Copyright

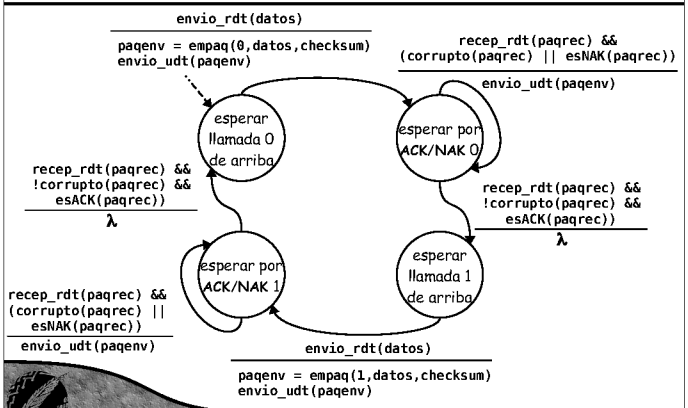
Contenidos

- Servicios y protocolos de la capa de transporte
- Multiplexado y demultiplexado de segmentos
- Transporte no orientado a la conexión (**UDP**)
- Teoría de transporte confiable de datos
- Transporte orientado a la conexión (**TCP**)
- Establecimiento y cierre de conexiones
- Teoría de control de congestión
- Control de congestión en **TCP**

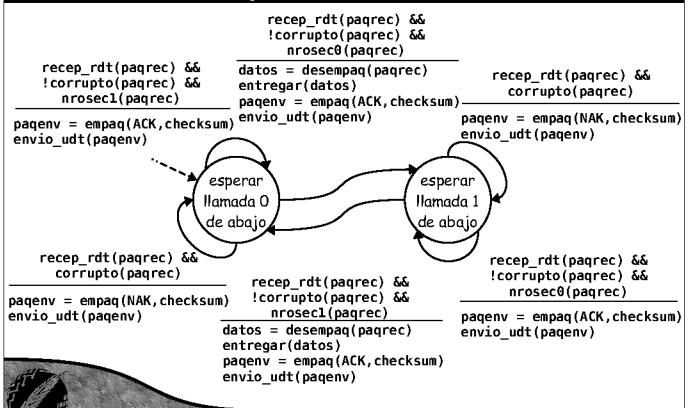
Paquetes duplicados

- Incorporar un esquema de numeración de los paquetes implica que el emisor marque cada paquete que envía con un cierto número
  - En el escenario que se corrompía el **ACK/NAK** enviado por el receptor, el emisor simplemente asume que se trataba de un **NAK** y reenvía el último paquete (usando el mismo número de paquete que la vez anterior)
  - El receptor en caso de recibir por segunda vez el mismo paquete simplemente lo descarta

Emisor RDT/2.1



Receptor RDT/2.1



Análisis de RDT/2.1

- **RDT/2.1** incorpora números de secuencia en los paquetes enviados por el emisor
  - Con sólo dos números de paquete es suficiente
- El emisor verifica la correcta recepción de los mensajes de **ACK/NAK**
- Los autómatas finitos que definen el protocolo requieren el doble de estados que en **RDT/2.0**
  - Los estados distinguen el número de secuencia del próximo paquete a ser enviado o recibido

