Transporte fiable
Ventana deslizante

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Arquitectura de Redes, Sistemas y Servicios
Grado en Ingeniería en Tecnologías de Telecomunicación, 2º

Stop & wait: prestaciones

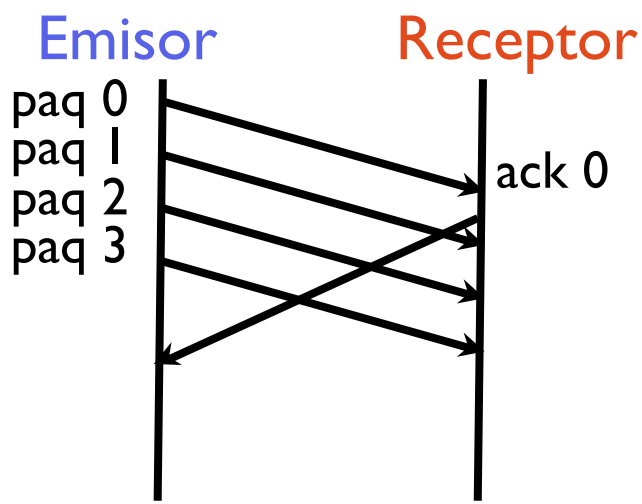
- Los protocolos de tipo stop & wait garantizan transporte fiable pero tienen poca eficiencia en enlaces en los que el RTT es grande comparado con el tiempo de transmisión de un paquete

- En el caso mejor transmitimos un paquete cada RTT por lo que el throughput de datos máximo es tamañodepaquete/RTT
- Si hay perdidas aun menos

- Puede ser aceptable en los casos en los que RTT y tiempo de transmisión del ACK es despreciable comparado con el tiempo de transmisión, pero necesitamos protocolos capaces de enviar a más velocidad sobre enlaces con velocidad de transmisión alta

Protocolos más eficientes

- Para aumentar la eficiencia, se envían varios paquetes (ventana de paquetes) mientras llega el ACK
  Varios paquetes en la red por confirmar
- **Ventana Deslizante (sliding-window)**
  - Se usan más números de secuencia que 0 y 1
  - Emisor y receptor necesitarán buffer para varios paquetes
  - Varias políticas para reaccionar a los errores
    - Go-Back N
    - Selective repeat
    - Son subtipos de sliding-window



Transporte fiable
Ventana deslizante : Go back-N

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Arquitectura de Redes, Sistemas y Servicios
Grado en Ingeniería en Tecnologías de Telecomunicación, 2º

Ventana deslizante : Go back-N

- Empezando por lo más simple:
- Número de secuencia en el paquete
- Se permite una ventana de N paquetes sin confirmar
- **Cada ACK confirma todos los paquetes anteriores (cumulative ACK)**
- Timeout al iniciar la ventana
- **Si caduca el timeout se retransmite la ventana**

- **Estado en el emisor**

  base= numero de secuencia mas bajo que aun no ha sido confirmado

  nextseqnum= siguiente numero de secuencia que voy a usar

  buffer con los paquetes enviados hasta que se confirmen y los descarte

- **Estado en el receptor**

  expectedseqnum= siguiente numero de secuencia que espero recibir

Ventana deslizante : Go back-N

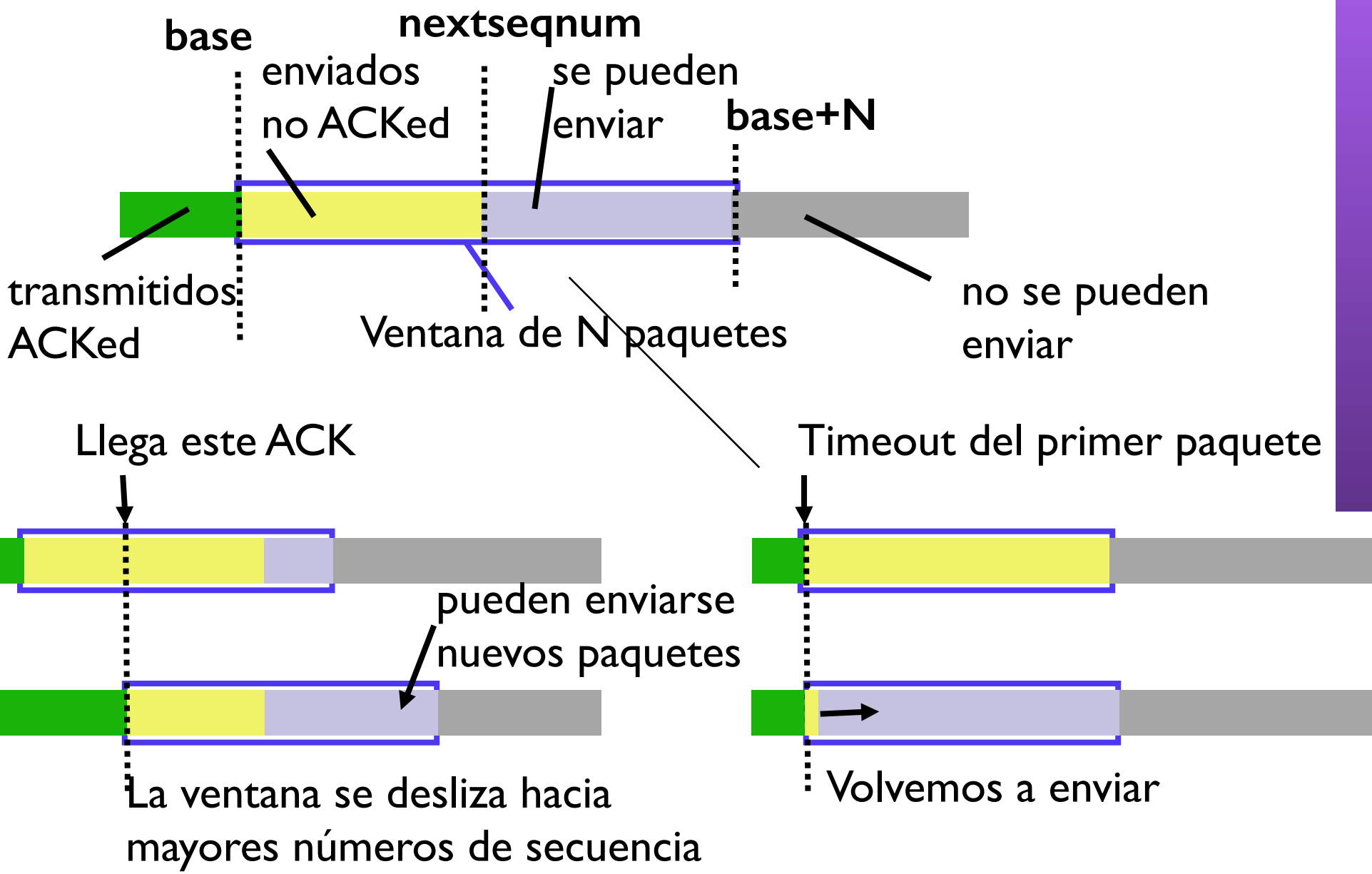
- **Emisor recibe datos de la aplicación**
  - Si puedo enviar (nextseqnum<base+N)
    - Envia el paquete
    - Si es el primero (nextseqnum==base)
      - Inicia temporizador
    - nextseqnum++
  - Si no puedo enviar (nextseqnum==base+N)
    - Rechazar el dato de la aplicación
    - La aplicación tendrá que esperar (normalmente tiene un buffer para que los paquetes esperen a que se puedan enviar)

