Tecnología Digital IV: Redes de Computadoras

Clase 9: Nivel de Transporte - Parte 3

Lucio Santi & Emmanuel Iarussi

Licenciatura en Tecnología Digital
Universidad Torcuato Di Tella

8 de abril de 2025

Agenda

- Servicios del nivel de transporte
- Multiplexación y demultiplexación
- Transporte no orientado a conexión: UDP
- Transferencia de datos confiable

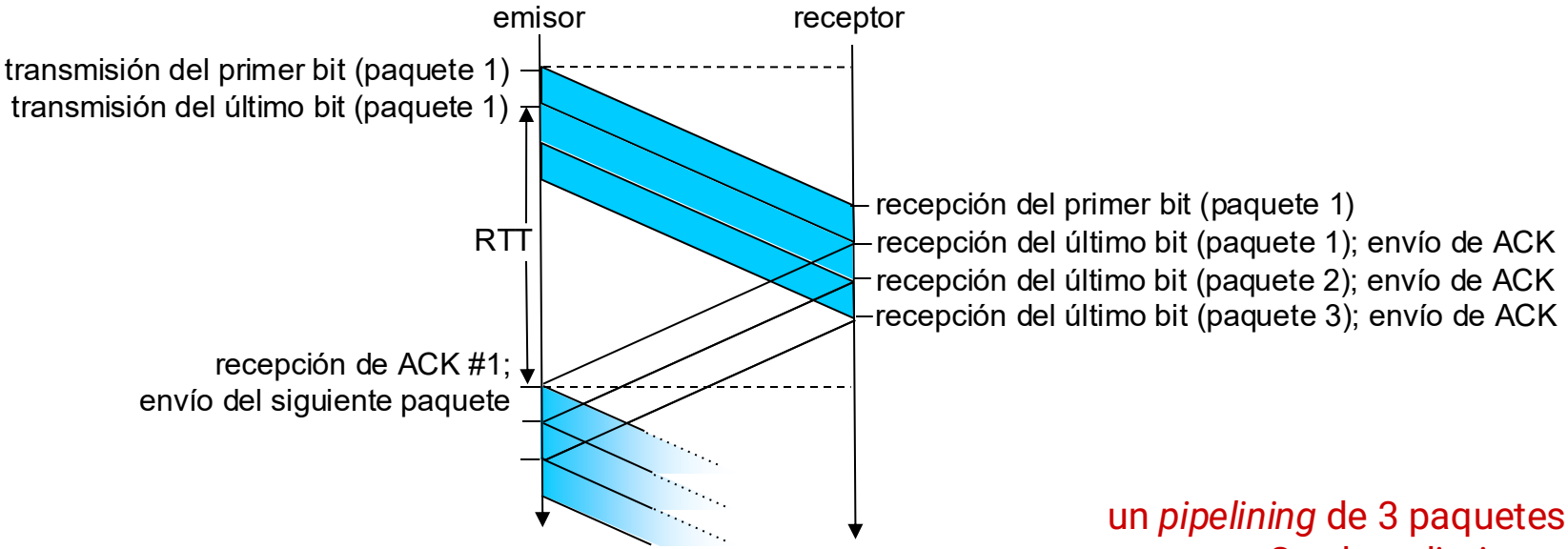
Protocolos confiables (continuación)

rdt3.0: *pipelining*

Podemos incrementar el rendimiento mediante la técnica de ***pipelining***

- El emisor mantiene **múltiples paquetes** “en vuelo” (en vez de sólo uno)
- Los números de secuencia respectivos se incrementan ante cada nueva transmisión
- Se deben gestionar *buffers* (en el emisor y/o receptor)