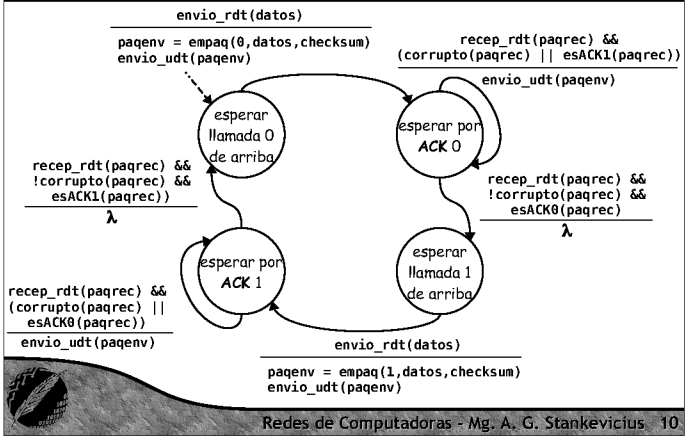
Análisis de RDT/2.1

- El receptor debe verificar que los paquetes recibidos tengan el número de secuencia esperado
  - Caso contrario, se trata de un paquete duplicado, el cual debe ser descartado
- El receptor no tiene forma de saber si el emisor recibió correctamente el último **ACK/NAK**
  - Se dará cuenta implícitamente que el emisor recibió correctamente el **ACK** cuando vea un paquete numerado con el próximo valor de la secuencia

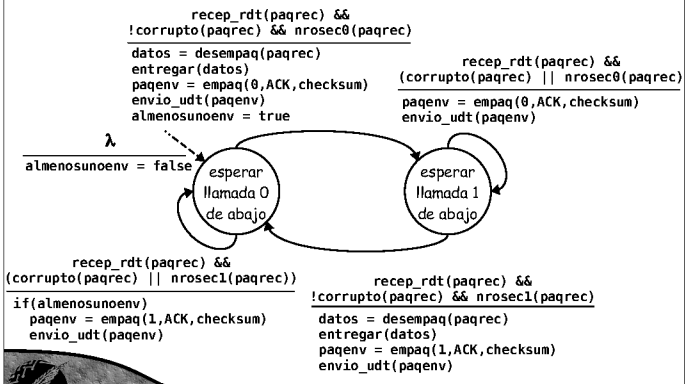
RDT/2.2

- El protocolo **RDT/2.1** admite ser optimizado evitando tener que hacer uso de los mensajes **NAK** de reconocimiento negativo
- La idea es que el receptor indique qué paquete está reconociendo al enviar un **ACK**
  - Debemos incorporar el número de secuencia en los mensajes enviados por el receptor
- La recepción por parte del emisor de un segundo **ACK** hace las veces de **NAK**
  - En ambos casos se debe reenviar el último paquete

Emisor RDT/2.2



Receptor RDT/2.2



RDT/3.0

- El protocolo **RDT/3.0** permite envío confiable de datos a través de un canal que puede causar errores a nivel de los bits y/o perder en su totalidad alguno de los mensajes enviados
  - El protocolo debe contemplar que se pierda un paquete o un mensaje de **ACK**
  - Se deben resolver dos problemas: cómo detectar las pérdidas y qué hacer cuando se produzca una
- Los mecanismos presentes en **RDT/2.2** no permiten detectar que se produjo una pérdida

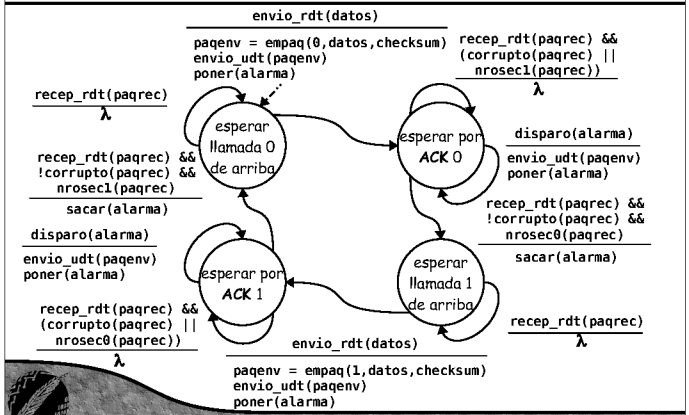
RDT/3.0

- Un posible solución es quedar a la espera de un paquete o un mensaje **ACK** y cuando se tenga la certeza de que se perdió reenviarlo
  - ¿Cuánto hay que esperar para tener la certeza que se perdió el último mensaje enviado?
  - Esta alternativa funciona, pero implica pagar un costo muy alto en eficiencia al perderse un mensaje
- Una mejor opción es quedar a la espera de una respuesta por un tiempo razonable y si no llega asumir que se perdió y reenviarlo

