Transporte fiable

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

Arquitectura de Redes, Sistemas y Servicios
3º Ingeniería de Telecomunicación

Prestaciones

- El protocolo anterior es fiable sigue siendo muy poco eficiente
- Ejemplo:
  Enlace de 1Gbps con un retardo de 15ms (4500Km), paquetes de 1000 bytes
  A que velocidad puedo enviar?
- Si los paquetes se pierden con probabilidad p

  RTT con probabilidad (1-p)

  RTT+TO con probabilidad (1-p)*p

  RTT+2TO con probabilidad $(1-p)*p^2$

  …

  RTT+n*TO con probabilidad $(1-p)*p^n$

- El timeout se procura elegir del orden del RTT
  - Mayor implica que reaccionamos despacio a los errores
  - Menor implica que se retransmiten paquetes que no hacia falta
- Las prestaciones son parecidas al anterior, pero con p probabilidad de perdida del paquete. Normalmente en Internet

  P(perdidadelpaqete) >> P(corrupciondelpaquete)

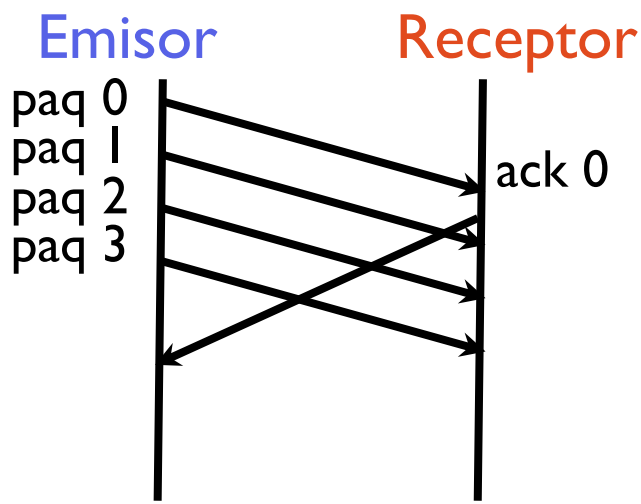
Conclusiones

- Hay mecanismos y protocolos que permiten conseguir un transporte fiable sobre una red no fiable
- Pero y las prestaciones?

  Si me bajo un fichero de 900MB por HTTP desde un servidor. El ping a ese servidor es de 60ms. Y mi acceso a Internet es de empresa a 100Mbps. Cuánto tardare como mínimo? Estoy limitado por el acceso?

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
  - 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



Go back-N

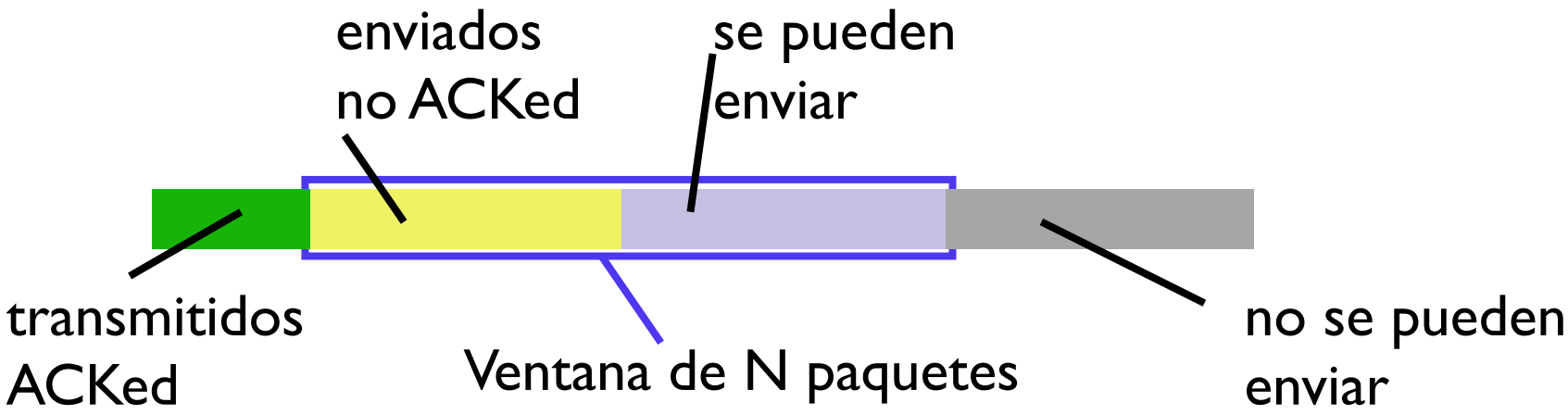
- Se utiliza número de secuencia en el paquete
- Se permite una ventana de N paquetes sin confirmar
- **Cada ACK confirma todos los paquetes anteriores (cumulative ACK)**
- Timeout al iniciar la ventana
- **Si caduca el timeout se retransmite la ventana**

