Transporte fiable

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

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

El problema del transporte fiable

- Enviar datos y asegurarme de que llegan correctamente
  - No se pierde ninguno
  - No se cambia ninguno
  - No se generan más (no hay duplicados)
  - Llegan en el mismo orden
- ¿Qué velocidades se consiguen? ¿Qué limitaciones tiene?

Red y transporte

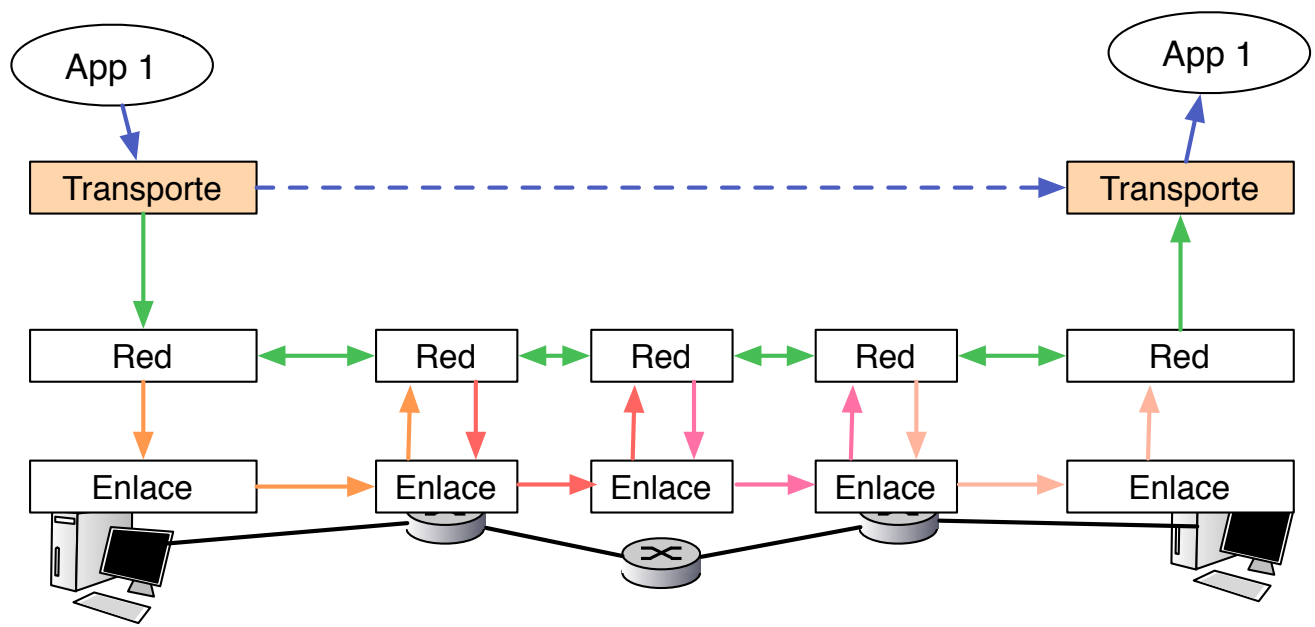
- Nivel de red: Comunicación lógica entre **hosts**

  Envía este paquete al nivel de transporte de la dirección IP a.b.c.d

  He recibido este paquete de la dirección IP x.y.z.t

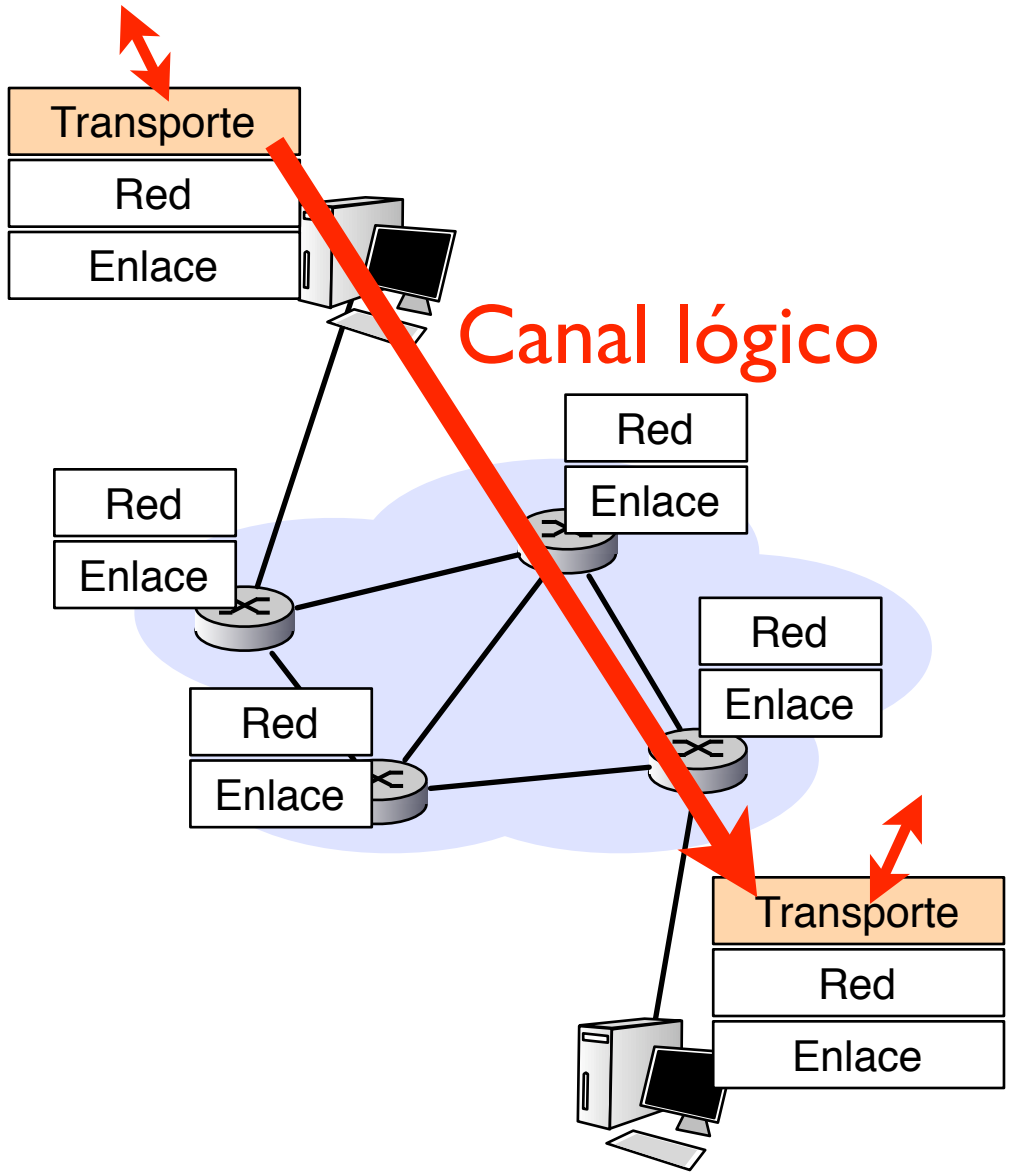
  No garantiza que todos los paquetes acaben llegando

- Nivel de transporte: Comunicación lógica entre **procesos**



Funciones del nivel de transporte

- Comunicación lógica entre aplicaciones
- Puede haber más de una aplicación en cada dirección IP
  - multiplexar aplicaciones
- Las aplicaciones quieren que todo lo que envían llegue
  - **Transporte fiable**
- Las aplicaciones ¿envían mensajes o establecen llamadas?
  - varios protocolos con interfaz de conexiones o mensajes
- No queremos saturar al receptor ni a la red
  - control de flujo y control de congestión



Transporte fiable

- Si hubiera un “Top ten” problemas de redes el transporte fiable sería un buen candidato para el primer puesto