Ventana deslizante : Go back-N

- **Emisor recibe un ACK (de ack_seq)**
  - Avanza la ventana hasta la posición confirmada (base=ack_seq)
  - Si aun quedan paquetes pendientes recalcular el temporizador
  - Al aumentar base puede enviar los siguientes paquetes que de la aplicación hasta llenar la ventana
- **Caduca el timeout del primer paquete de la ventana**
  - Reenvía todos los paquetes enviados en la ventana (desde base hasta nextseqnum-1)
  - Reinicia el temporizador

Ventana deslizante : Go back-N

Ventana deslizante

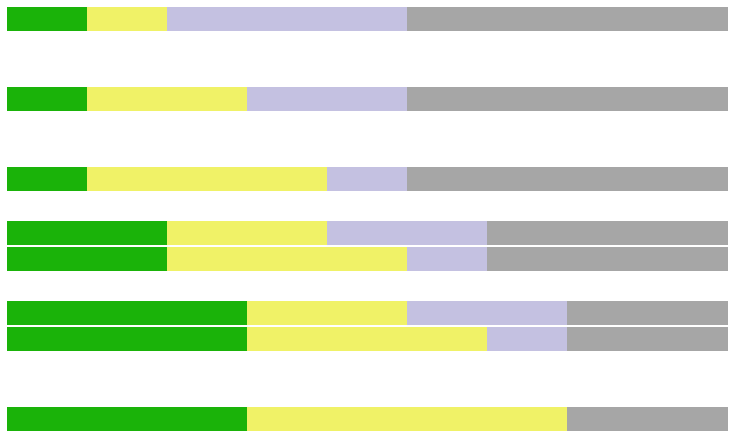


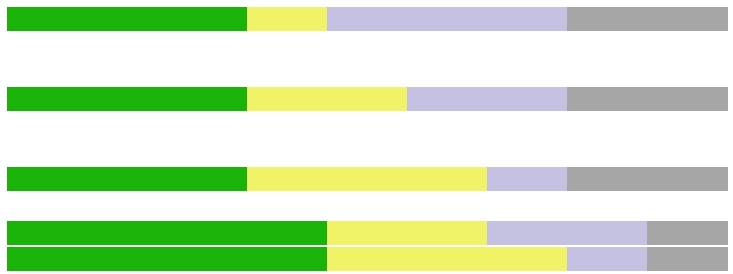
Ventana deslizante : Go back-N

- **Receptor recibe el paquete esperado**
  - Envía un ACK indicando el siguiente esperado
  - Entrega datos al nivel superior
- **Receptor recibe otro paquete**
  - Envía un ACK indicando el paquete que estoy esperando
  - Descarta los datos

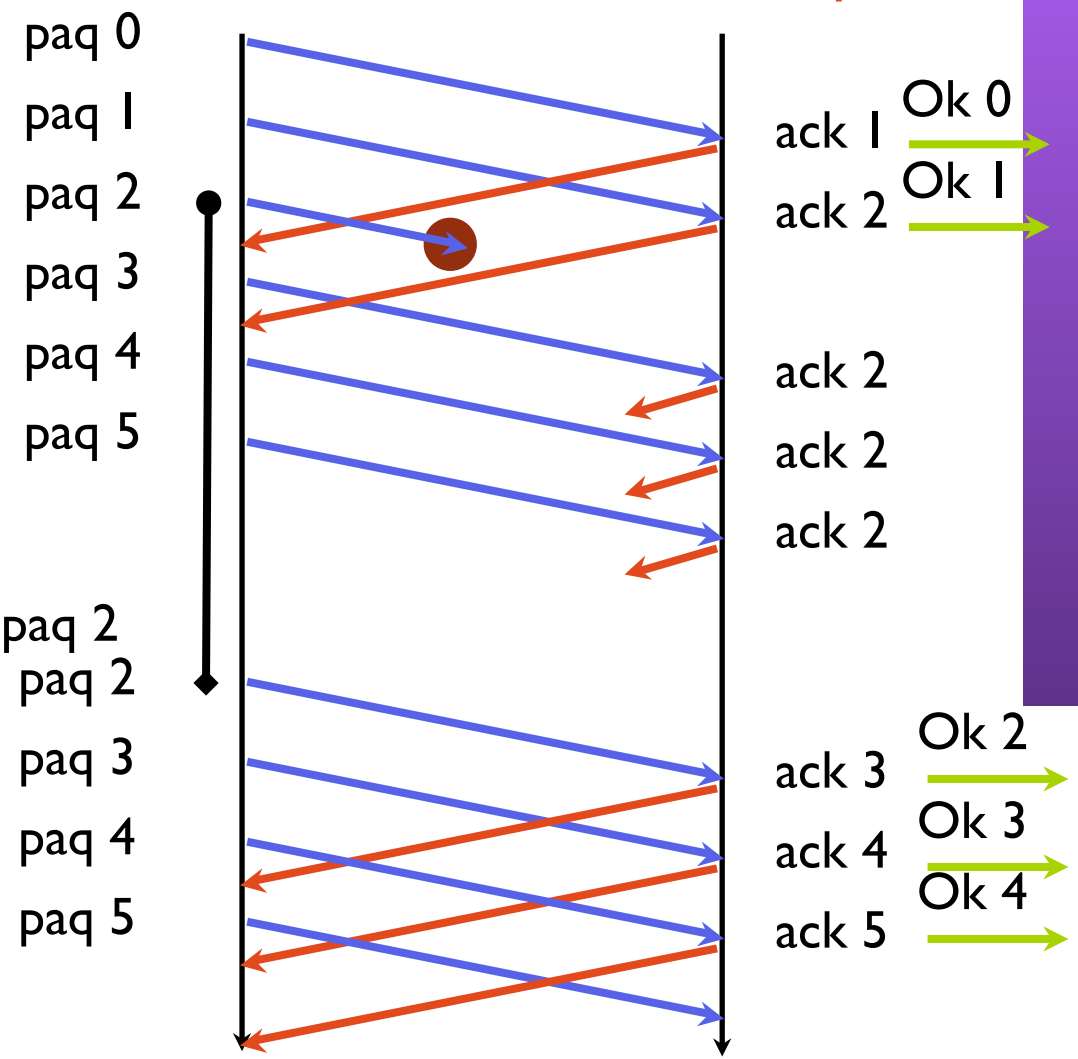
Ventana deslizante : Go back-N

Ventana de 4 paquetes





...