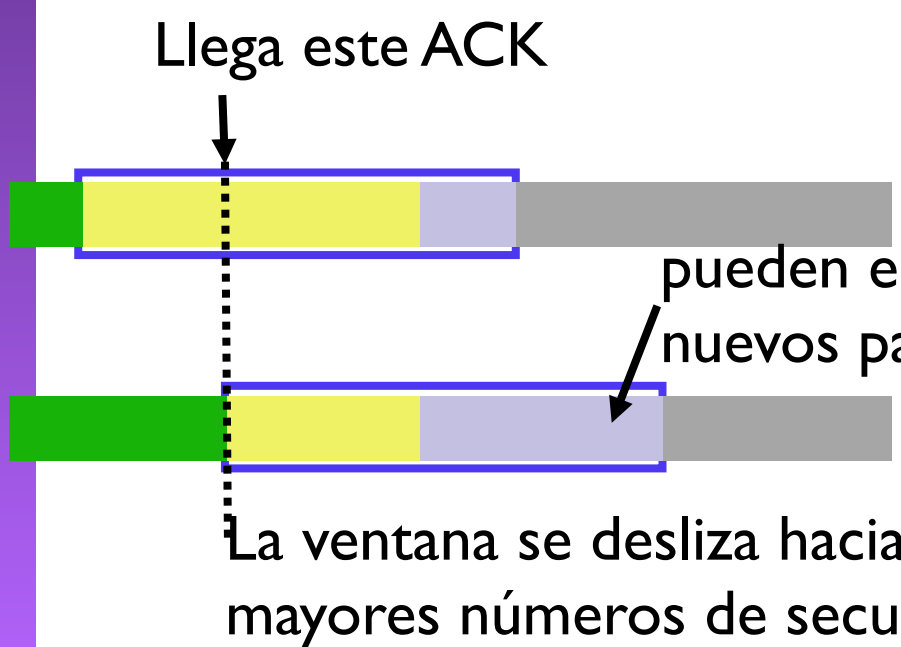
Eventos en el emisor (Go back-N)

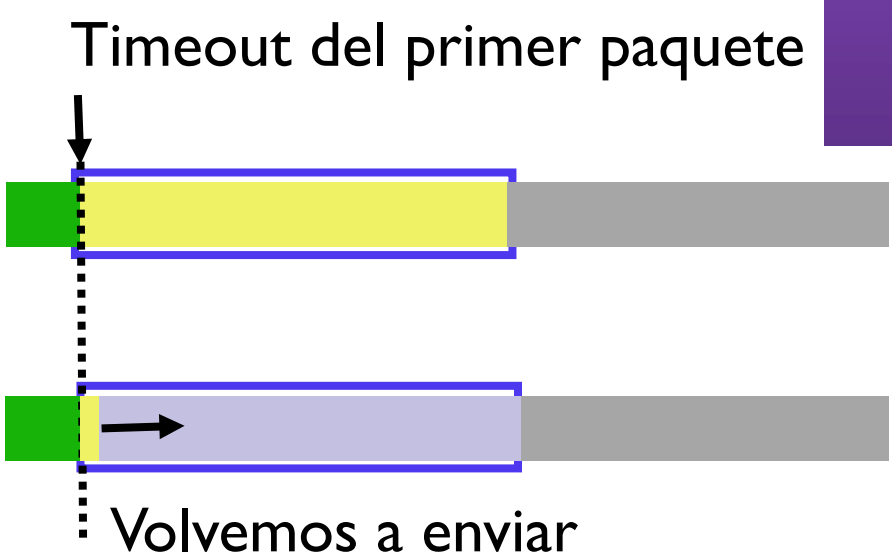
- **Recibo un ACK**
  - Avanza la ventana hasta la posición confirmada
  - Envía los siguientes paquetes hasta llenar la ventana
  - Reinicia el timeout si envías paquetes nuevos
- **Caduca el timeout del primer paquete de la ventana**
  - Reenvía todos los paquetes de la ventana
  - Reinicia el timeout