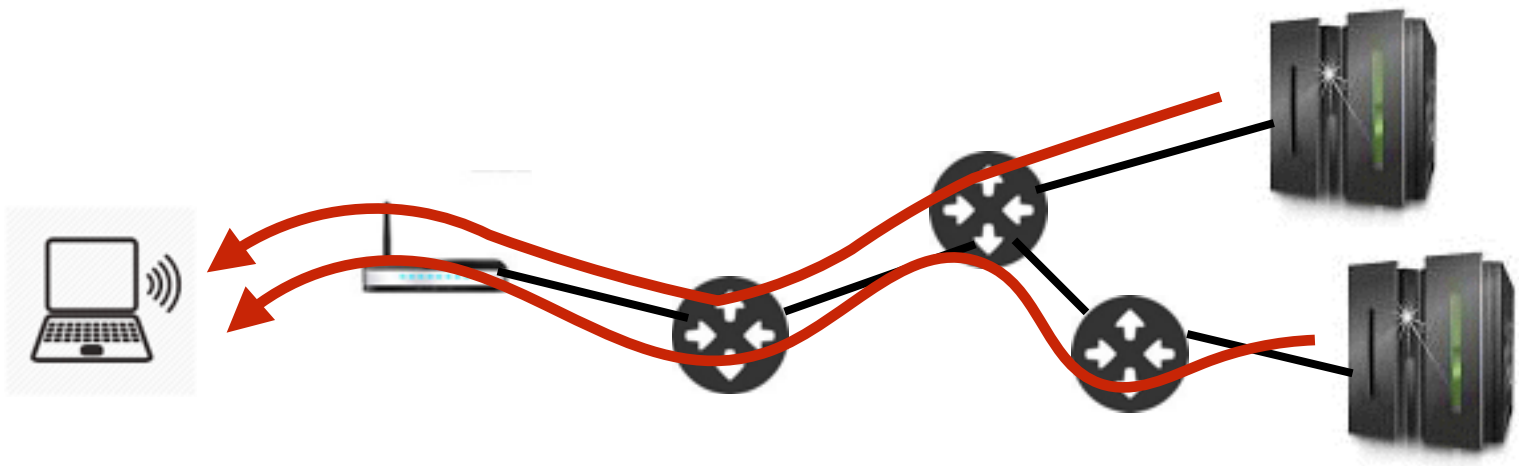
  Kurose

Ejemplo de problemas

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

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

Problema de ejemplo



- Un usuario residencial tiene un acceso de 6Mbps/512kbps
- Un ping hasta youtube da un RTT de 65ms hasta otro servidor el ping es de 83ms
- ¿A que velocidad debería bajar un fichero?
- ¿A que velocidad bajá un video de youtube?
- ¿Por qué no es a 6Mbps ?
- De que depende y como se puede optimizar

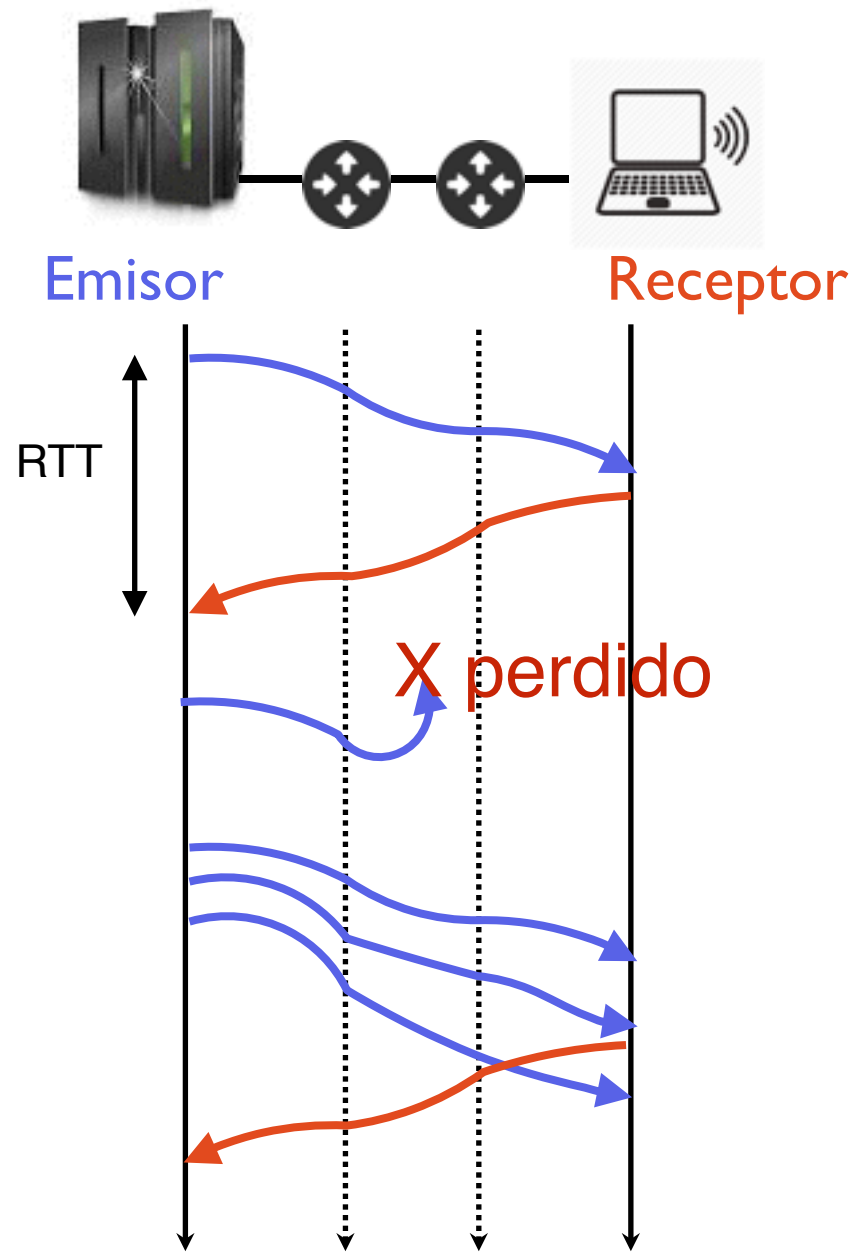
Escenario

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

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

Escenario en general

- Un origen y un destino
- Enviar una cantidad de información, más de lo que cabe en un paquete.
- RTT (round trip time) tiempo que tarda un paquete en ir y volver
- Vtx es diferente a lo largo del camino. Habra mas de un salto probablemente
- Problemas
  - paquetes que se entregan con errores
  - paquetes que se pierden

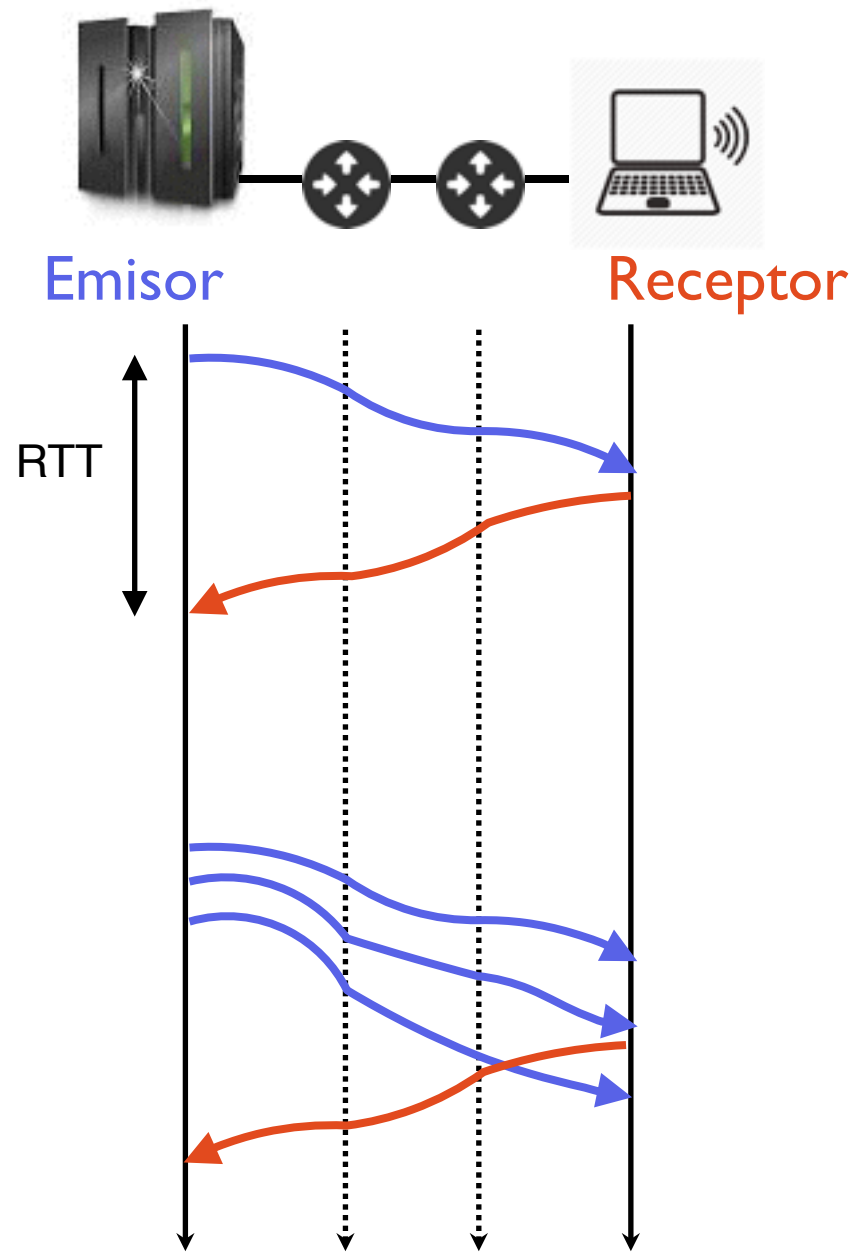


Transporte fiable?

- Puede en estas condiciones organizar una forma de envío que me garantice que llegue una información?

- Si la información es un solo paquete?
- Si la información tengo que dividirla en varios paquetes?