Ventana deslizante : Go back-N

- Simple (¿sabríais programarlo?)
- Más eficiente que stop & wait
- ¿Cuál es la velocidad máxima?
- N s / RTT ?

- Mejoras fáciles a go-back N?

Transporte fiable
Ventana deslizante : Go back-N

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Arquitectura de Redes, Sistemas y Servicios
Grado en Ingeniería en Tecnologías de Telecomunicación, 2º

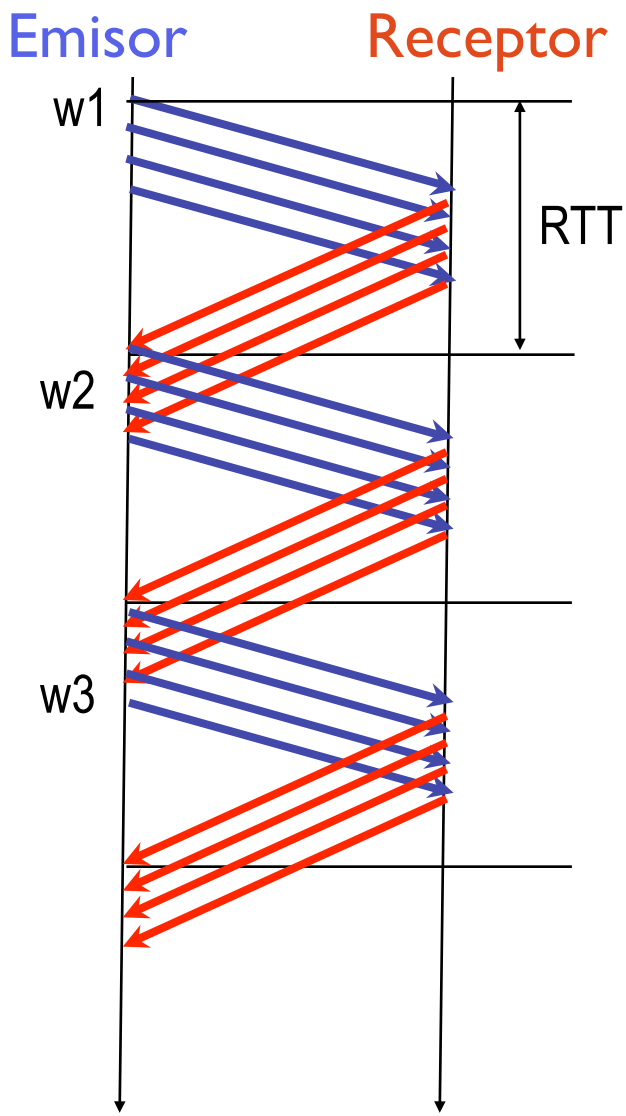
Velocidad de go-back N

- Supongamos que la aplicación envía constantemente, siempre hay datos pendientes
- Enviamos

  w=N s bytes cada RTT
- La ventana empieza a repetirse en cuanto recibimos el primer ACK, no el último
- La velocidad máxima es

  w/RTT N s/RTT

- Pero solo si la ventana se envía entera antes de que vuelva un ACK



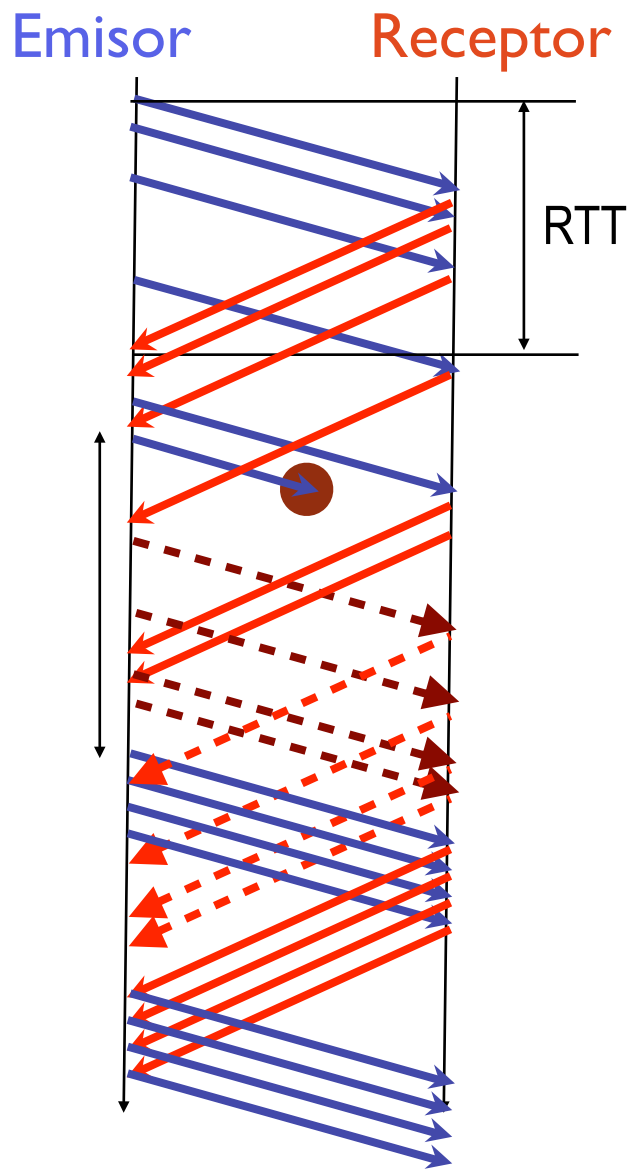
Velocidad de go-back N

- Si hay errores habrá timeouts y retransmisiones
- Si la aplicación no envía datos irá mas despacio

- w/RTT es un limite del protocolo.