Rendimiento utilizando *pipelining*



un *pipelining* de 3 paquetes aumenta **3x** el rendimiento del protocolo

$$U = \frac{3L / R}{\text{RTT} + L / R} = \mathbf{0.00081}$$

Pipelining vía *Go-Back-N (GBN)*

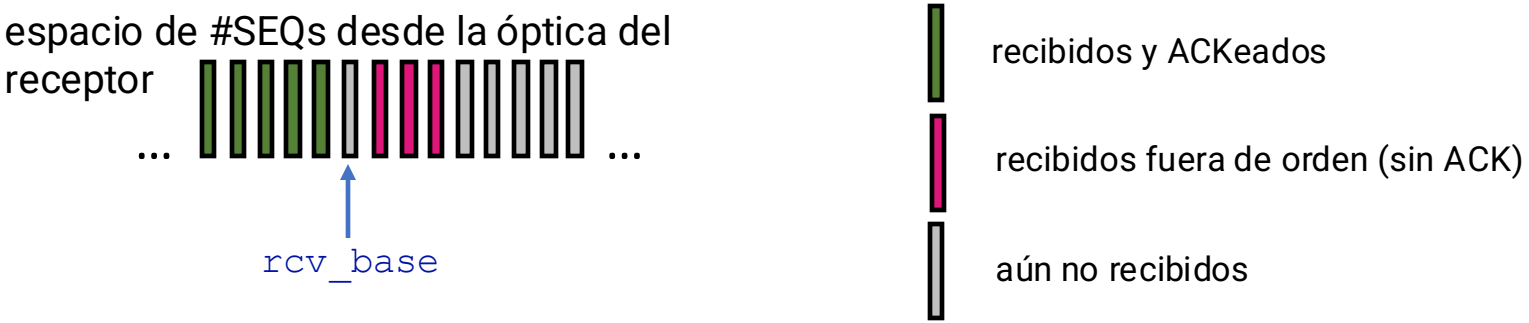
- El emisor mantiene una **ventana** de hasta *N* paquetes en vuelo
  - Utiliza un número de secuencia (**#SEQ**) de *k* bits en el header

- **ACK acumulativo**, ACK(*n*): reconoce todos los paquetes hasta el de #SEQ *n* (incluido)
  - Al recibir un ACK(*n*), se desplaza la ventana hacia adelante para comenzar en ***n+1***
- El **timer** corresponde al paquete en vuelo más antiguo
- *timeout(n)* dispara una retransmisión del paquete *n* junto con **todos** los de #SEQ más grande en la ventana

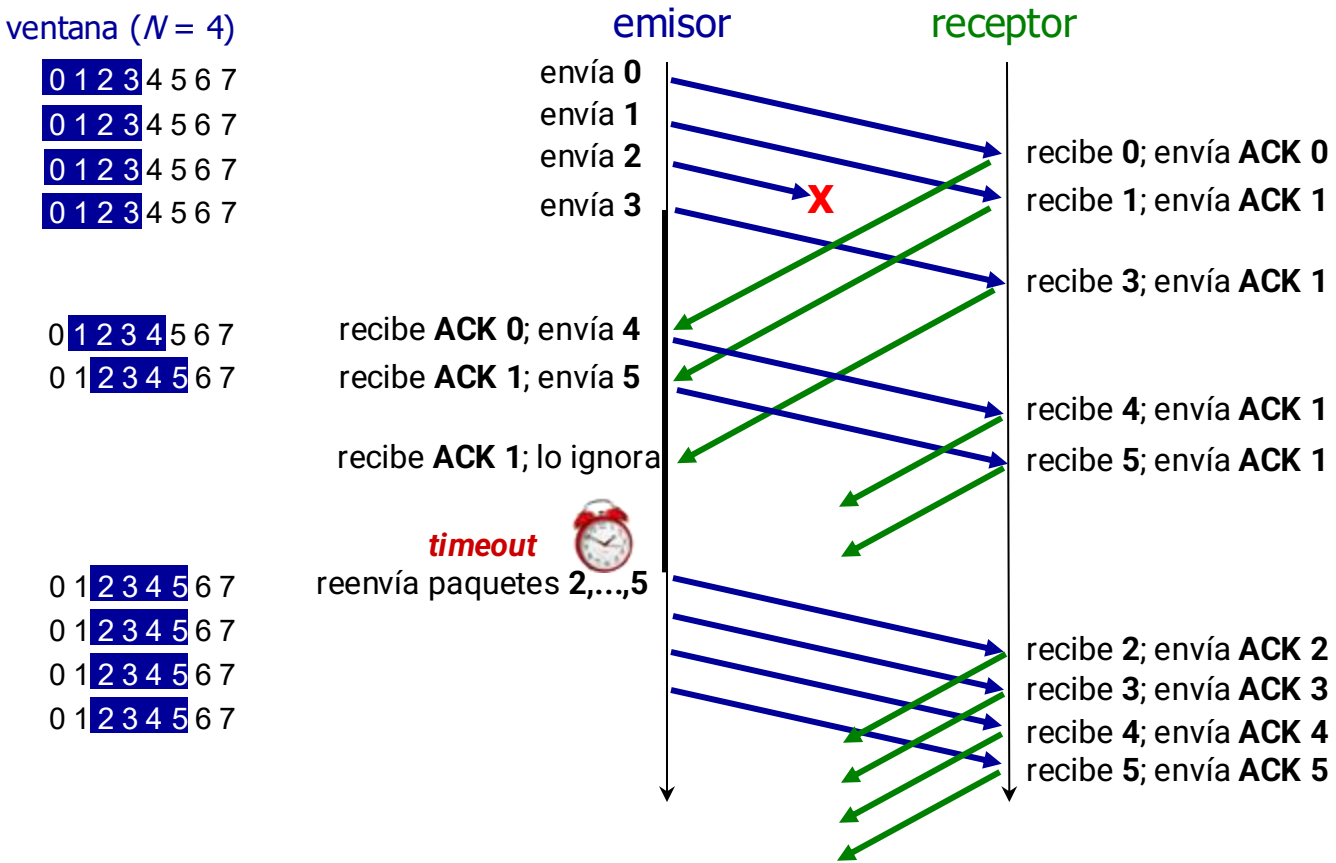
Demo: https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/go-back-n-protocol/index.html