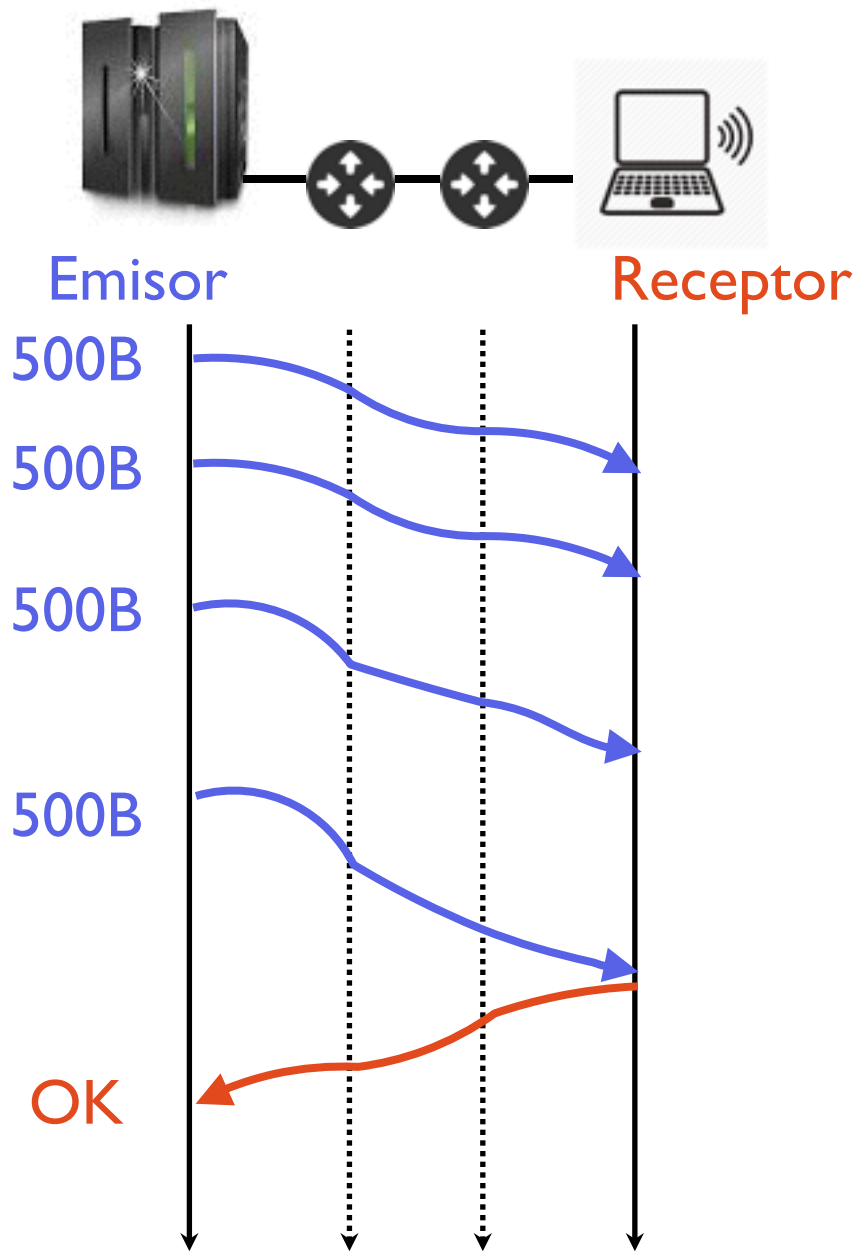
- Enviar y confirmar…



Transporte fiable?

- Midiendo el rendimiento
- Cuanto tardo en enviar toda la información que quiero enviar t

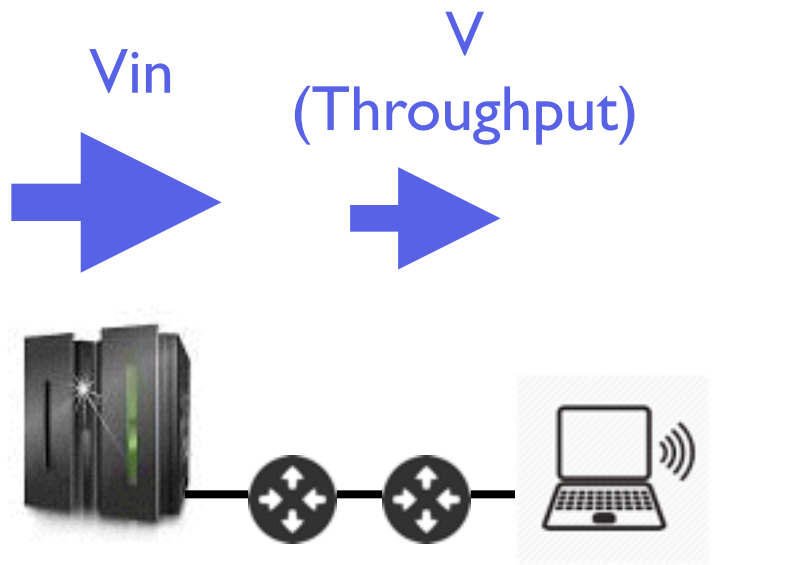
- A que velocidad media estoy enviando la información ?
- Throughput
- totalenviado/t

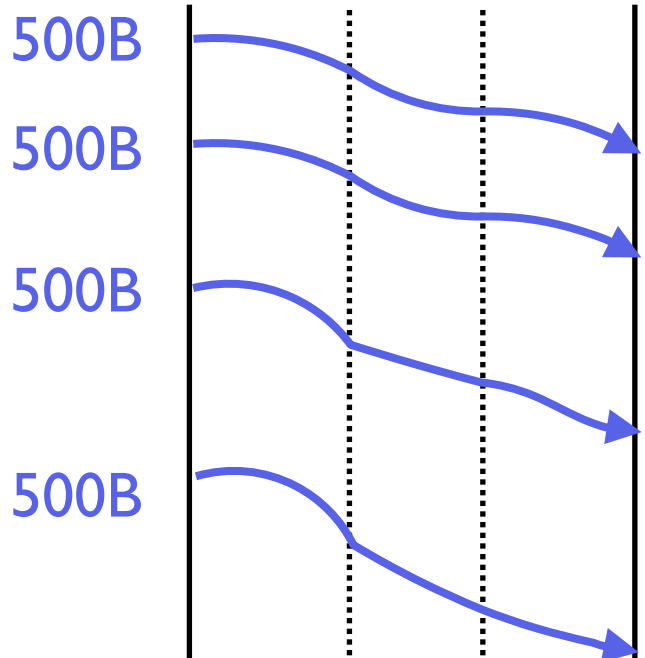


Transporte fiable?

- Y si lo que quiero enviar no es una cantidad determinada de información sino por ejemplo un video en tiempo real?
- El video generara una cantidad de información por segundo Vin