  Depende del valor que elijamos para N y s

- Analizar las prestaciones de estos casos es más difícil



Velocidad de go-back N

- Puede que no de tiempo a enviar la ventana antes de que vuelva el ACK

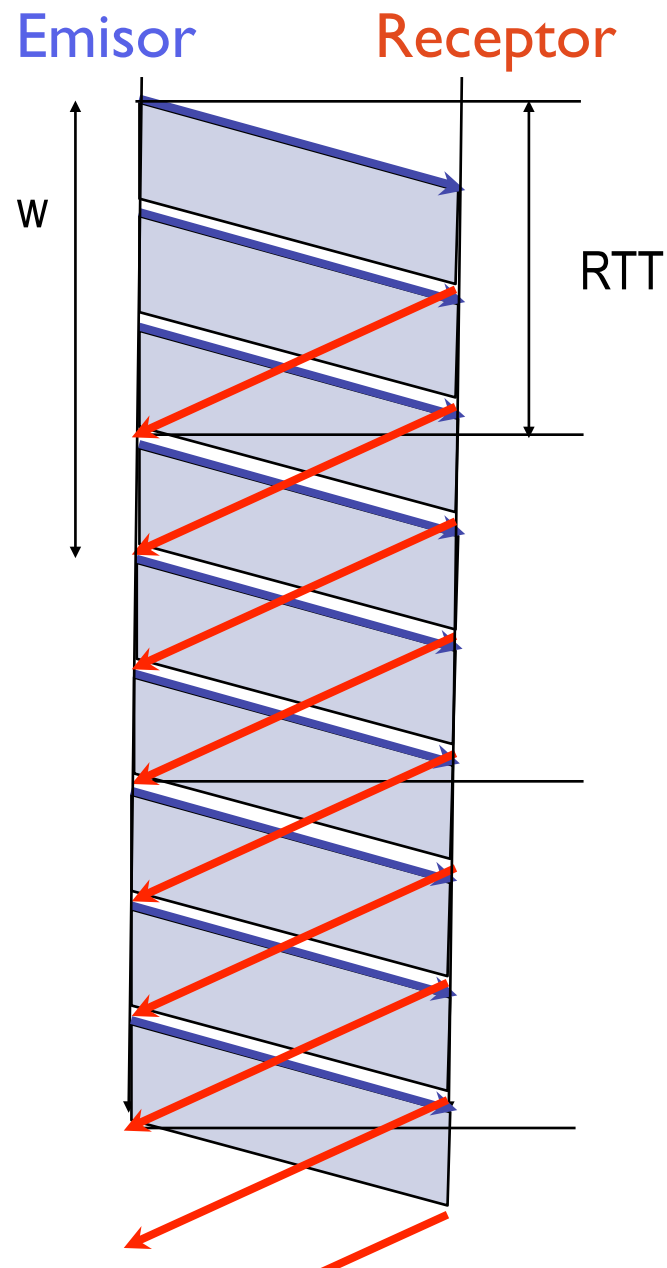
  Ejemplo con N=4

  El throughput aqui es

  3s/RTT<Ns/RTT

- El problema es que no nos da tiempo a enviar w bytes en el tiempo RTT
- Eso ocurre si

  w/C > RTT o sea si C<w/RTT
- El limite es w/RTT salvo que el canal tenga menor C (Vtx)



Velocidad de go-back N

- En resumen
- Si w < C*RTT

  El límite de velocidad lo pone el protocolo w/RTT

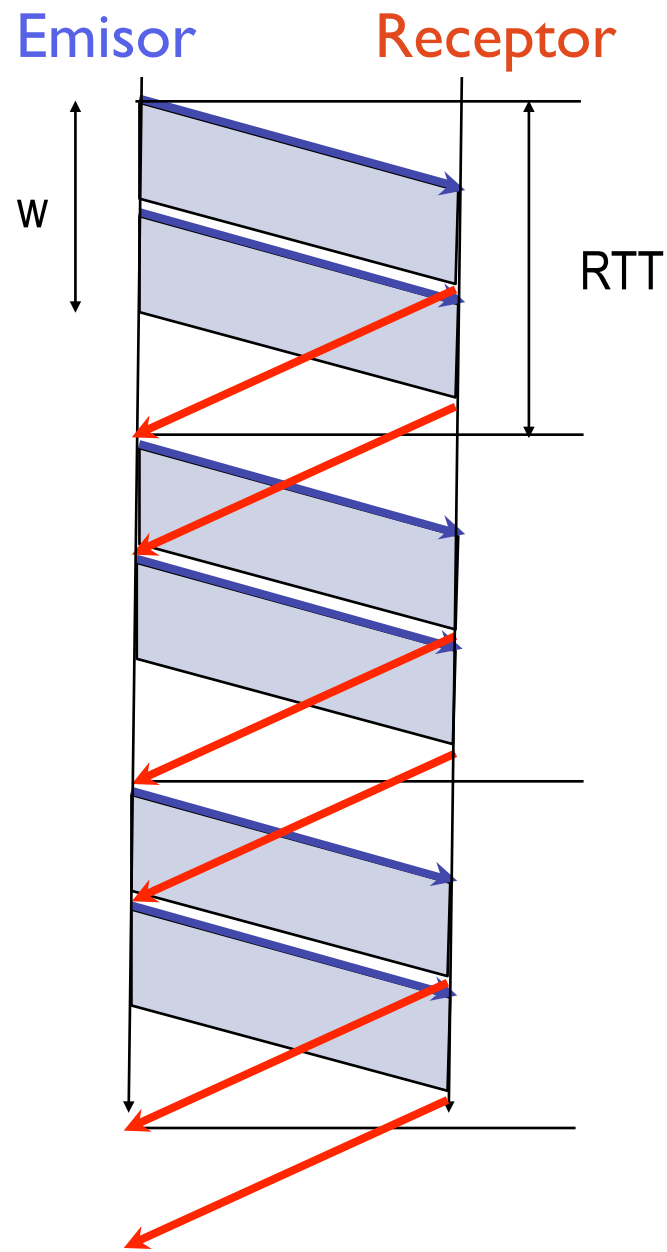
  La ventana cabe en el canal

- Si w > C*RTT

  El limite lo pone el canal

  La ventana no cabe en el canal

  Estamos aprovechando al máximo el canal



Transporte fiable
Ventana deslizante : selective-repeat

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Arquitectura de Redes, Sistemas y Servicios
Grado en Ingeniería en Tecnologías de Telecomunicación, 2º

Mejoras a Go-back N

- Aprovechar los paquetes que llegan aunque se hayan perdido los anteriores?
  - Receptor más complicado. Necesita buffer para los paquetes recibidos pero no entregados a la aplicación
  - Reenviar si se reciben ACKs duplicados?