Eventos en el emisor (Go back n)

Ventana deslizante





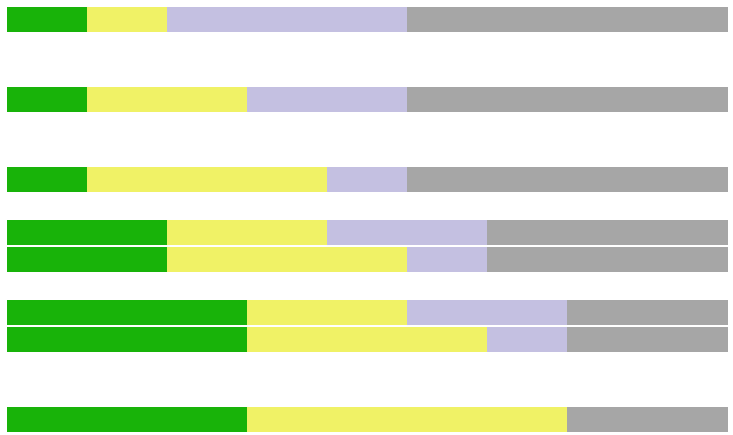


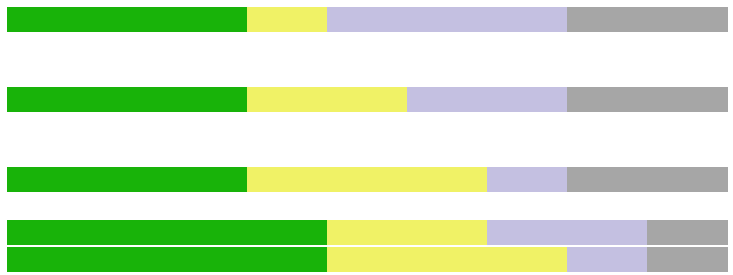
Eventos en el receptor (Go back-N)

