

Temas 3 y 4

6.263/16.37

La capa de enlace de datos: protocolos ARQ

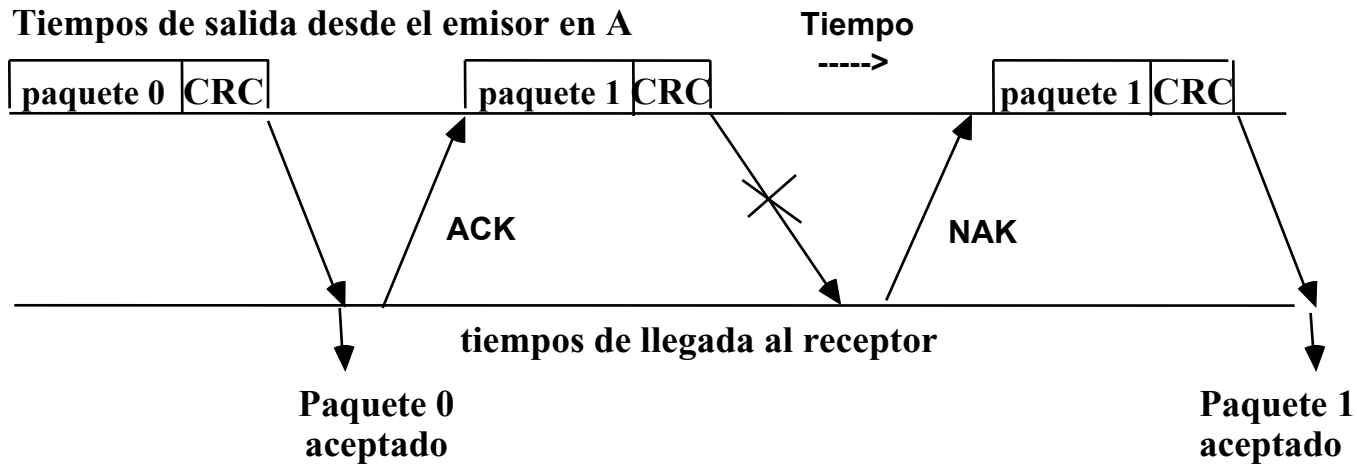
Eytan Modiano

MIT, LIDS

Solicitud de repetición automática (ARQ)

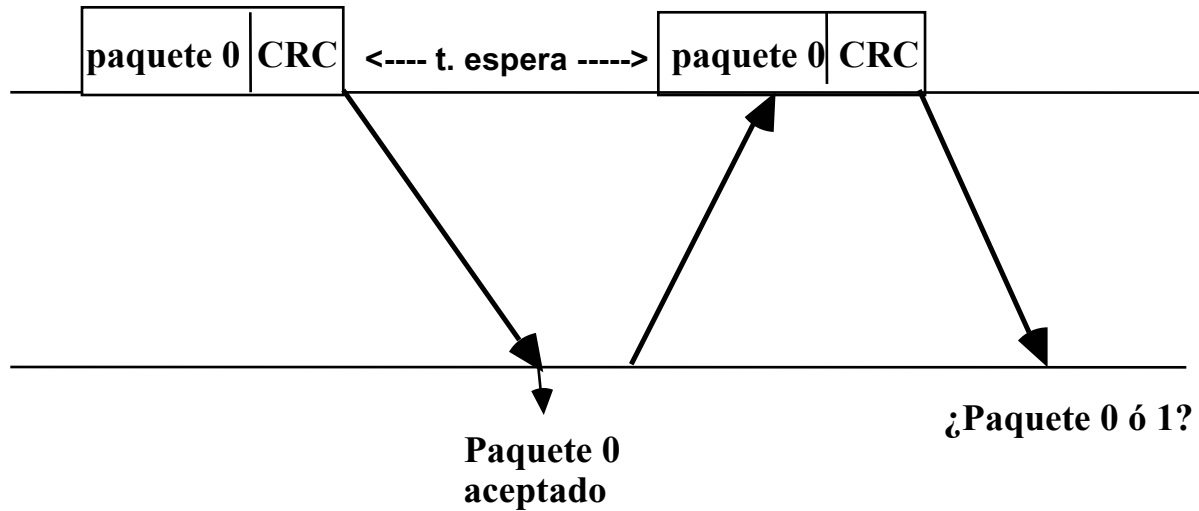
- Cuando el receptor detecta errores en un paquete, ¿cómo informa al emisor para que reenvíe el paquete correspondiente?
- Los sistemas que solicitan automáticamente la retransmisión de paquetes perdidos o con errores se denominan sistemas ARQ.
- Existen tres sistemas comunes:
 - Parada y espera (*Stop&Wait*)
 - Retroceso N (*Go Back N*)
 - Repetición selectiva