- Confirmar/rechazar paquetes por separado

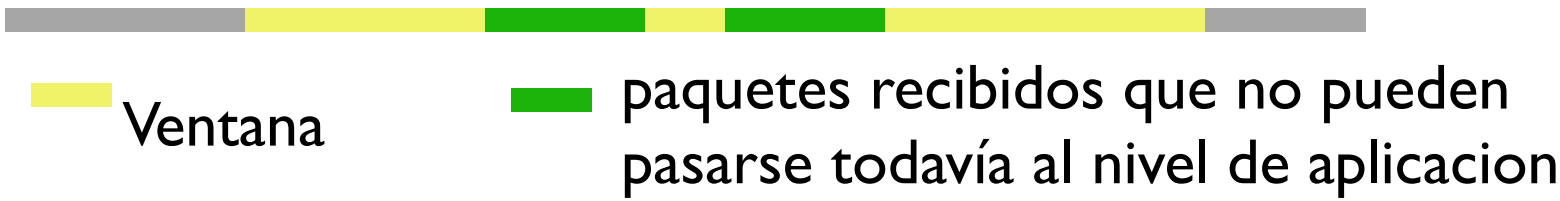
Problemas de selective repeat

- Más complejo de programar: buffer de paquetes en el receptor, array de temporizadores en el emisor...

- Normalmente para implementar ventanas deslizantes el numero de secuencia es un campo de la cabecera
  - tiene un numero de bits limitado
  - las reglas anteriores hay que programarlas con contadores que se dan la vuelta
  - En go-back N es facil porque con n bits puedo hacer go-back 2^n-1
  - Pero en selective repeat hay algun problema...

Selective Repeat

- El receptor confirma (ACK) individualmente cada paquete
  - Mantiene en buffer los paquetes recibidos a la espera de reconstruir la secuencia y pasarlos al nivel de aplicación



- Se reenvian los paquetes no confirmados por timeout
  - **Timeout individual por cada paquete**
- Ventana de N paquetes que pueden enviarse sin recibir ACK

Eventos en el emisor (SR)

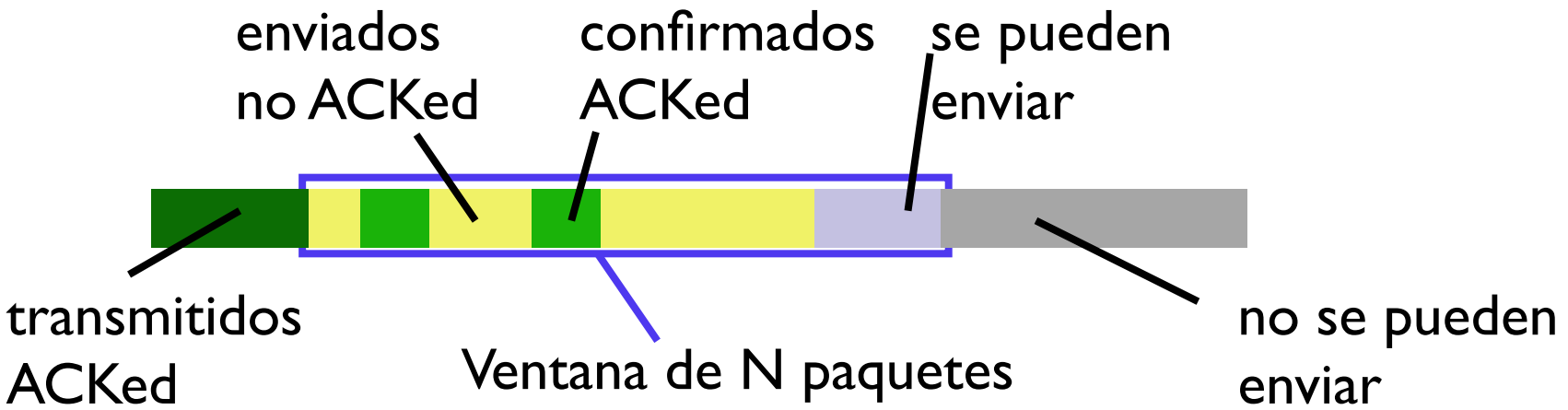
- **ACK recibido**
  - Se cancela el timeout de ese paquete
  - Si se puede avanzar la ventana se avanza hasta donde se pueda
  - Si la ventana avanza se envían paquetes nuevos si hay disponibles y se inician sus timeouts
- **Timeout de un paquete**
  - El paquete se reenvía
  - Se reinicia el timeout de ese paquete

Eventos en el receptor (SR)

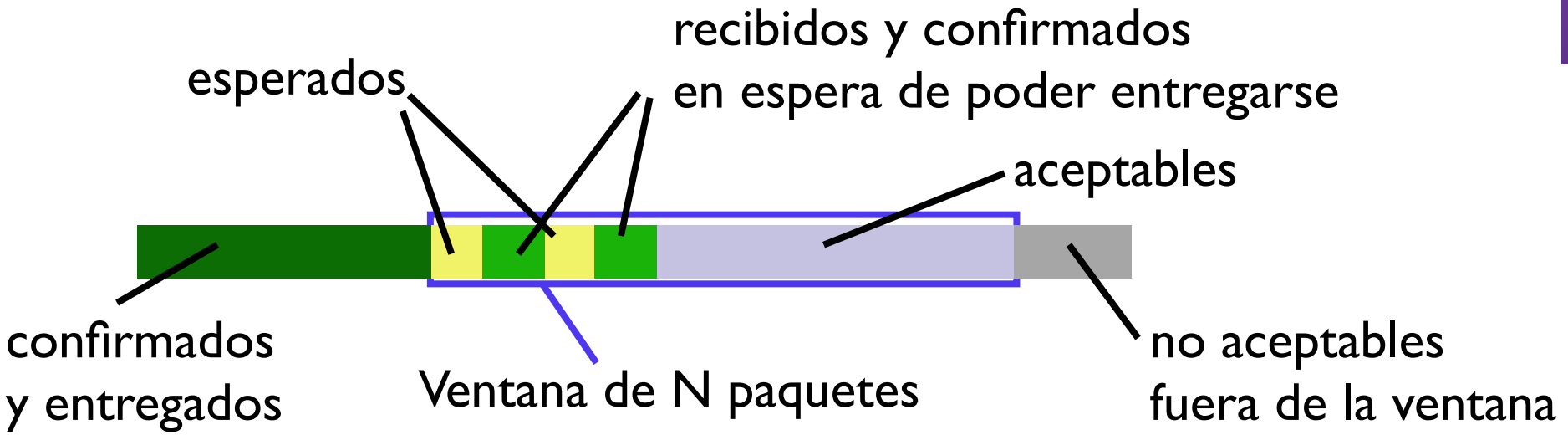
- El receptor tiene un buffer (y por tanto ventana) limitada
- **Recibidos datos en la ventana**
  - Se envía ACK de ese paquete
  - Se guarda el paquete en su posicion del buffer
  - Si el paquete es el esperado se entregan todos los paquetes continuos disponibles en la ventana al nivel superior y se avanza hasta donde se pueda
- **Recibidos datos anteriores a la ventana**
  - Se ignoran los datos
  - Se envía ACK de ese paquete
- **Recibidos datos posteriores a la ventana**
  - Se ignoran y no se envía nada