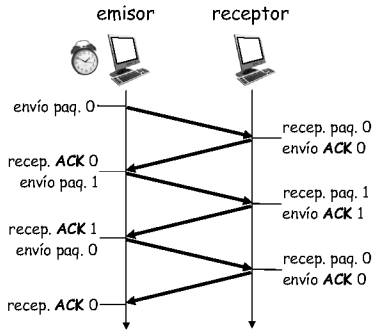
RDT/3.0

- ¿Qué sucede si en realidad el paquete estaba retrasado, pero no se había perdido?
  - El reenvió generará un mensaje duplicado, pero el protocolo ya maneja correctamente esa situación
  - Emisor y receptor deben indicar el número de secuencia en sus mensajes
- Para implementar esta política hace falta contar con un temporizador programable

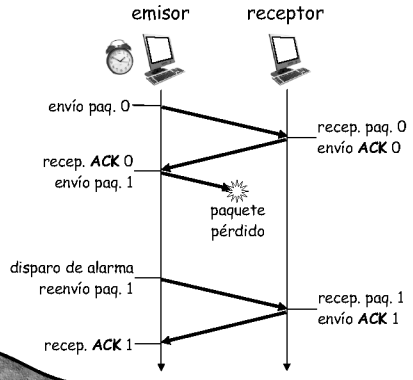
Emisor RDT/3.0



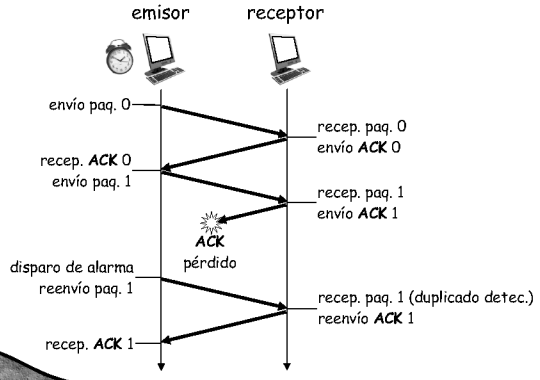
RDT/3.0 ideal



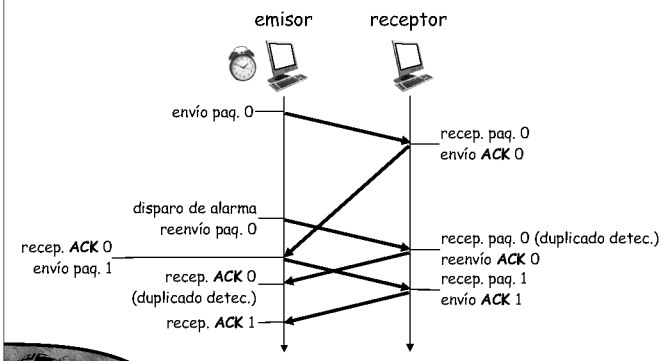
Pérdida de un paquete



Pérdida de un ACK



Reenvío prematuro



Análisis de RDT/3.0

- El protocolo **RDT/3.0** cumple con el objetivo propuesto, esto es, permite la transmisión confiable de datos sobre un canal no confiable
- No obstante, este protocolo presenta un desempeño bastante pobre, lo que lo torna poco aplicable a escenarios del mundo real
  - Esta es una característica propia de los protocolos de la familia "stop-and-wait"