Go-Back-N: receptor

- El receptor siempre envía un ACK para el paquete con el #SEQ más alto (en orden) recibido correctamente
  - Puede generar ACKs duplicados
  - Debe recordar el siguiente #SEQ esperado, `rcv_base`
- Al recibir un paquete fuera de orden,
  - Es posible almacenarlo o descartarlo (decisión de implementación)
  - Se reenvía ACK para el paquete con el #SEQ más alto recibido



Go-Back-N: ejemplo



Pipelining vía *Selective Repeat*

- Con **Selective Repeat**, el receptor reconoce uno a uno cada paquete recibido correctamente
  - Almacena en *buffers* los paquetes para entregarlos (en orden) a la capa superior, eventualmente
- El emisor dispara un *timeout* y retransmite cada paquete individualmente
  - Mantiene un *timer* por cada paquete sin ACK
- La ventana de emisión consta de *N* #SEQs consecutivos

Ventanas en *Selective Repeat*

Demo: https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/selective-repeat-protocol/index.html

Acciones de emisor y receptor en *SR*

Emisor

Datos desde capa superior

- Si el próximo #SEQ disponible está dentro de la ventana, enviar el paquete

timeout(*n*)

- Reenviar paquete *n*; reiniciar *timer*

ACK(*n*) en [`send_base`, `send_base`+*N*]

- Marcar paquete *n* como recibido
- Si *n* es el #SEQ más chico sin ACK, desplazar la base de la ventana al próximo #SEQ sin ACK

Receptor

Paquete *n* en [`rcv_base`,`rcv_base`+*N*-1]

- Enviar ACK(*n*)
- Si está fuera de orden, almacenarlo
- Si no, entregarlo (junto con otros en orden); desplazar ventana al siguiente #SEQ aún no recibido