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

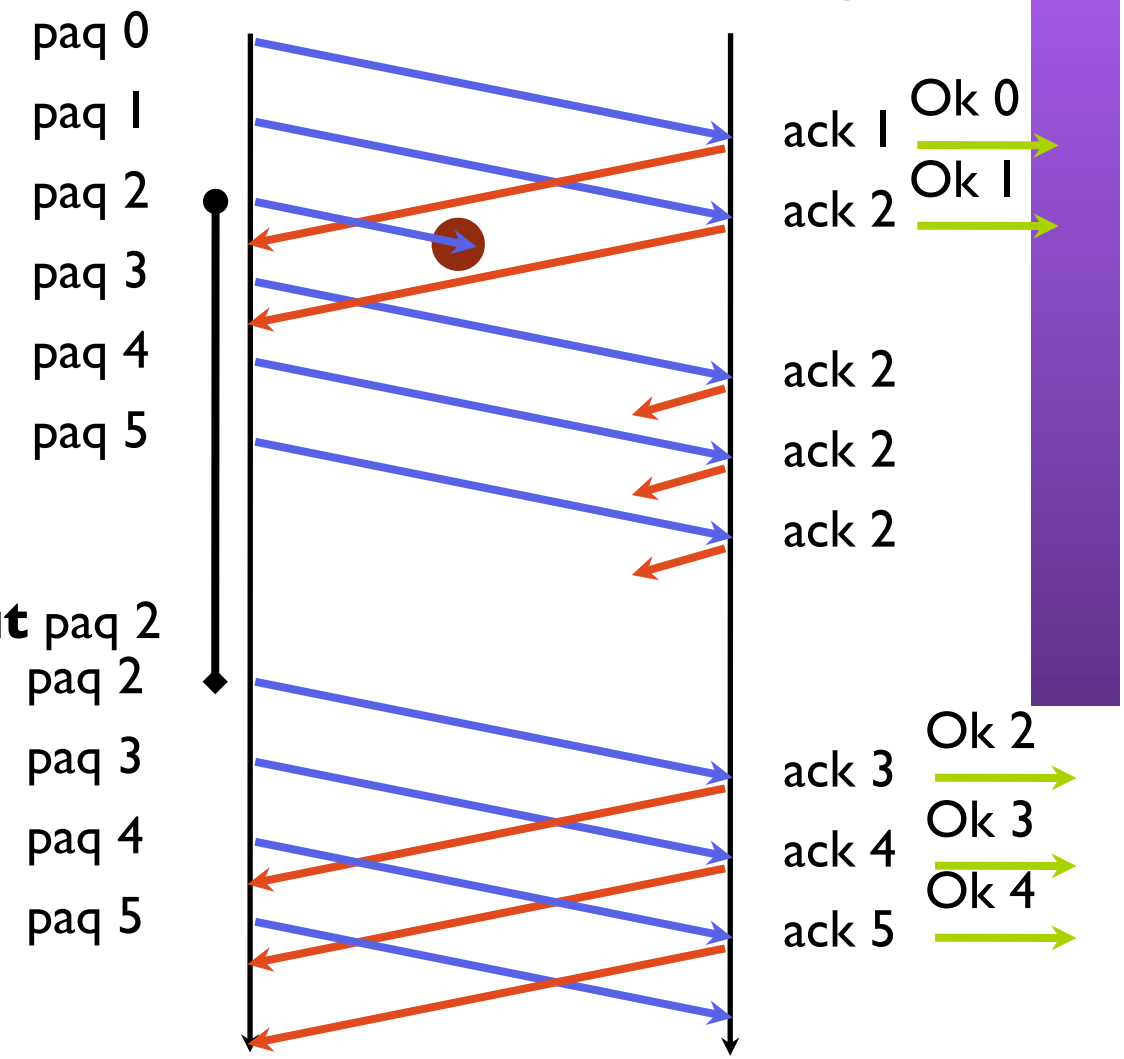
Go back-N

Ventana de 4 paquetes



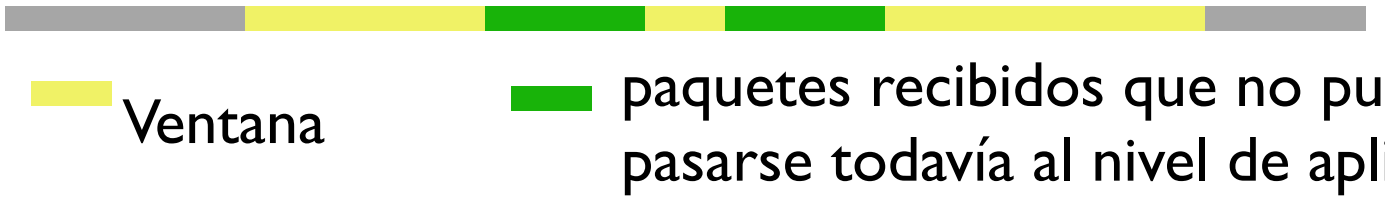


...



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)

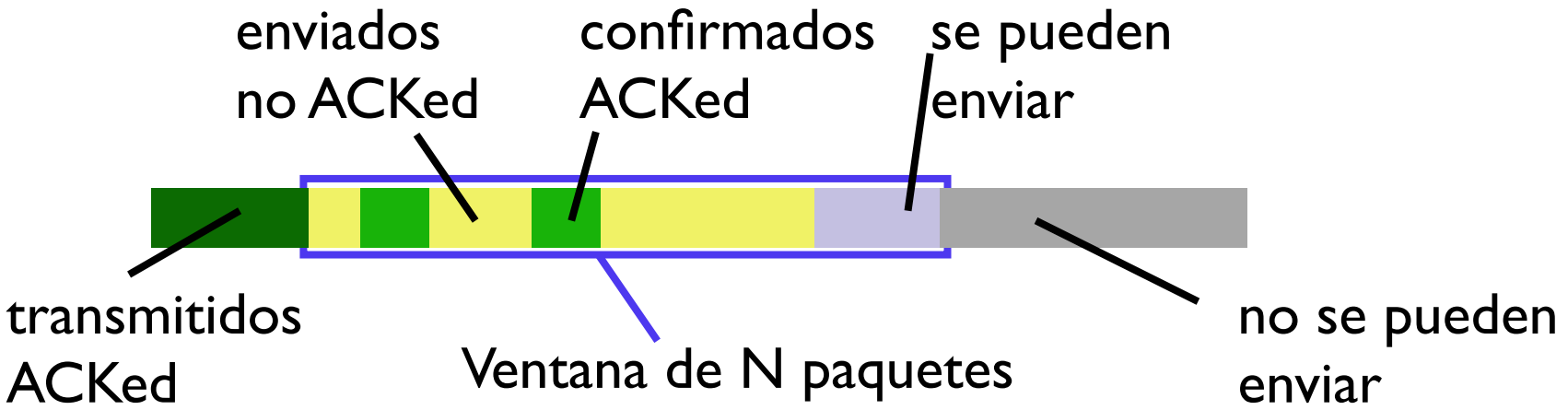
- **ACK recibido**
  - Se cancela el timeout de ese paquete
  - Si se puede avanzar la ventana se avanza hasta donde se pueda
  - Si la ventana avanza se envían paquetes nuevos si hay disponibles y se inician sus timeouts
- **Timeout de un paquete**
  - El paquete se reenvía
  - Se reinicia el timeout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