Selective Repeat

- Ventana deslizante del emisor

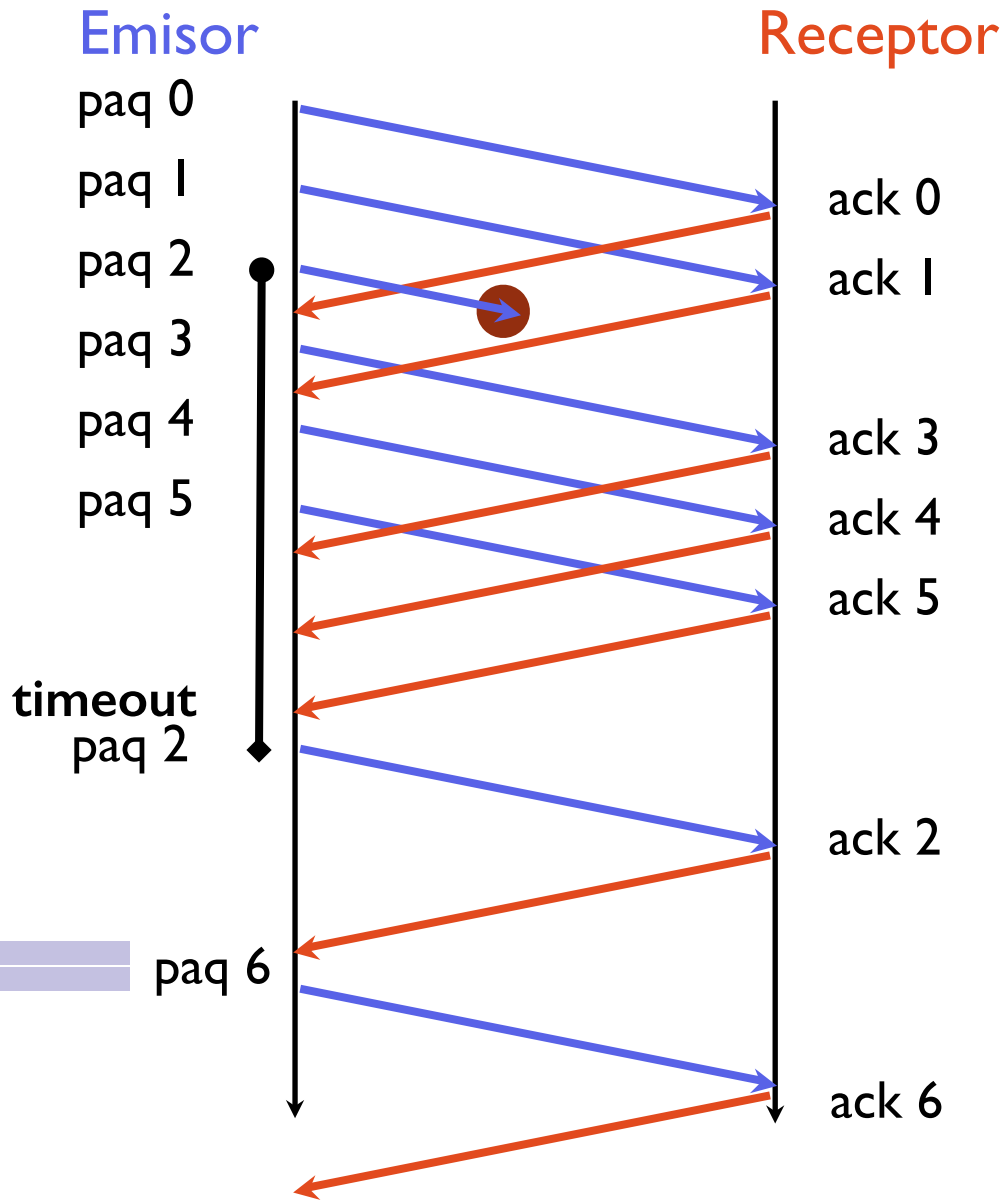


- Ventana deslizante del receptor



Selective repeat

Ventana de 4 paquetes

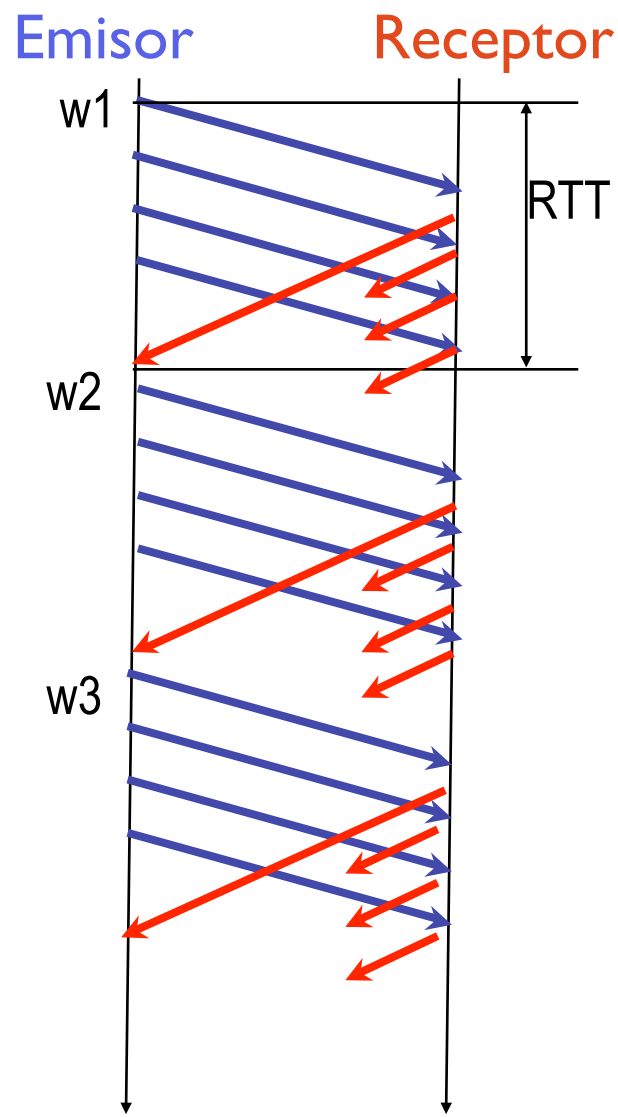


Aquí ACK no indica
el que espero recibir
sino indica el que **confirmo**

Eficiencia Selective Repeat

- Transporte fiable garantizado (en un escenario en el que se pierdan paquetes)
- Eficiencia mejor que stop-and-wait
- Parecida a la del go-back N
- El caso mejor es igual que en go-back N
- Si w>C*RTT
  - thrmax = C
- Si w<C*RTT
  - thrmax = w/RTT