- Throughput que consigo
- Si V < Vin
  - El vídeo acabará parando





Protocolo de transporte fiable
Stop & wait

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

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

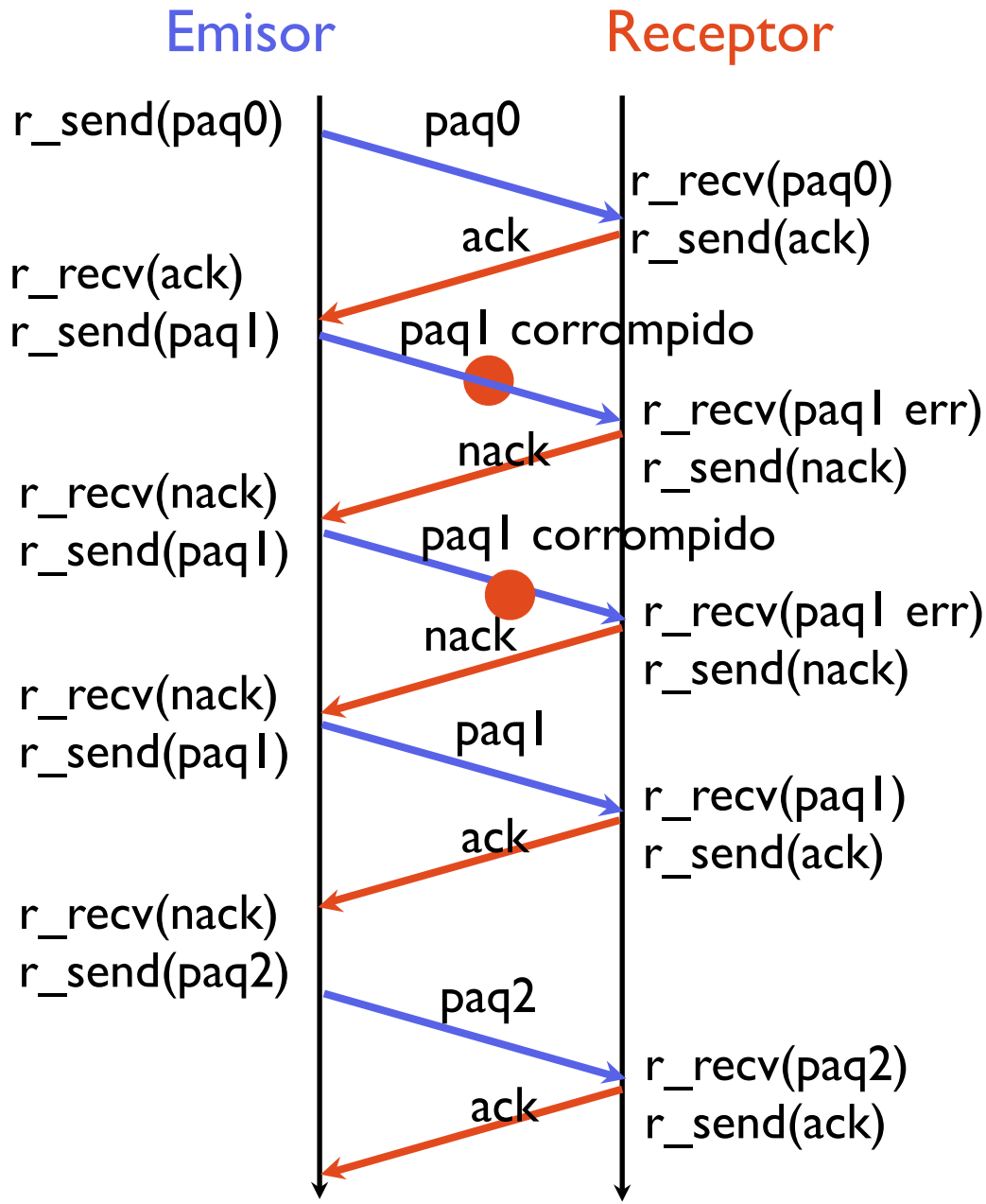
Idea Stop & wait

- Enviar un paquete con una parte de los datos
  - Si recibe los datos correctamente el receptor debe contestar enviando una confirmación (ACK)
  - Si recibe los datos pero incorrectamente el receptor debe contestar enviando una no-confirmacion (NACK)

- El emisor tras enviar los datos espera a que pase una de estas cosas
  - Si recibe un ACK puede proceder a enviar el siguiente paquete de datos
  - Si recibe un NACK vuelve a enviar el mismo paquete de datos

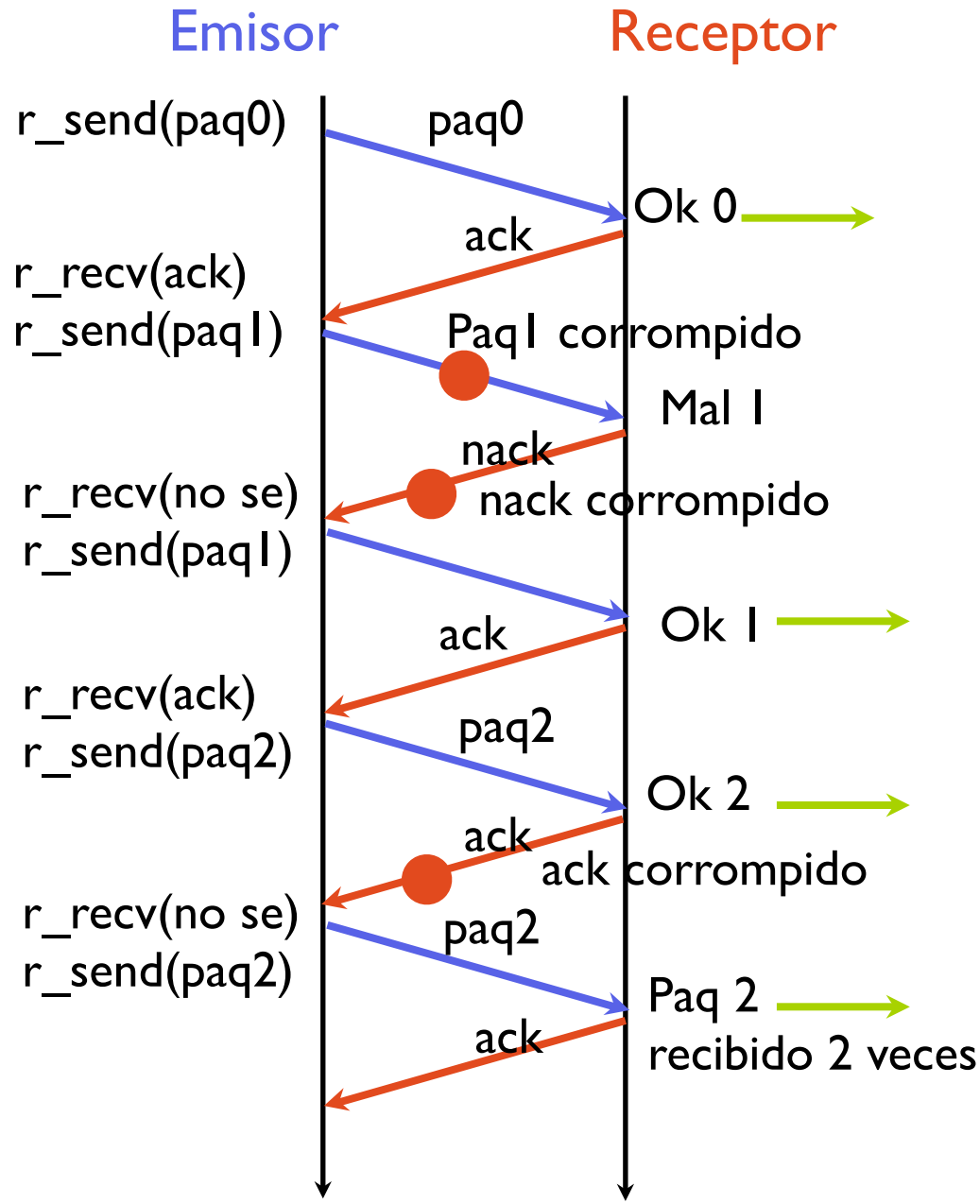
Stop-and-wait

- El emisor controlado por el receptor
  - ACK (recibido OK manda otro)
  - NACK (recibido mal manda otra vez el mismo)
  - Mientras no me dice nada no envío
- Esto puede considerarse también control de flujo (el emisor envía cuando el receptor le da permiso) = regulación de flujo por el receptor



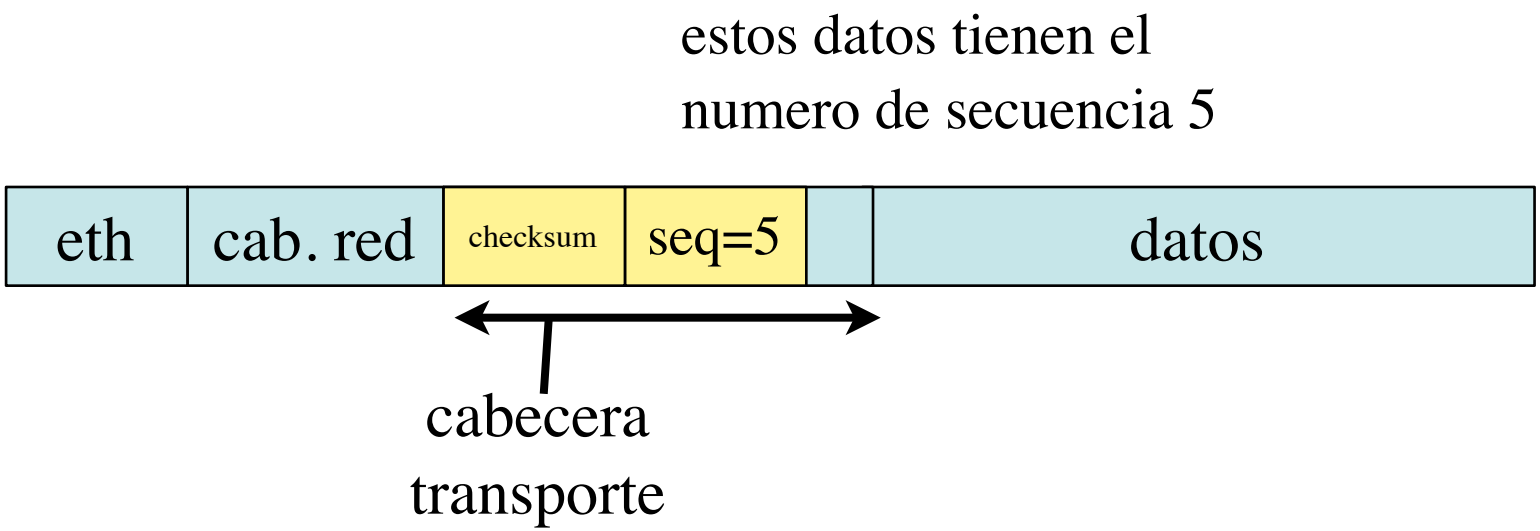
Problemas con stop-and-wait

- ¿Qué pasa si hay un error en la transmisión del ACK o NACK?
- Soluciones complican el protocolo
  - Detección de errores para ACK y NACK?
    - y que pasa si se pierden las confirmaciones del ACK/NACK
  - Checksums que permitan no solo detectar sino corregir errores?
    - Mucha información
  - Reenviar los datos si no entiendo el ACK/NACK ??
    - **Nuevo problema: paquetes duplicados**