Protocolo puro de parada y espera



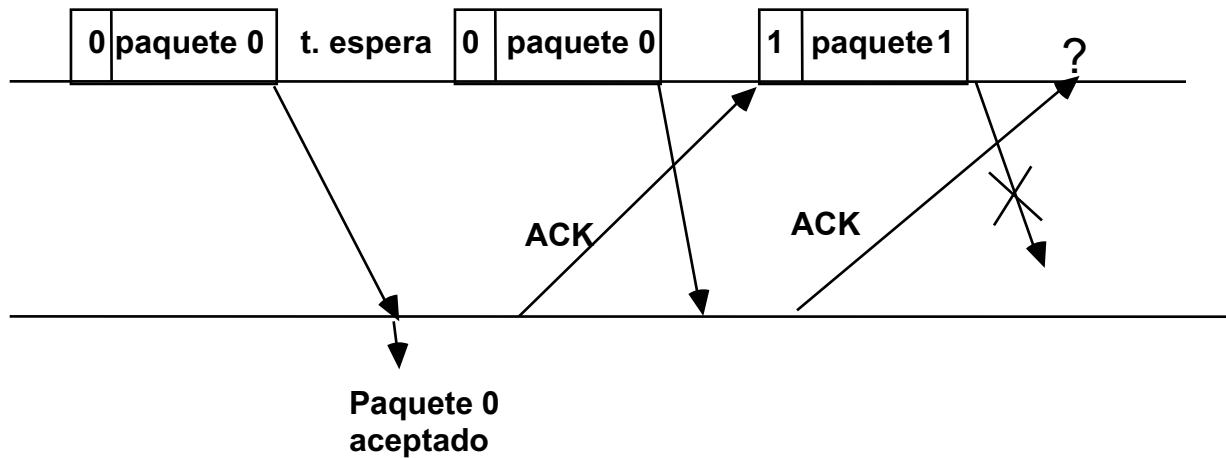
- Problema: pérdida de paquetes
 - El emisor permanecerá para siempre a la espera de una confirmación
- Los paquetes se pueden perder debido a errores en las tramas
- Solución: utilice el tiempo de espera (TO)
 - El emisor retransmite el paquete pasado un tiempo de espera

El uso de tiempos de espera para paquetes perdidos requiere números de secuencia



- Problema: a no ser que los paquetes estén numerados, el receptor no puede saber qué paquetes ha recibido
- Solución: utilizar paquetes numerados (números de secuencia)

Los números de petición son necesarios en las confirmaciones (ACK) para poder distinguir los paquetes



- **NÚMEROS DE PETICIÓN:**

- En vez de enviar un "ack" o un "nak", el receptor envía el número del paquete que espera en ese momento.
- Los números de secuencia y de petición se pueden enviar en módulo 2.