Desempeño de RDT/3.0

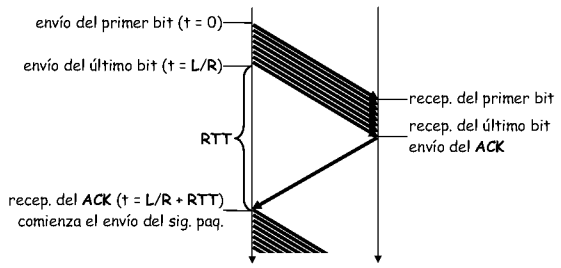
- Consideremos la siguiente situación:
  - Se dispone de una línea dedicada de 1 Gb/s que une dos de las oficinas de una cierta compañía
  - Se intercambian paquetes de 1 KB y el **RTT** entre estas oficinas es de 30 ms

$$d_{trans} = \frac{\text{L (cantidad de bits)}}{\text{R (ancho de banda)}} = \frac{8000 \text{ b}}{10\text{^}9 \text{ b/s}} = 8 \text{ microseg.}$$

$$u_{enlace} = \frac{\text{L/R}}{\text{RTT + L/R}} = \frac{.008 \text{ ms}}{30.008 \text{ ms}} = 0.027\%$$

1 paq. / 30 ms = 1 Gb/s x 0.027% = 33.3 KB/s

Operatoria stop-and-wait

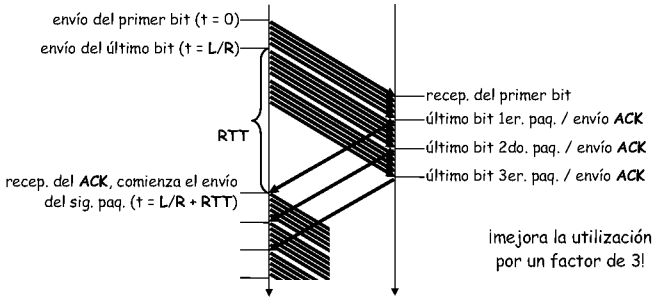


$$u_{enlace} = \frac{L/R}{RTT + L/R} = \frac{.008\ ms}{30.008\ ms} = 0.027\%$$

Operatoria en pipeline

- El factor de utilización obtenido de apenas 0.027% es a las claras inaceptable
- Habría que permitir una operatoria en pipeline a fin de elevar el factor de ocupación
  - La idea básica es permitir que varios paquetes estén en camino al mismo tiempo
  - A tal efecto hace falta incrementar los números de secuencia disponibles
  - También hace falta alguna forma de almacenamiento intermedio tanto en emisor como receptor

Operatoria en pipeline



$$u_{enlace} = \frac{3 \times L/R}{RTT + L/R} = \frac{.024\ ms}{30.008\ ms} = 0.08\%$$

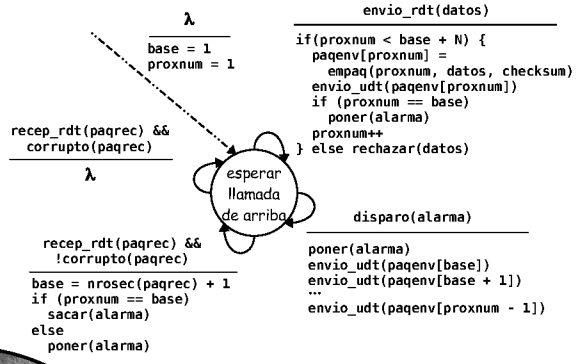
Operatoria en pipeline

- Go-Back N (**GBN**)
  - El emisor puede tener hasta **n** paquetes aun sin confirmar
  - El receptor sólo envía **ACK**s acumulativos (¡no tolera huecos!)
  - El emisor sólo requiere un temporizador para el paquete más antiguo sin reconocer
- Selective Repeat (**SR**)
  - El emisor puede tener hasta **n** paquetes aun sin confirmar
  - El receptor envía **ACK**s individuales para cada paquete
  - El emisor mantiene un temporizador para paquete aun no reconocido