Solución

- Los protocolos más usados utilizan contra esto números de secuencia del paquete
- El paquete va etiquetado con un numero de secuencia que permite confirmarlo/ rechazarlo indicando cual



- El numero de secuencia es un campo del paquete por lo que podrá tener una serie finita de valores
- Aunque es fácil asignar bits para que el numero de secuencia pueda crecer mucho antes de dar la vuelta, veamos primero las bases con números de secuencia en rangos limitados

Con sólo 2 números de secuencia 0 y 1

- El receptor solo entrega a la aplicación el paquete correcto
- Mejora de nombre
  - Para el receptor

    ACK seq=0 y NACK seq=1

    significan lo mismo

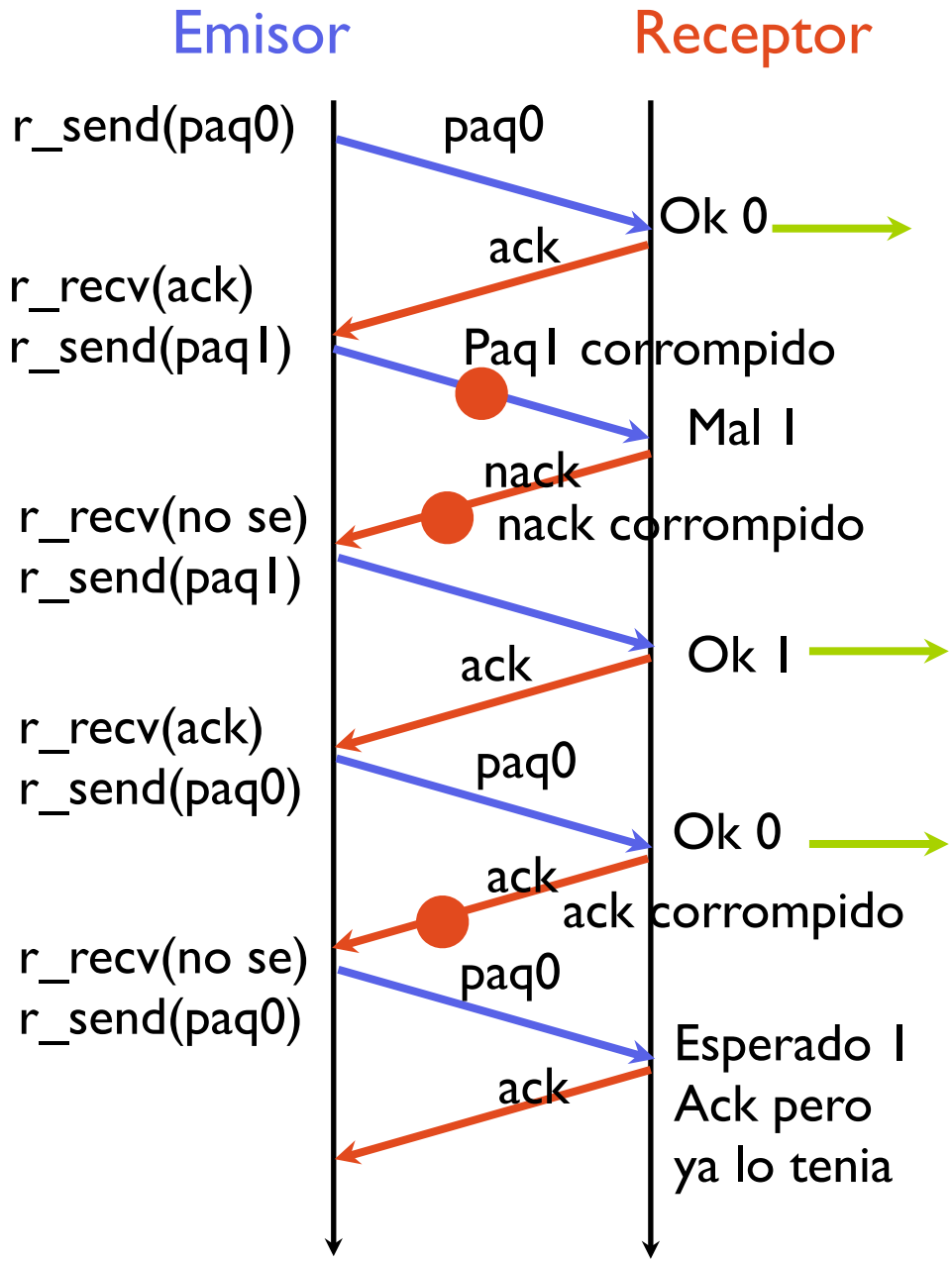
    diremos

    ACK 1 = esperando el 1

    (algunos llaman RR1

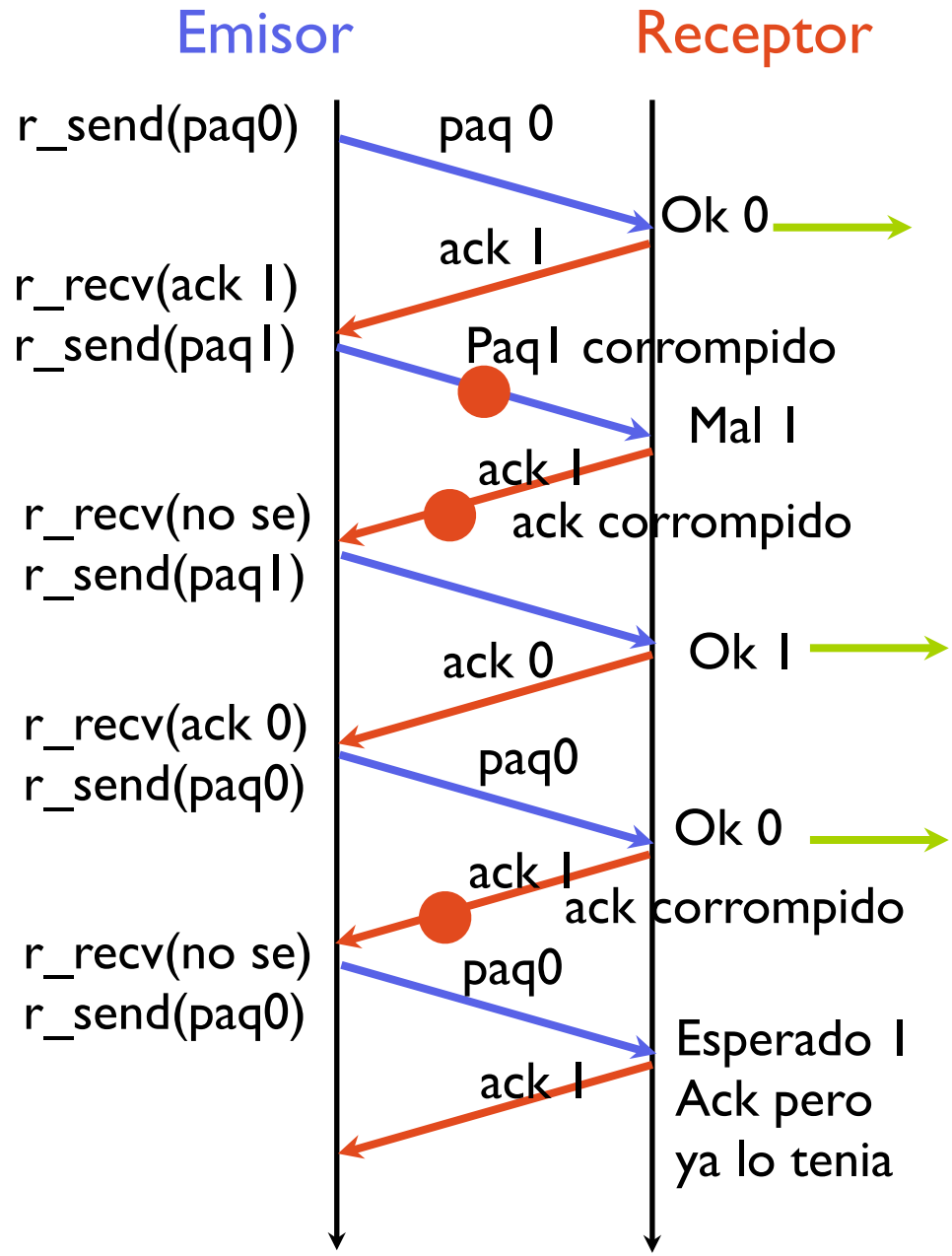
    Ready to receive 1)
  - **ACK 0 = esperando el 0**

    (en vez de ACK seq=1 y NACK seq=0)



