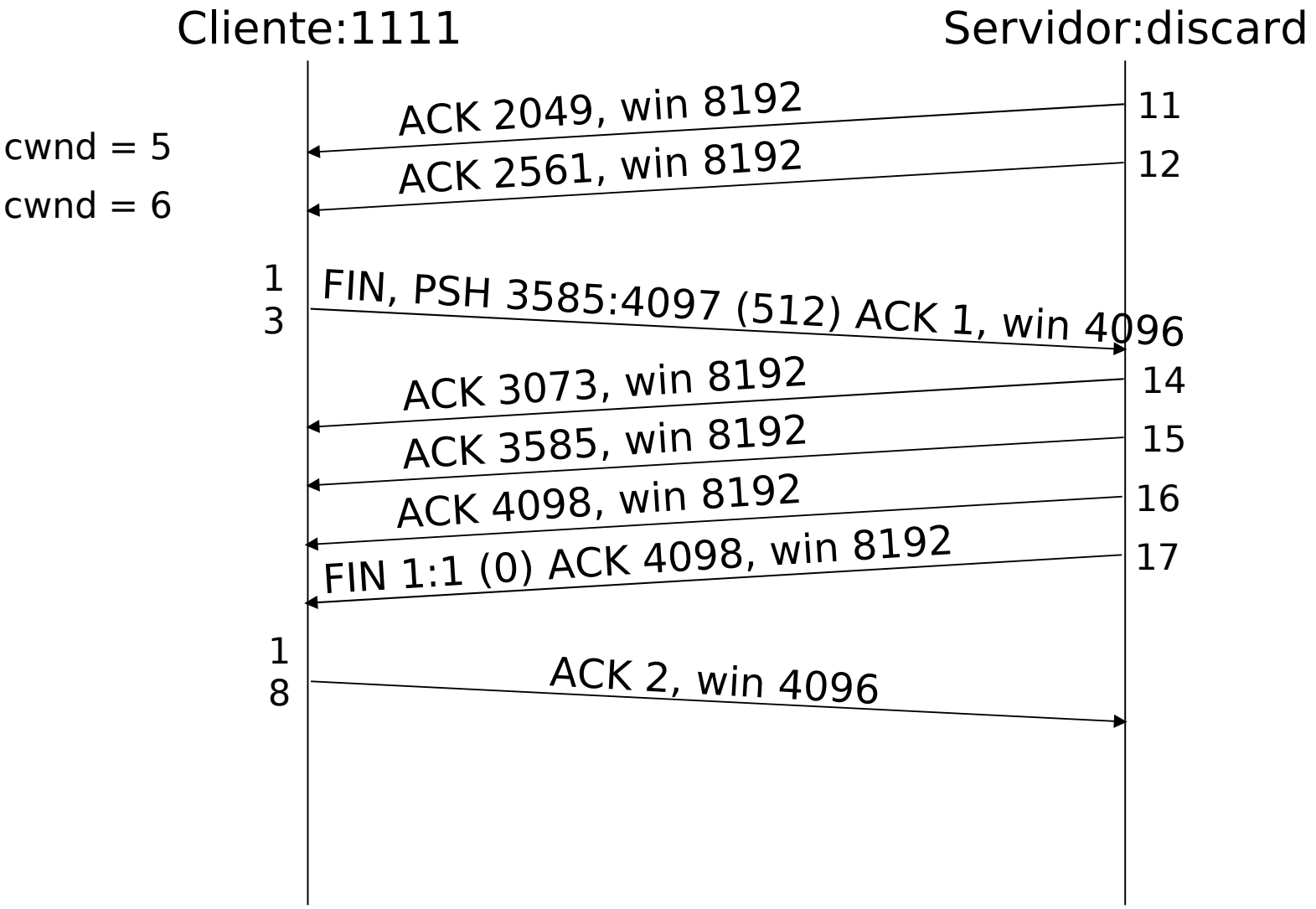
Algoritmo de inicio lento

- Algoritmo de inicio lento: "*La velocidad a la que se inyectan paquetes nuevos a la red es la velocidad a la que se reciben ACKs del otro extremo*".
- Para la implementación del algoritmo de inicio lento se trabaja con otra ventana, la ventana de congestión (cwnd), controlada únicamente por el emisor.
- El funcionamiento del algoritmo del inicio lento es muy simple:
  - Inicialmente, en el establecimiento de conexión, la ventana de congestión se inicializa a 1 segmento.
  - Cada vez que se recibe un ACK se incrementa en 1 el valor de la ventana de congestión.
  - El emisor puede enviar el mínimo entre la ventana ofrecida por el receptor y la ventana de congestión.

Algoritmo de inicio lento



Algoritmo de inicio lento



Algoritmo de inicio lento

- Segmentos de 512 bytes.
- Al recibir el segmento 2 el emisor ➔ cwnd = 2 ➔ Envía dos segmentos más.
- Al recibir el segmento 5 ➔ cwnd = 3 ➔ ¿Por qué no envía tres segmentos más?
  - Segmento 5: ACK 1025
  - Siguiente segmento a enviar: 1537:2049 ➔ Bytes enviados y pendientes de ACK: 1537-1025: 512 bytes (1 segmento)
  - Segmentos a enviar: 3 (cwnd) – 1 (enviados y pendientes ACK) = 2 segmentos