Esto funciona correctamente si suponemos que:

- 1) Las tramas viajan en orden (FCFS) por los enlaces
- 2) El CRC detecta los errores siempre
- 3) El sistema se inicializa adecuadamente

Protocolo de parada y espera

Algoritmo en el emisor (nodo A)

(con condición inicial SN=0)

- 1) Acepta el paquete de una capa superior, si está disponible; le asigna un número SN
- 2) Transmite el SN del paquete en una trama con número de secuencia SN
- 3) Espera una trama sin errores de B:
 - i. si lo recibe y contiene $RN > SN$ en el campo del número de petición, establece SN en RN y va a 1
 - ii. si no lo recibe en un tiempo dado, va a 2

Parada y espera

Algoritmo en el receptor (nodo B)

(con condición inicial $RN=0$)

- 1) Siempre que se recibe de A una trama sin errores con un número de secuencia igual a RN , se entrega el paquete recibido a una capa superior y se incrementa RN .
- 2) En tiempos arbitrarios, dentro del retardo delimitado tras recibir de A cualquier trama sin errores, se transmite hacia A una trama con el RN en el campo del número de petición.

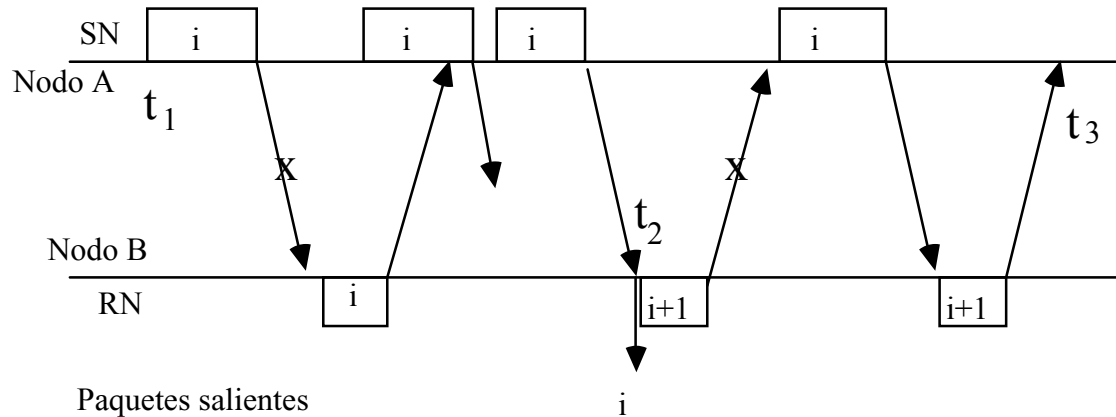
Corrección de parada y espera con SN y RN enteros

- Suponer que, en la transmisión desde A hasta B:
 - Todos los errores se detectan como tales
 - Inicialmente no hay tramas en el enlace: $SN=0$ y $RN=0$
 - Las tramas se pueden perder o retrasar arbitrariamente
 - Cada trama se recibe correctamente con, al menos, cierta probabilidad $q>0$.
- Dividir la prueba de corrección en dos partes:
 - **SEGURIDAD:** demuestra que no se ha entregado ningún paquete desordenado en más de una ocasión
 - **PROGRESO:** demuestra que todos los paquetes se entregan tarde o temprano

Seguridad

- Inicialmente no hay tramas en el enlace, el paquete 0 es el primer paquete aceptado en A, el único al que se le asigna $SN=0$ y deberá ser el que libere B, si B llega a liberar un paquete
- Posteriormente (mediante inducción), si B ha liberado paquetes hasta $n-1$ (incluido), RN se actualiza a n cuando se entrega $n-1$ y, a continuación, solo se podrá liberar n

PROGRESO



t_1 = tiempo en que A comienza a transmitir el paquete i

t_2 = tiempo en que B recibe y libera i correctamente e incrementa RN a $i+1$

t_3 = tiempo en que SN se incrementa a $i+1$

Demostraremos que $t_1 < t_2 < t_3 < \infty$. \Rightarrow Progreso

Discusión sobre el progreso

- Sean $SN(t)$ y $RN(t)$ valores de SN y RN en un tiempo t

Según el algoritmo:

- (1) $SN(t)$ y $RN(t)$ se incrementan en t y $SN(t) \leq RN(t)$ para todo t
- (2) Según la seguridad (dado que i no se envía antes que t_1)
 $RN(t_1) \leq i$ y $SN(t_1) = i$

- De (1) y (2), deducimos que $RN(t_1) = SN(t_1) = i$
- RN se incrementa en t_2 y SN en t_3 , por lo que $t_2 < t_3$
- A transmite i repetidamente hasta t_3 y hasta t_2 cuando se recibe correctamente. Como $q > 0$, t_2 es finito
- B transmite $RN=i+1$ una y otra vez hasta que se recibe correctamente en t_3 ; $q > 0$ implica que t_3 es finito.