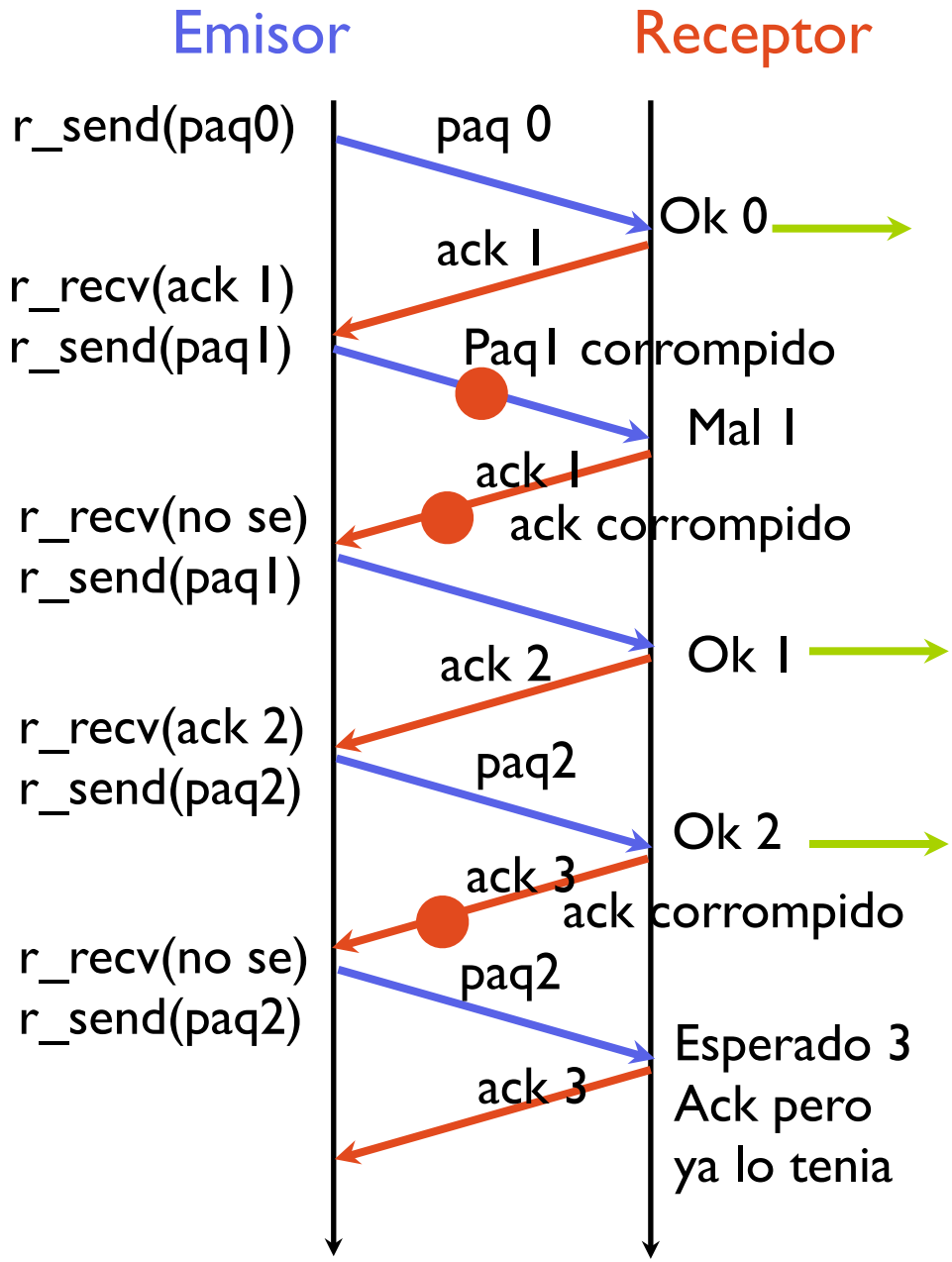
Con sólo 2 números de secuencia 0 y 1

- Normalmente no se usa NACK
- Se envia una confirmación ACK que indica el proximo paquete que espero recibir
- En lugar de decir NACK dices que sigues esperando recibir el que esperabas antes
- Normalmente se usan mas de 2 números de secuencia



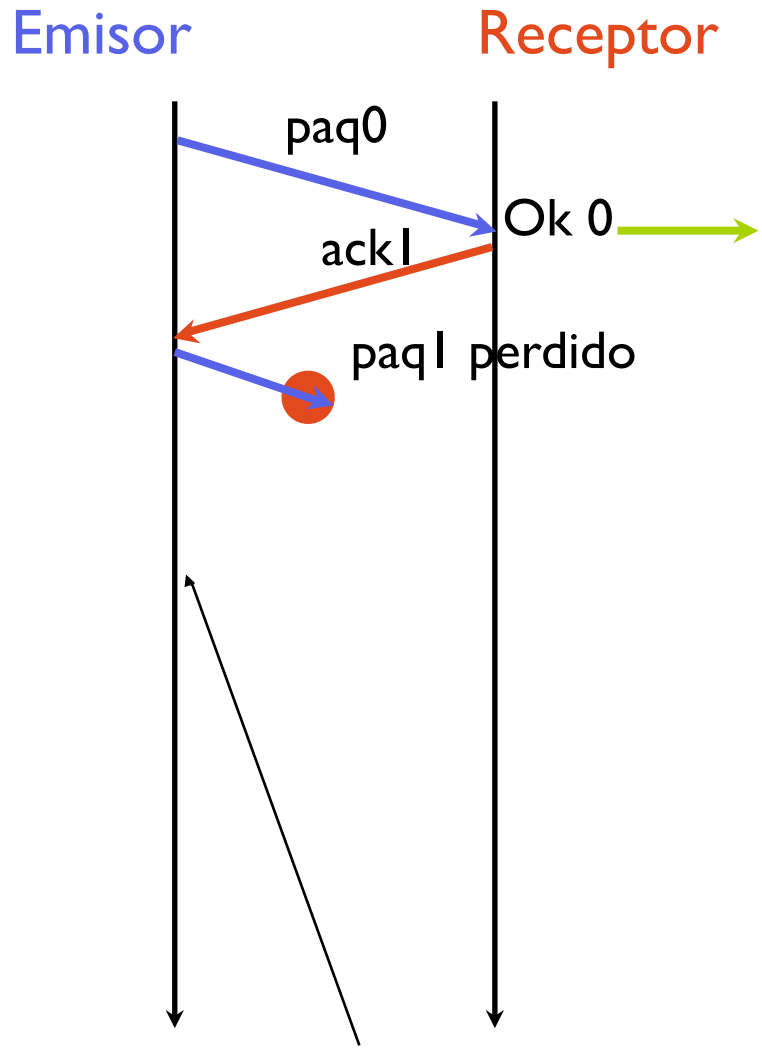
Con más números de secuencia

- Normalmente no se usa NACK
- Se envia una confirmación ACK que indica el proximo paquete que espero recibir
- En lugar de decir NACK dices que sigues esperando recibir el que esperabas antes
- Normalmente se usan mas de 2 números de secuencia



Pérdidas de paquetes

- ¿Y si el paquete no llega a su destino?
- El receptor no puede enviar NACK ni ACK…
- El emisor no va a hacer nada hasta que el receptor le diga si lo ha recibido
- Bloqueo !!!