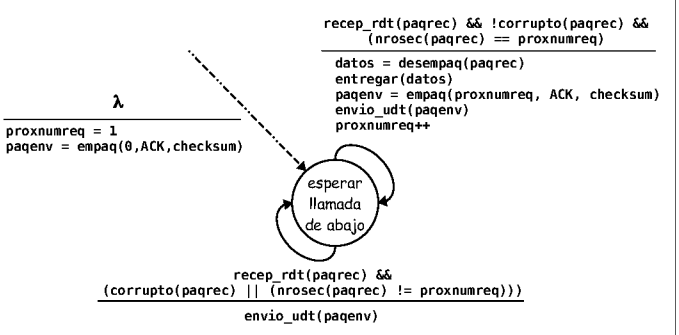
Protocolo Go-Back-N

- El protocolo **GBN** es una de las formas de implementar la operatoria en pipeline
  - Se reservan **k** bits del encabezado para los números de secuencia de los paquetes
  - Se permiten hasta una ventana deslizante de **n** paquetes en tránsito, esto es, aquellos cuales cuya confirmación de recepción aún no ha sido recibida

Emisor GBN



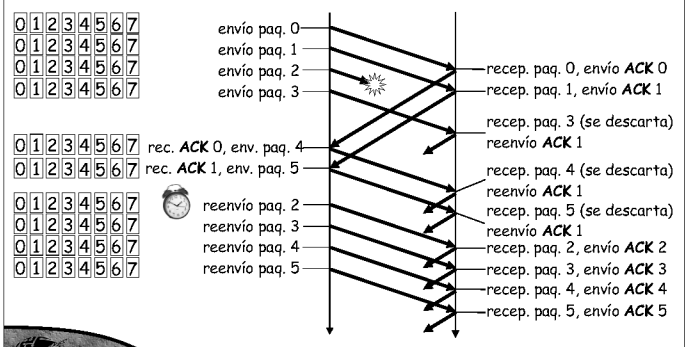
Receptor GBN



Análisis GBN

- El receptor sólo envía un mensaje de **ACK** para el paquete correctamente recibido con mayor número de secuencia
  - Esta política puede generar mensajes de **ACK** duplicados
  - El receptor sólo necesita recordar el número de secuencia del próximo paquete que desea
- Los paquetes recibidos fuera de orden son descartados
  - El receptor no requiere almacenamiento intermedio

GBN en acción



Repetición Selectiva

- El protocolo Repetición Selectiva (**SR**) es otra implementación de la operatoria en pipeline
  - La idea central es disminuir el número de paquetes a ser reenviados al recuperarse de una pérdida
- A diferencia de **GBN**, el receptor reconoce por separado a cada uno de los paquetes correctamente recibidos
  - El receptor debe contar con un almacenamiento intermedio para alojar los paquetes recibidos correctamente pero fuera de orden

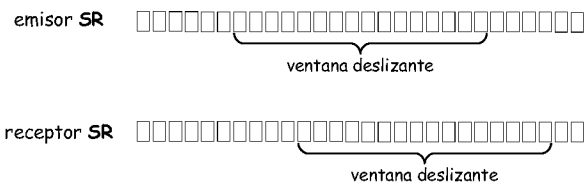
Repetición Selectiva

- El emisor sólo debe reenviar aquellos paquetes para los que aún no se haya recibido el mensaje de **ACK** correspondiente
  - Esto implica que debemos contar con un temporizador independiente para cada paquete enviado cuyo **ACK** asociado aún no haya sido recibido
- El emisor cuenta con una ventana deslizante de **n** números consecutivos de secuencia
  - La ventana representa el conjunto de paquetes que tiene permitido tener en tránsito al mismo tiempo

Visión del emisor y receptor

- enviados, **ACK** ya recibido
- enviados, esperando **ACK**
- usables, a ser enviados
- no usables

- recibidos fuera de orden, **ACK** ya enviado
- a la espera, aún no recibidos
- aceptables
- no usables