Eventos en el receptor (SR)

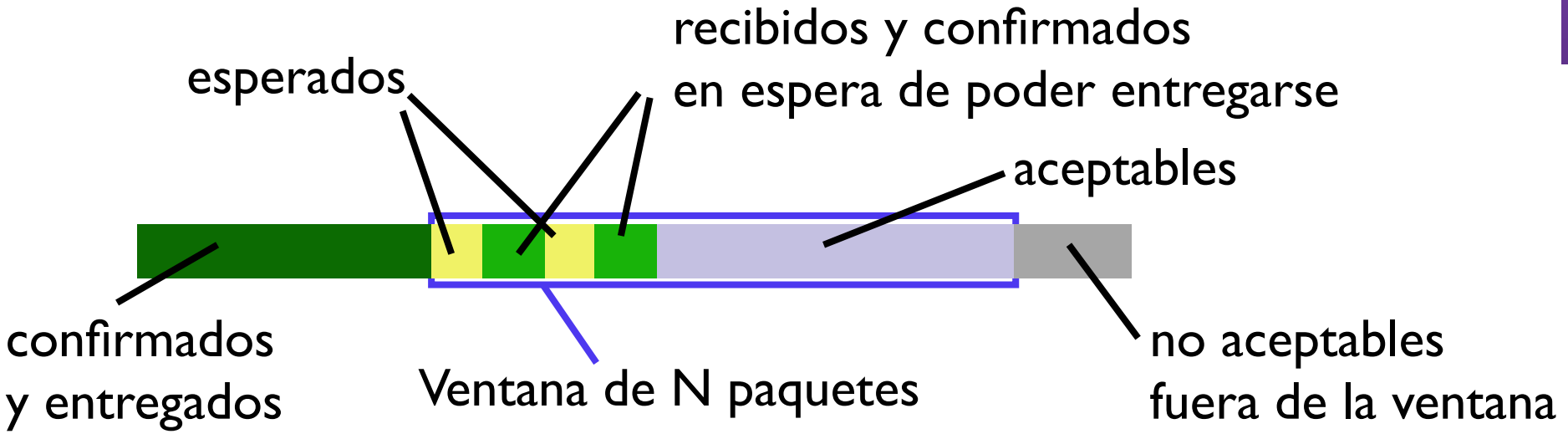
- El receptor tiene un buffer (y por tanto ventana) limitada
- **Recibidos datos en la ventana**
  - Se envía ACK de ese paquete
  - Se guarda el paquete en su posicion del buffer
  - Si el paquete es el esperado se entregan todos los paquetes continuos disponibles en la ventana al nivel superior y se 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