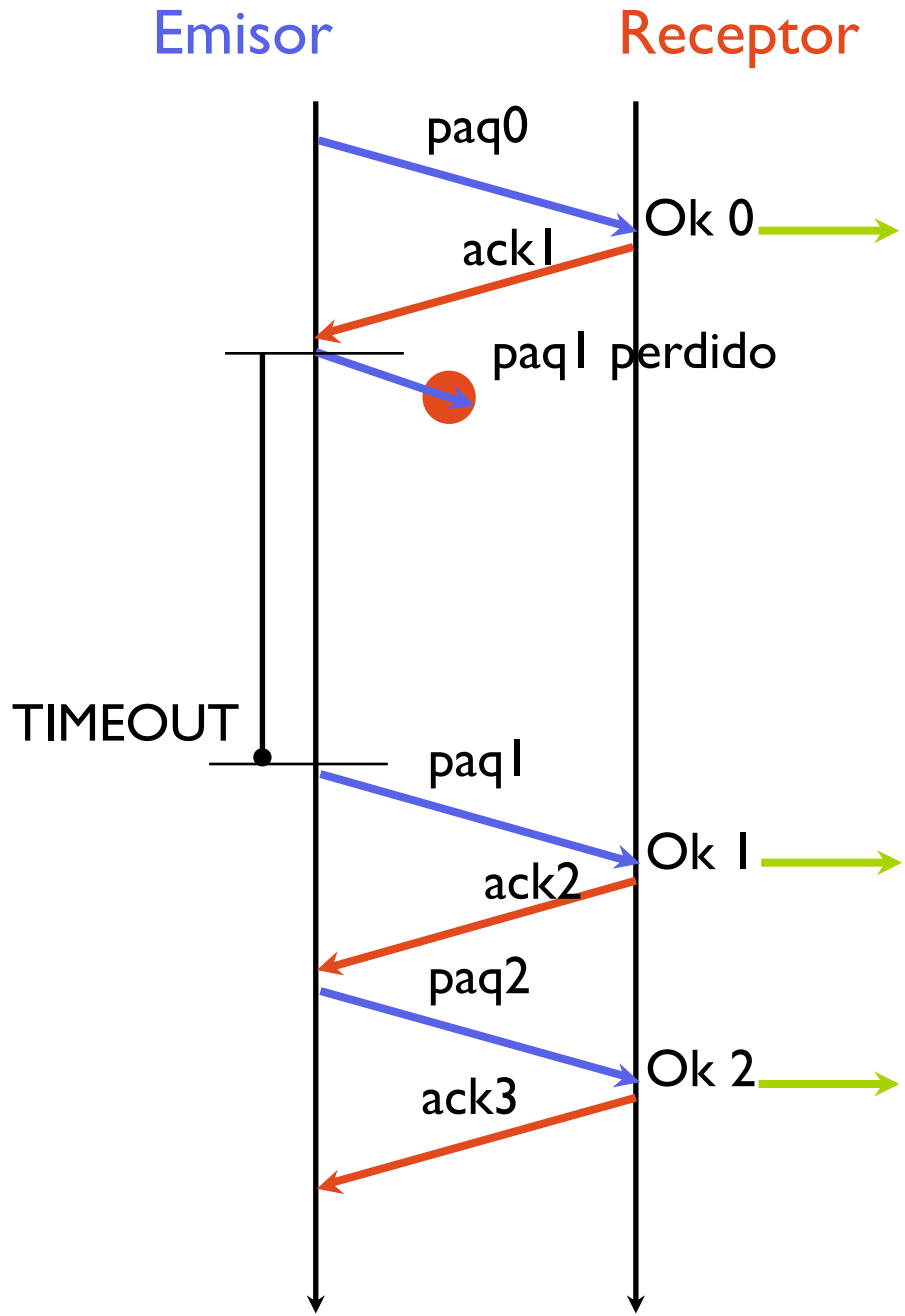


Emisor no puede enviar hasta recibir el ACK

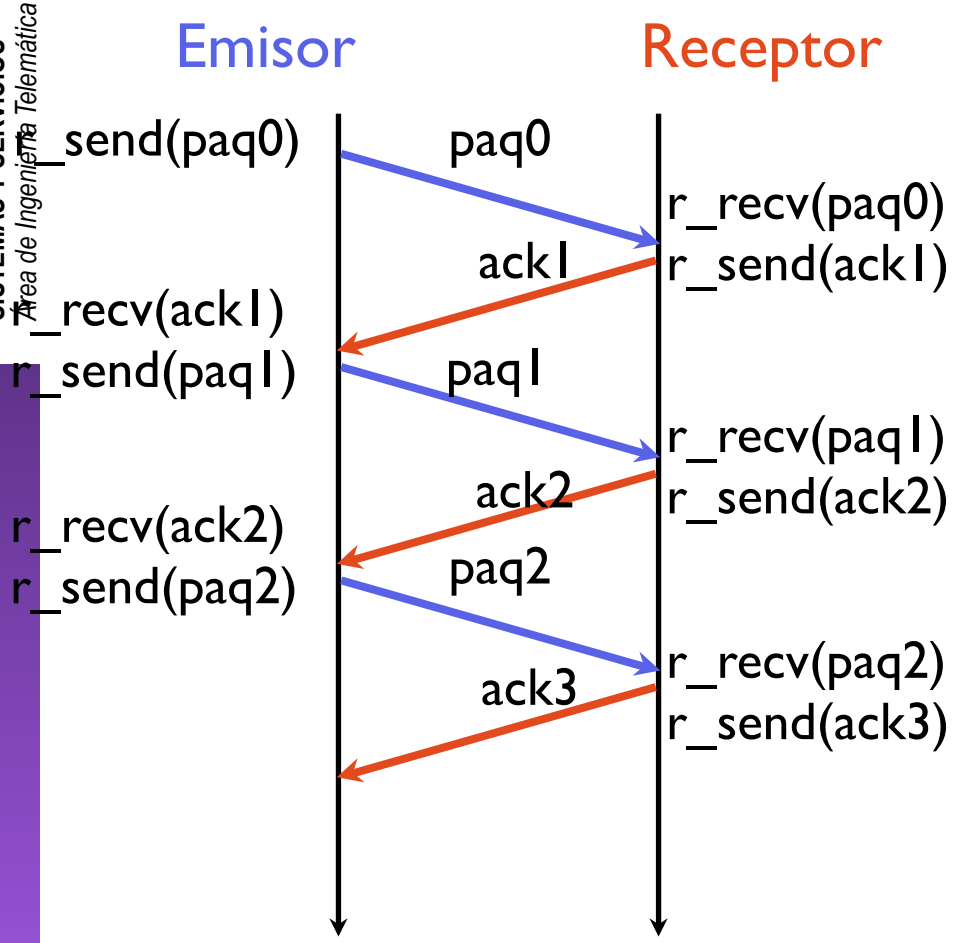
- Hay alguna manera de resolverlo con condiciones sobre los paquetes?
- NO
- [vease el famoso problema de los generales bizantinos]

Pérdidas de paquetes

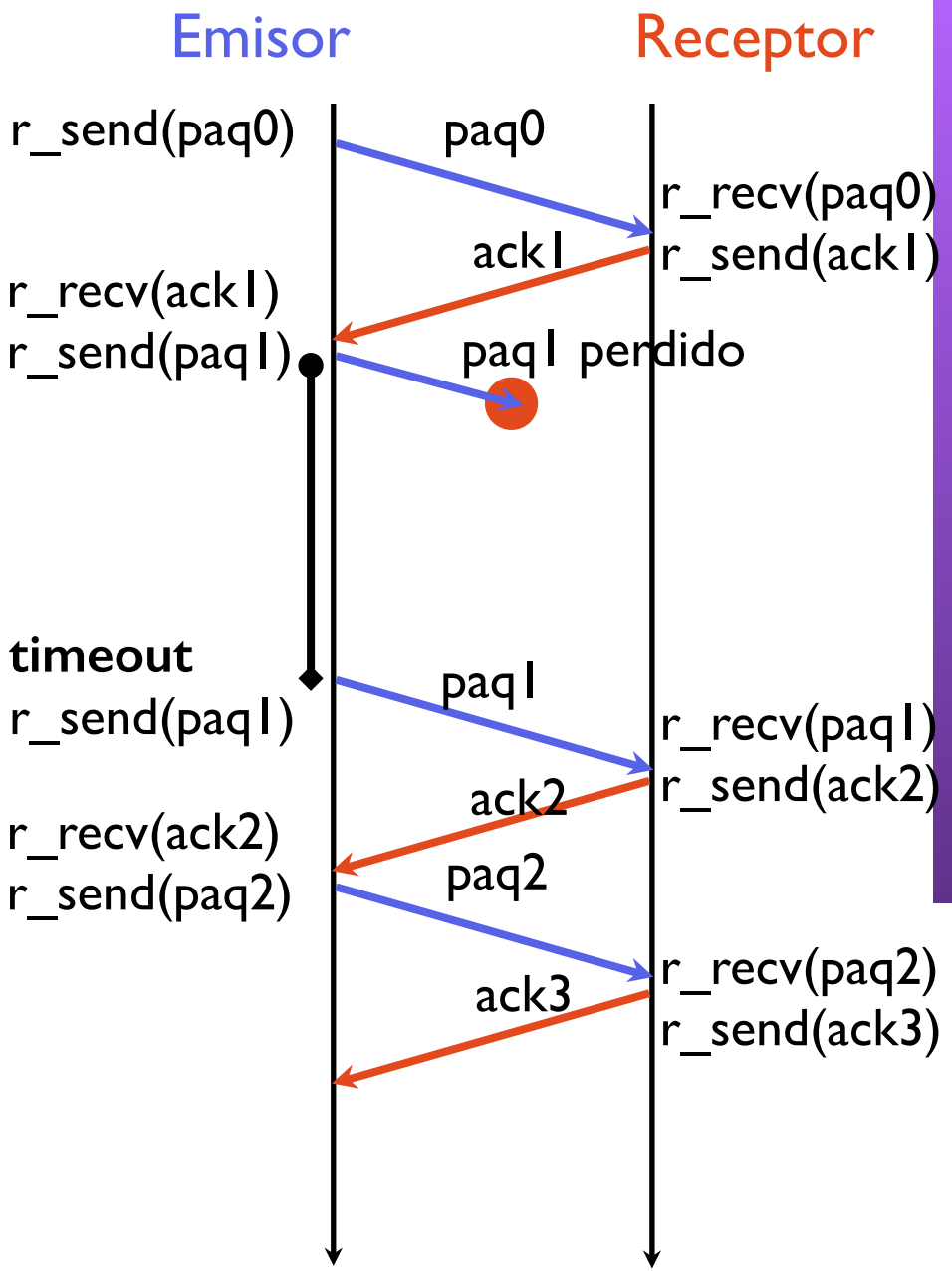
- Se puede resolver usando el tiempo (y no se puede garantizar cuanto tardara en resolverse)
- Si se pierde un paquete el emisor se queda bloqueado en un estado
- Para romper el bloqueo usamos un temporizador en el emisor
  - Al enviar un paquete de datos ponemos en marcha un temporizador
  - Si transcurrido un tiempo, no se ha recibido ACK (TIMEOUT), reenviamos el paquete
- El receptor no se modifica
- El emisor insiste hasta estar seguro de que el receptor lo ha recibido