- O bien
  - thrmax = min{ w/RTT, C }

- Si hay errores depende de como sean estadísticamente los errores, es más difícil de analizar



Transporte fiable en Internet

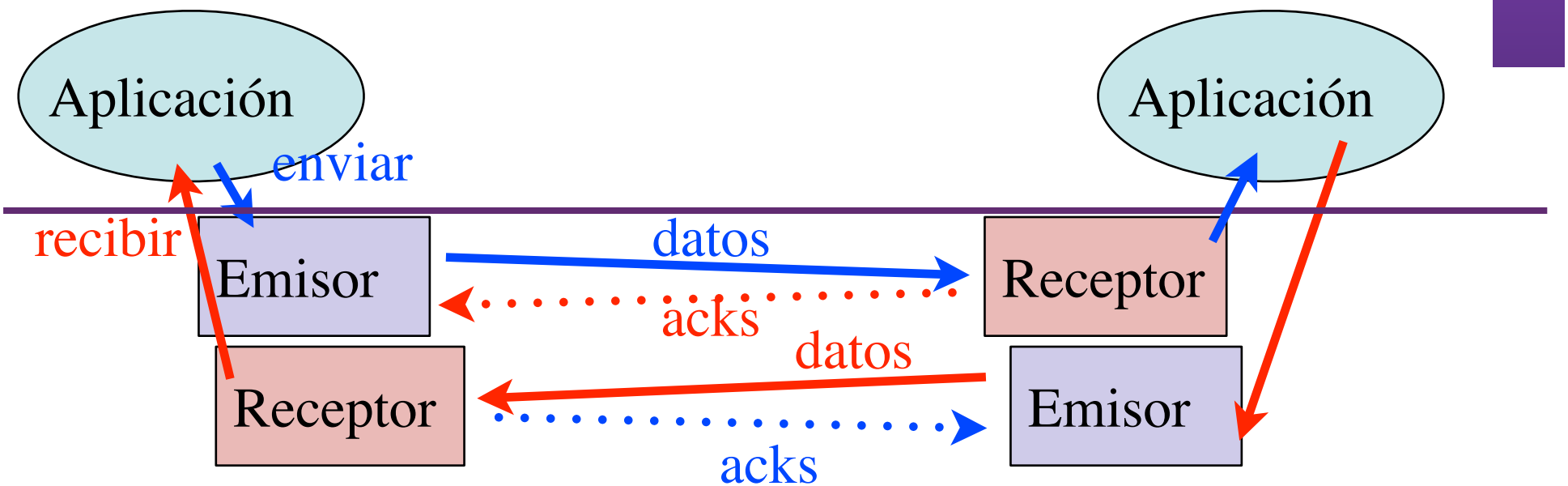
Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Arquitectura de Redes, Sistemas y Servicios
Grado en Ingeniería en Tecnologías de Telecomunicación, 2º

Sesiones/conexiones

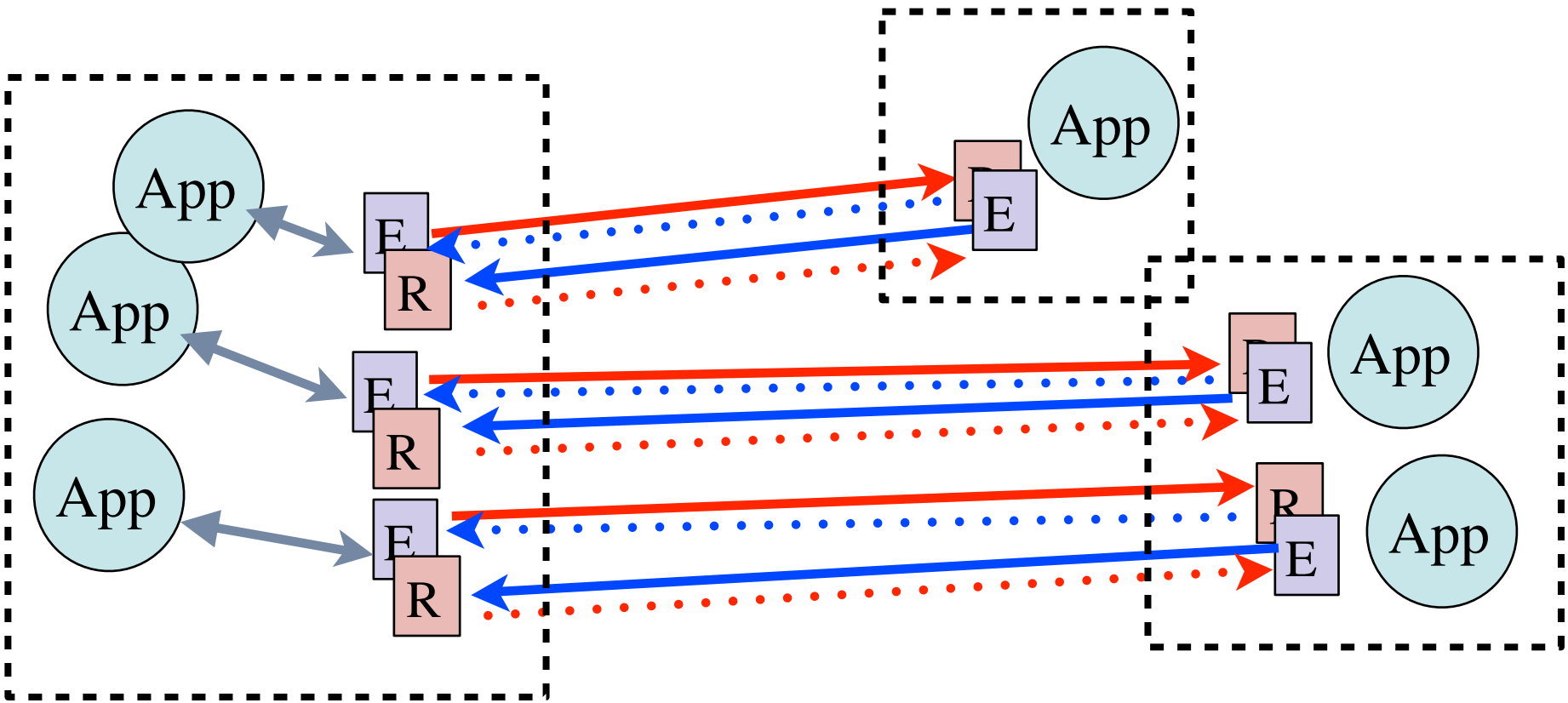
- Hasta ahora era un emisor y un receptor
- El nivel de transporte tiene que gestionar eso para diferentes sesiones de aplicación