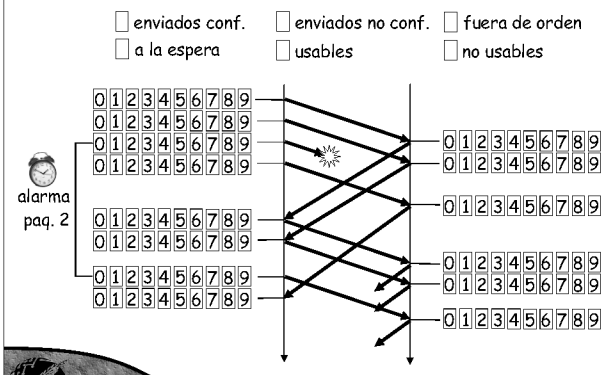
Emisor SR

- El emisor **SR** debe reaccionar ante los eventos que se presenten de la siguiente manera:
  - Al recibir un nuevo paquete a ser enviado debe verificar si el siguiente número de secuencia se encuentra dentro de la ventana
  - Al dispararse la alarma de un paquete debe reenviar sólo ese paquete y debe reiniciar el temporizador
  - Al recibir un **ACK** debe marcar ese paquete como recibido y en caso de ser el menor número de paquete no reconocido, debe avanzar la ventana hasta el próximo paquete aún sin reconocer

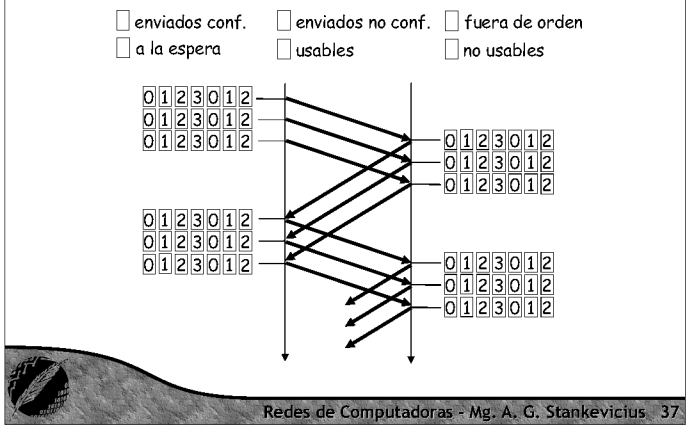
Receptor SR

- El receptor **SR** debe reaccionar ante los eventos que se presenten de la siguiente manera:
  - Al recibir un paquete con número de secuencia **base** se envía su **ACK** y se avanza la ventana hasta el próximo paquete esperado pero aún no recibido
  - Al recibir un paquete con número de secuencia entre **base+1** y **base+n-1** se envía su **ACK** y se guarda provisoriamente en el almacenamiento intermedio
  - Al recibir un paquete con número de secuencia entre **base-n** y **base-1** sólo se envía su respectivo **ACK**
  - En cualquier otro caso, no se toma acción alguna

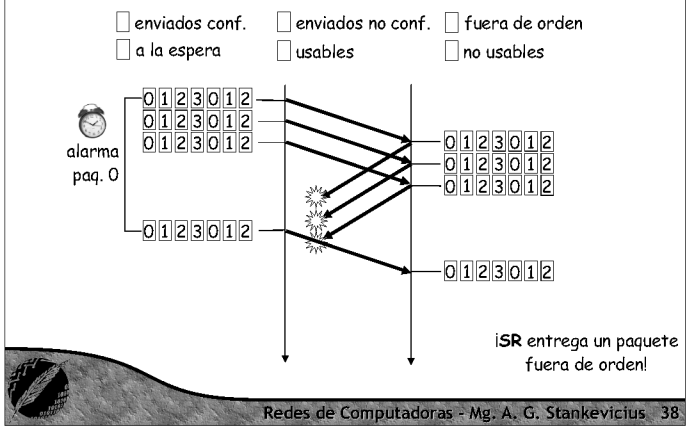
Protocolo SR en acción



El dilema del receptor SR



El dilema del receptor SR



Análisis SR

- El receptor **SR** no tiene forma de distinguir entre estos dos escenarios
  - En el primer caso el funcionamiento es el esperado, pero en el segundo caso, **SR** termina entregando un paquete fuera de orden
- ¿Qué relación tiene que cumplirse entre el tamaño de ventana y el conjunto de números de secuencia disponibles?
  - El tamaño de ventana debe ser menor o igual a la mitad de la cantidad de números de secuencia

¿Preguntas?