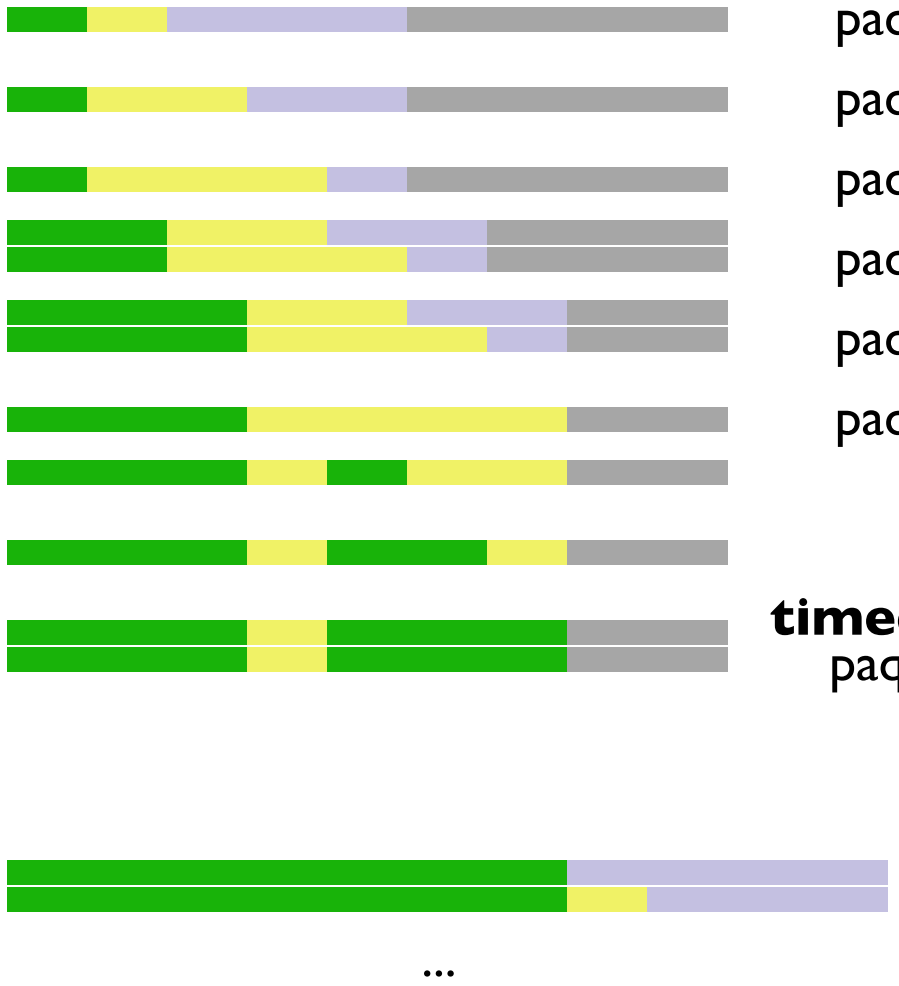


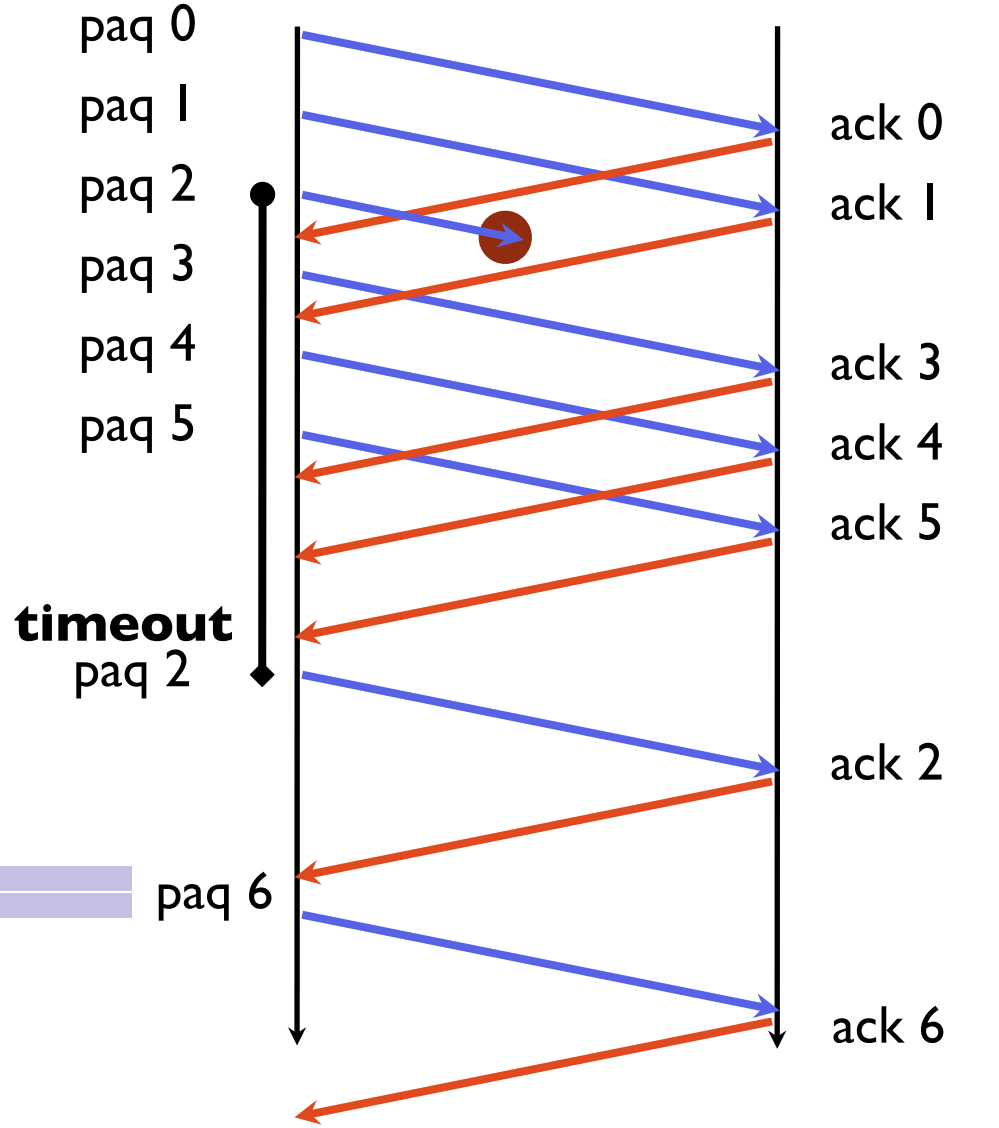
- Ventana deslizante del receptor



Selective repeat

Ventana de 4 paquetes

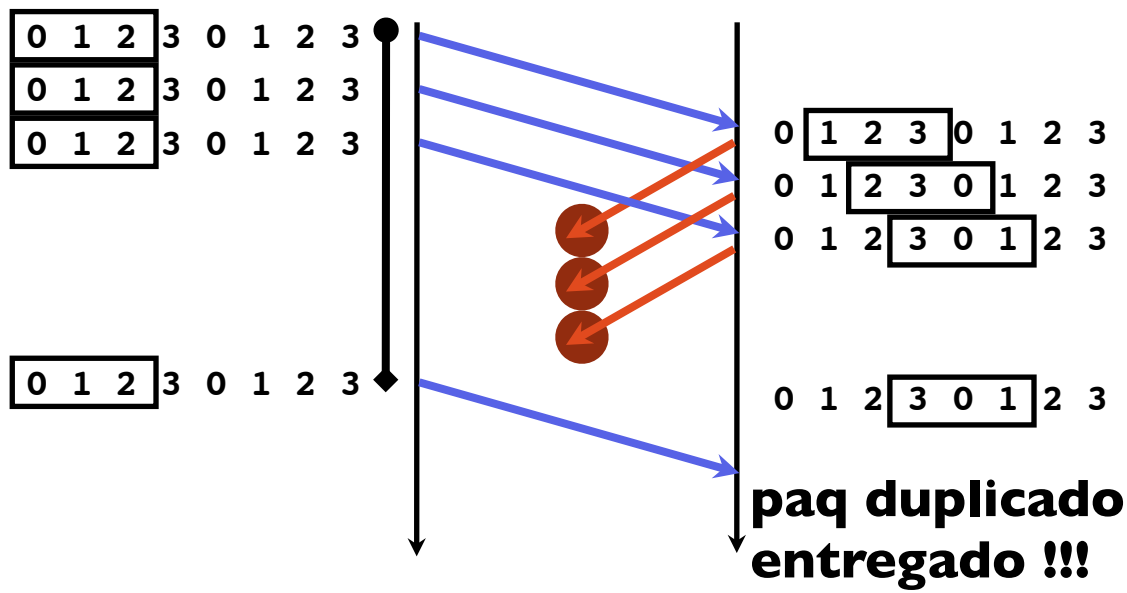


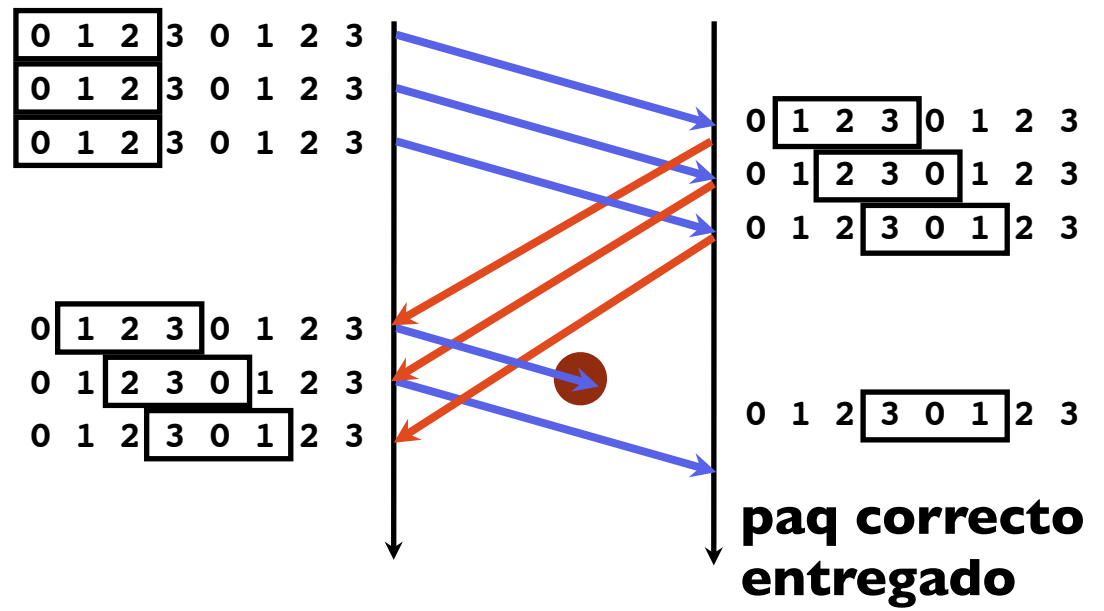


Aquí ACK no indica
el que espero recibir
sino indica el que **confirmo**

El problema del selective repeat

- Número de secuencia finito
- Ejemplo
  - seq= {0,1,2,3}
  - N=3
- El receptor no puede notar la diferencia entre los dos escenarios