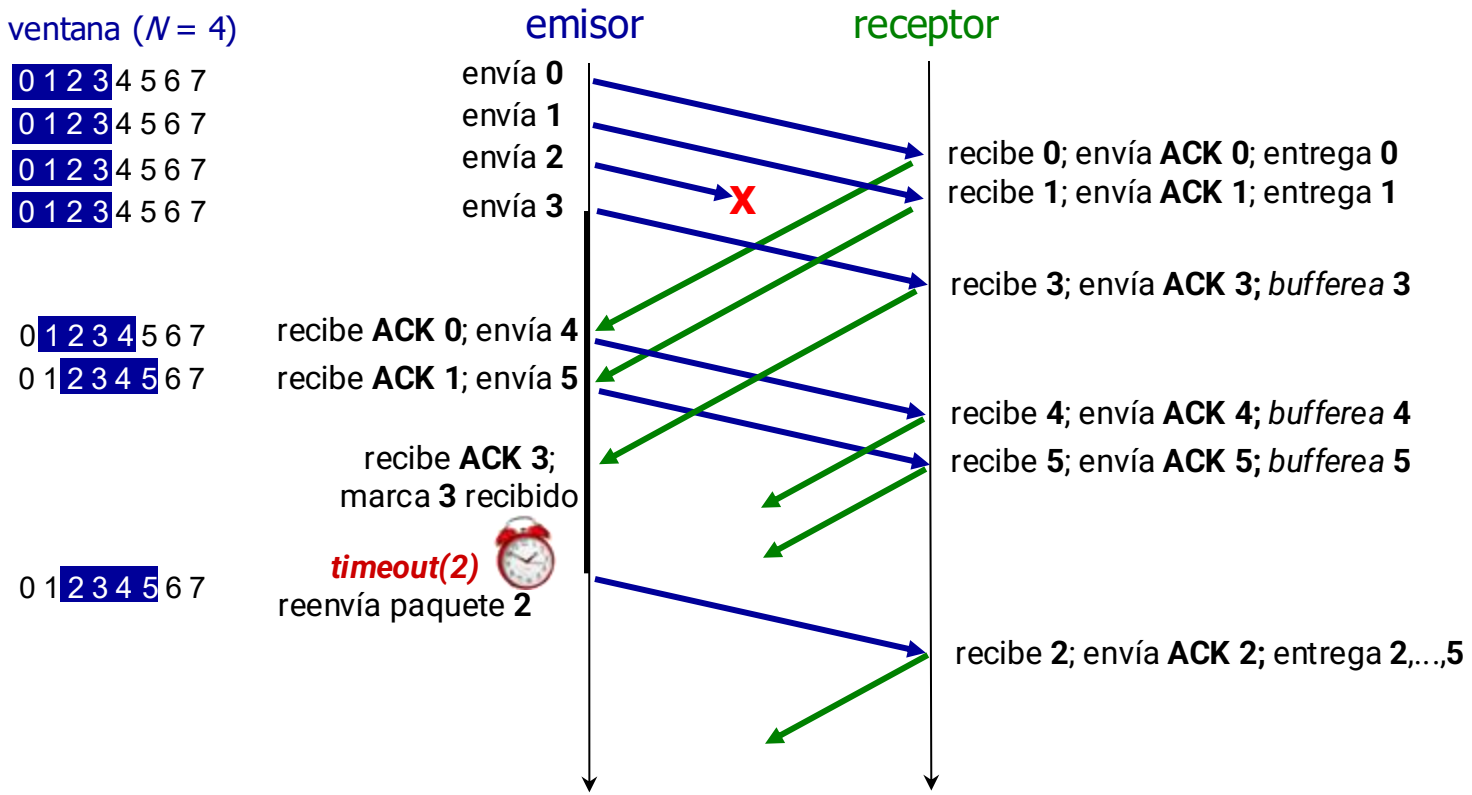
Paquete *n* en [`rcv_base`-*N*,`rcv_base`-1]

- Enviar ACK(*n*)

En cualquier otro caso

- Ignorar y descartar el paquete

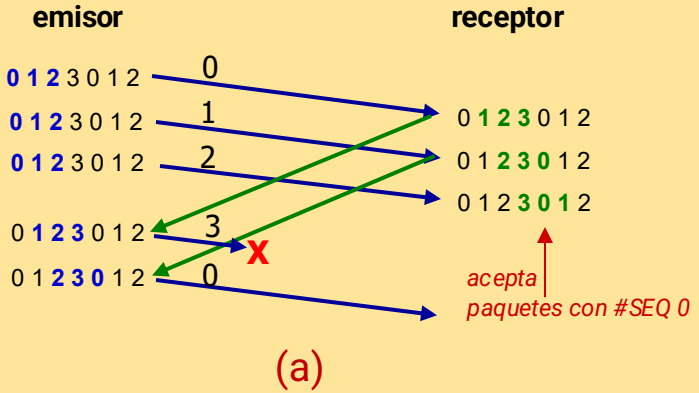
Selective Repeat: ejemplo



Ejercicio!

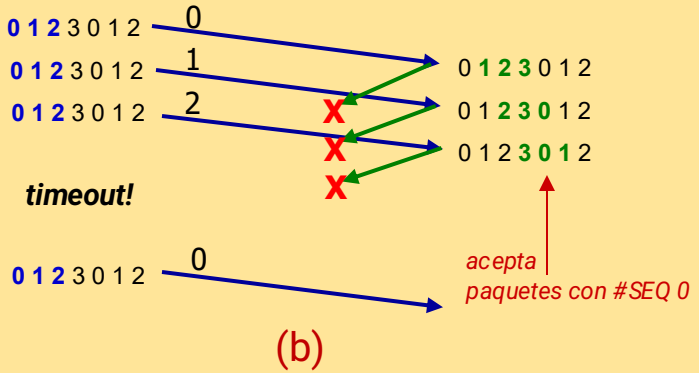
Considerar estos dos escenarios:

- #SEQs: 0, 1, 2, 3 (*k* = 2 bits)
- Ventana: *N* = 3



(a)

¿Cuál es la relación que se necesita entre el tamaño de la ventana *N* y la cantidad de bits *k* de los números de secuencia para evitar el problema del escenario (b)?



(b)