Corrección de parada y espera con SN y RN binarios (finitos)

- Suponer que las tramas viajan ordenadas por el enlace

Obsérvese que con SN y RN enteros:

$$SN=RN \quad (\text{de } t_1 \text{ a } t_2) \quad \text{ó} \quad (3)$$

$$SN=RN-1 \quad (\text{de } t_2 \text{ a } t_3) \quad (4)$$

Como las tramas viajan ordenadas, los números de secuencia que llegan a B y los números de petición que llegan a A aumentan, por lo que un solo *bit* puede resolver la ambigüedad entre (3) y (4)

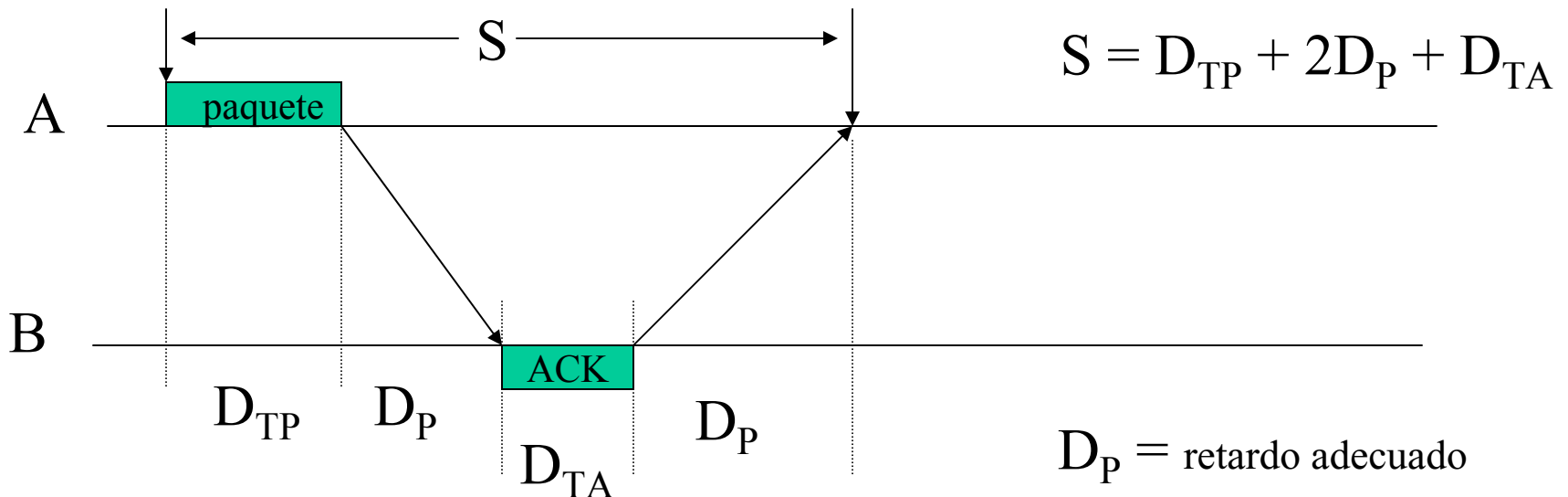
- $RN = 0$ y $SN = 1$ ó $RN = 1$ y $SN = 0$
=> el paquete recibido es un paquete antiguo
- $RN = 0$ y $SN = 0$ ó $RN = 1$ y $SN = 1$
=> el paquete recibido es nuevo

Eficiencia del sistema de parada y espera

Sea S = el tiempo total entre la transmisión de un paquete y la recepción de su confirmación (ACK)