Ejemplos

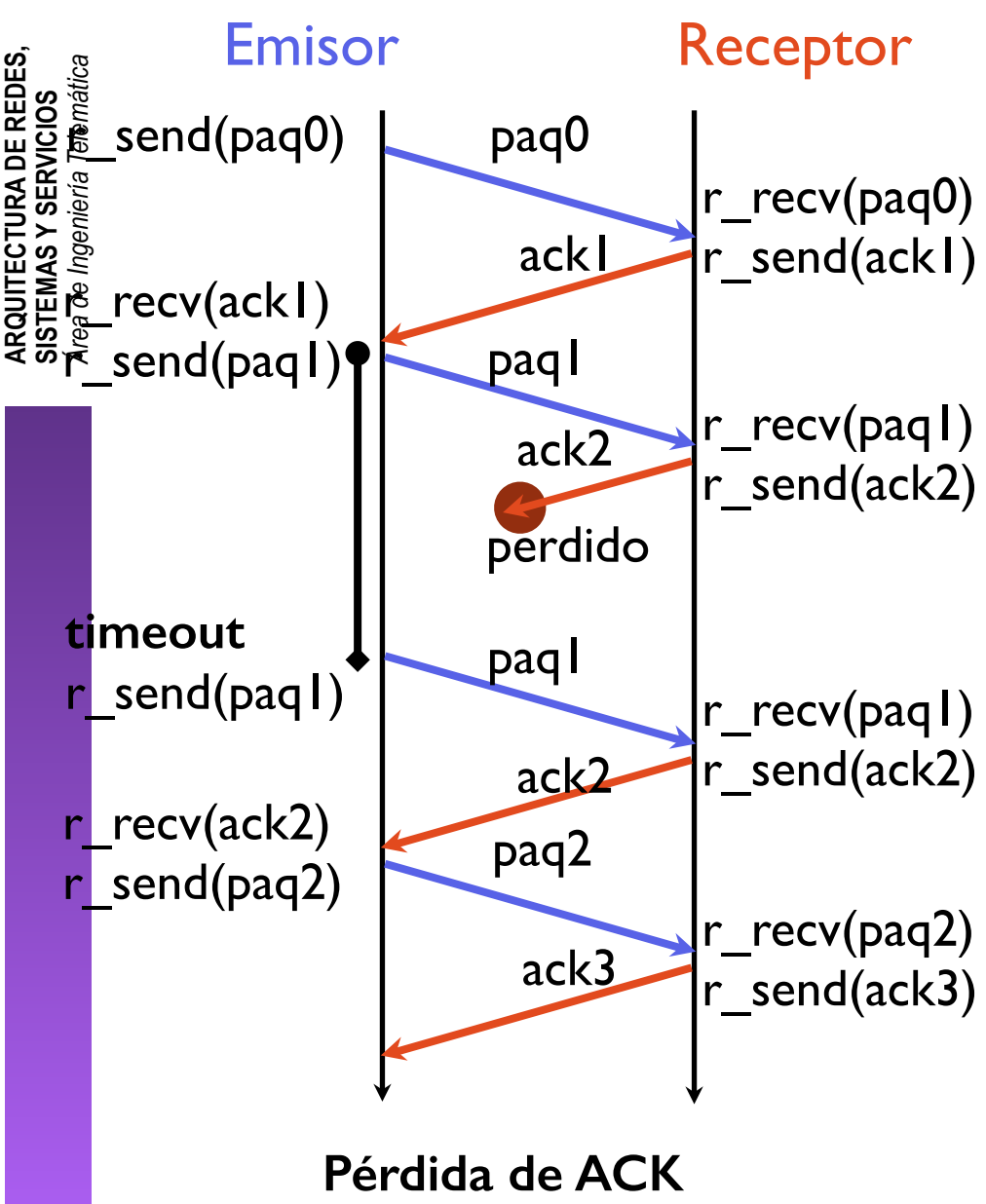


Operación normal

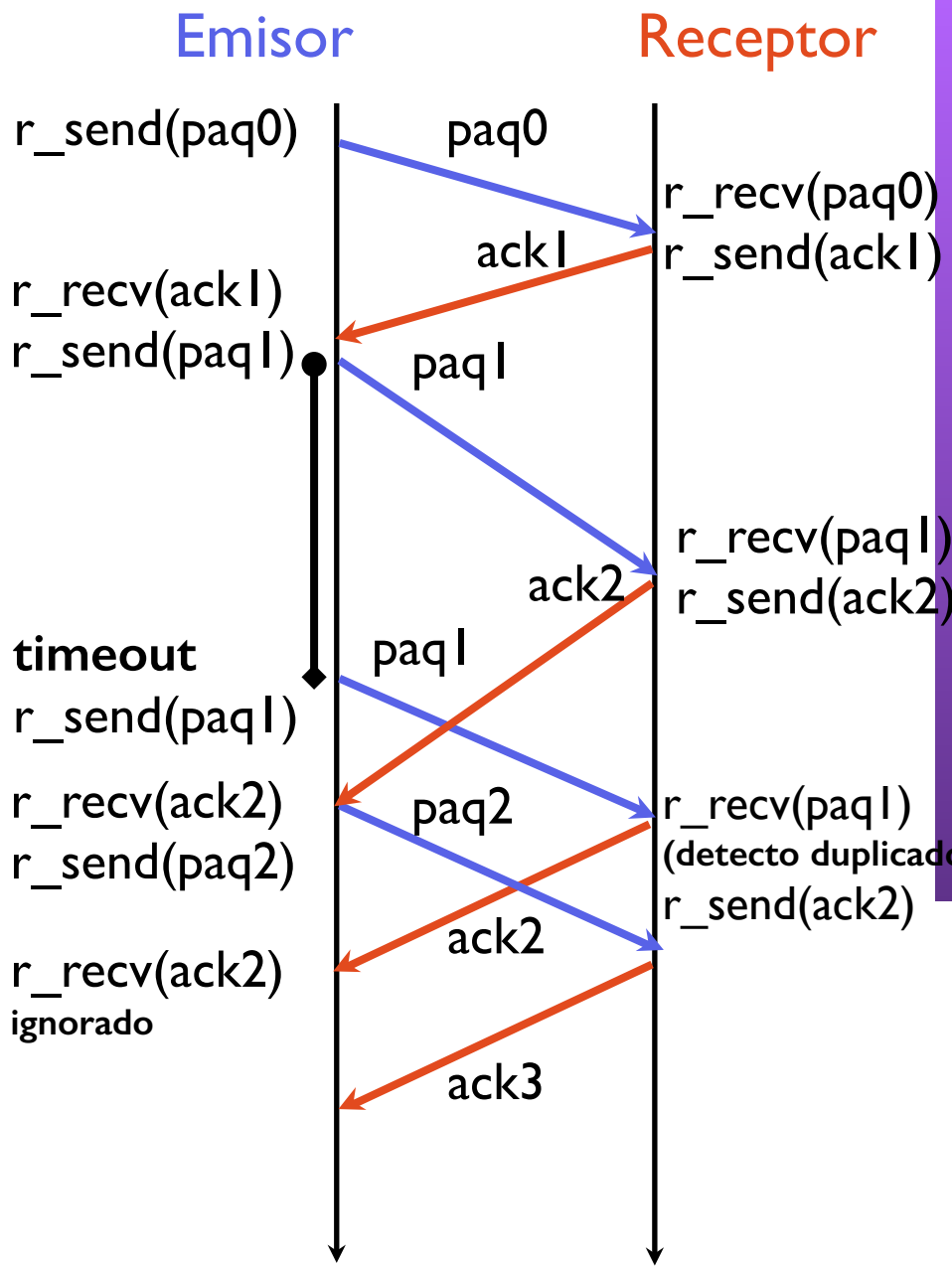


Pérdida de paquete

Ejemplos



Pérdida de ACK



Timeout prematuro

Hasta ahora

- Protocolo mas simple
  - Stop and wait
  - Con números de secuencia para no entregar duplicados
  - Con ACK que indica cual es el dato que espero
  - Con timeout para el caso de que se pierdan paquetes
- **Garantiza fiabilidad sobre un canal con errores de bits o perdidas**
  - Si el protocolo funciona durante todo el tiempo que haga falta y no se pierden todos los paquetes al final la información llega
  - Lo que no garantiza es cuanto tarda en llegar
- Problemas
  - ¿Cuanto tarda? = ¿Cómo de rápido es el protocolo?