  Los parametros de los protocolos stop&wait, go-back-N, selective repeat,
  - tienen sentido para una comunicacion entre un origen y un destino
  - una conversación son dos sentidos emisor->receptor
  - un nivel de transporte tiene que gestionar varias conversaciones a la vez



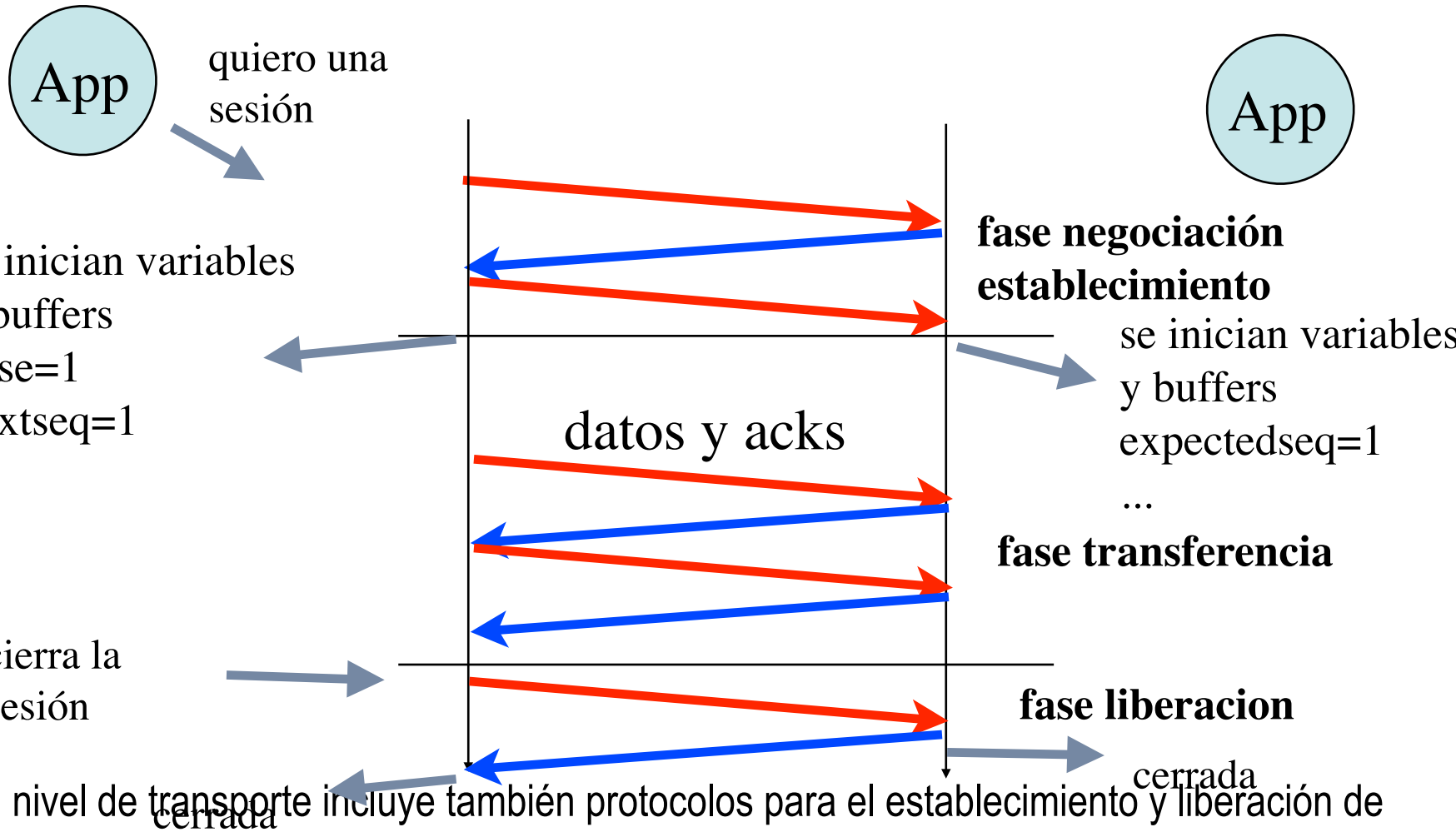
Sesiones/conexiones

- Cada pareja Emisor-Receptor necesita sus propias variables del protocolo de ventana deslizante (base, nextseq, expectedseq, bufferes de paquetes...)
- El nivel de transporte debe gestionar todo esto
- **Sesiones de nivel de transporte (se suelen llamar conexiones)**



Sesiones/conexiones

- Antes de empezar una comunicación hay que avisar para que se inicialicen las variables



- El nivel de transporte incluye también protocolos para el establecimiento y liberación de las sesiones del nivel de transporte

Conclusiones

- Hay mecanismos y protocolos que permiten conseguir un transporte fiable sobre una red no fiable
- Incluso hay mecanismos que permiten conseguir un transporte fiable y razonablemente eficiente, utilizando números de secuencia y ventanas deslizantes
  - Go-back N
  - Selective repeat
- Sus limitaciones de velocidad dependen de la ventana elegida y de su relación con los parámetros del canal (C,RTT)