D_{TP} = tiempo de transmisión del paquete

Eficiencia (sin errores) = D_{TP}/S



$$S = D_{TP} + 2D_P + D_{TA}$$

$$E = D_{TP} / (D_{TP} + 2D_P + D_{TA})$$

D_P = retardo adecuado

D_{TA} = T. de trans. del ACK

D_{TP} = T. de trans. del paquete

Parada y espera en presencia de errores

Sea P = la probabilidad de que se produzca un error en la transmisión de un paquete o en su confirmación

$$S = D_{TP} + 2D_P + D_{TA}$$

TO = el intervalo de tiempo de espera (*timeout*)

X = la cantidad de tiempo que lleva transmitir un paquete y recibir su confirmación.

Este tiempo se tiene en cuenta en las retransmisiones debidas a errores

$$E[X] = S + TO * P / (1 - P),$$

$$\text{Eficiencia} = D_{TP} / E[X]$$

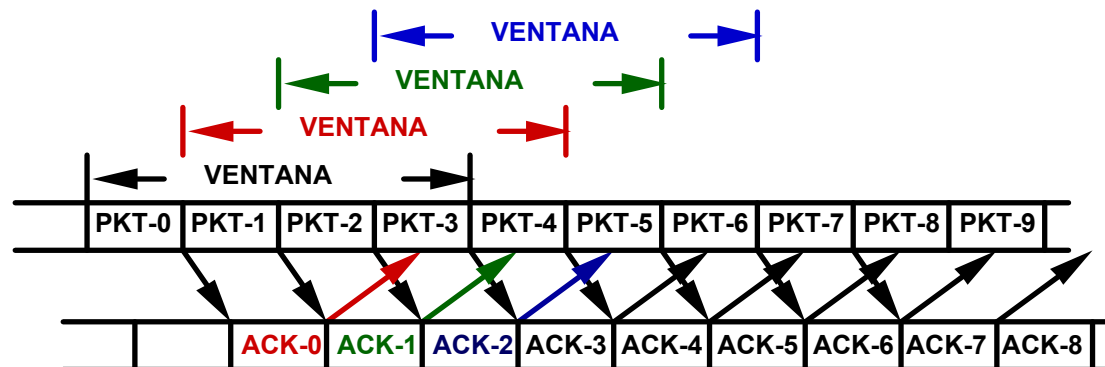
Donde:

TO = D_{TP} en un sistema *Full duplex* (bidireccional)

TO = S en un sistema *Half duplex* (de un solo sentido)

ARQ con retroceso N (ventana deslizante)

- El sistema de parada y espera es ineficaz si el retardo de propagación es mayor que el tiempo de transmisión de los paquetes:
 - Sólo es posible enviar un paquete cada RTT
- El sistema de retroceso N permite la transmisión de nuevos paquetes antes de que se confirmen los anteriores
- El retroceso N utiliza un mecanismo de ventanas en el que el emisor puede enviar los paquetes que se encuentran dentro de una “ventana” (intervalo):
 - La ventana avanza a medida que se confirman las recepciones de los paquetes anteriores



Características del Retroceso N

- Tamaño de la ventana = N:
 - El emisor no puede enviar el paquete $i+N$ hasta que haya recibido la confirmación (ACK) del paquete i
- El receptor funciona igual que en Parada y espera:
 - Recibe los paquetes ordenados
 - El receptor no puede aceptar paquetes fuera de la secuencia
 - Se envía $RN = i + 1 \Rightarrow$ confirma todos los paquetes hasta i (incluido)
- Uso de la superposición de confirmaciones (*piggybacking*):
 - Cuando el tráfico es bidireccional, se insertan los RN en los paquetes que van en la otra dirección:

Cada paquete contiene un campo SN que indica su número de secuencia y un campo RN que confirma los paquetes en la otra dirección