- Y entrega datos duplicados como buenos
- Que relación debe haber entre #secuencia y N?





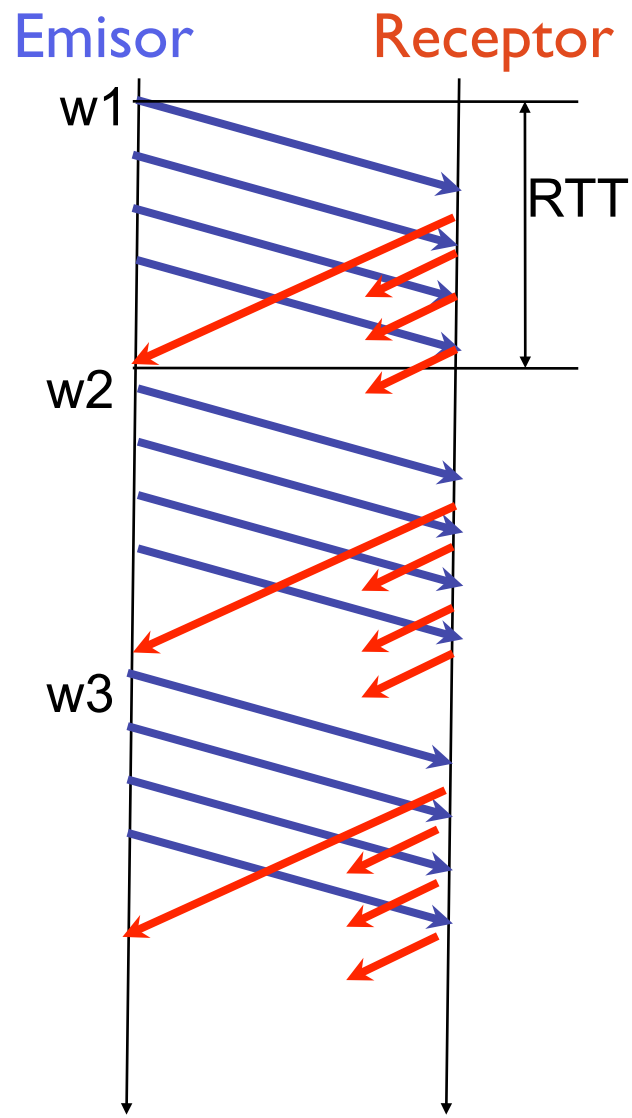
Go back-N y Selective Repeat

- Transporte fiable garantizado (en un escenario en el que se pierdan paquetes)
- Eficiencia mejor que stop-and-wait
- Cuanto mejor?
  - Análisis exacto más difícil. Aproximación tipica para caso mejor

    1 ventana cada RTT

    v=tamañoventana/RTT

- También proporcionan control de flujo dado que permiten al emisor enviar datos limitados por la ventana



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

  Si me bajo un fichero de 900MB por HTTP desde un servidor. El ping a ese servidor es de 60ms. Y mi acceso a Internet es de empresa a 100Mbps. Cuánto tardare como mínimo? Estoy limitado por el acceso?
- Estos principios serán las bases para los protocolos de transporte de Internet

Próxima clase:

- protocolo de transporte de Internet TCP