<-- Cabecera de la trama ----->

	SN	RN		Paquete	CRC
--	----	----	--	---------	-----

ARQ con Retroceso N

- El emisor tiene una "ventana" de N paquetes que puede enviar sin confirmación
- Esta ventana abarca desde el último valor RN obtenido del receptor (llamado $SN_{\text{mín}}$) a $SN_{\text{mín}} + N - 1$
- Cuando el emisor llega al final de su ventana o finaliza el tiempo de espera, retrocede y retransmite el $SN_{\text{mín}}$

Sea $SN_{\text{mín}}$ el paquete con menor número aún no confirmado

Sea $SN_{\text{máx}}$ el número del siguiente paquete que se aceptará desde la capa superior (es decir, el siguiente paquete nuevo que se transmitirá)

Retroceso N

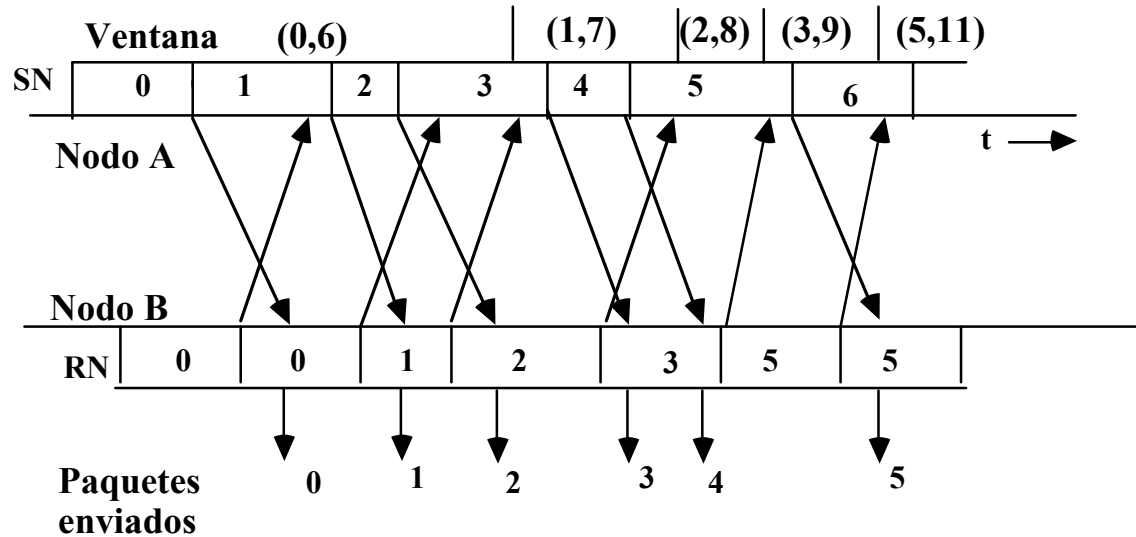
Reglas del emisor

- $SN_{\text{mín}} = 0; SN_{\text{máx}} = 0$
- Repetir:
 - Si $SN_{\text{máx}} < SN_{\text{mín}} + N$ (toda la ventana aún sin enviar)
Enviar paquete $SN_{\text{máx}}$;
 $SN_{\text{máx}} = SN_{\text{máx}} + 1;$
 - Si el paquete llega del receptor con $RN > SN_{\text{mín}}$
 $SN_{\text{mín}} = RN;$
 - Si $SN_{\text{mín}} < SN_{\text{máx}}$ (sigue habiendo paquetes sin confirmar) y el emisor no puede enviar paquetes nuevos
Elegir algún paquete entre $SN_{\text{mín}}$ y $SN_{\text{máx}}$ y reenviarlo
- La última regla dice que, cuando no es posible enviar paquetes nuevos, se debe reenviar un paquete antiguo (aún sin confirmar):
 - Puede haber dos razones para no poder enviar un paquete nuevo:
No se recibe ninguno nuevo de la capa superior
La ventana ha caducado ($SN_{\text{máx}} = SN_{\text{mín}} + N$)
 - No hay ninguna regla para saber qué paquete se debe reenviar
Enviar el menos reciente

Reglas del receptor

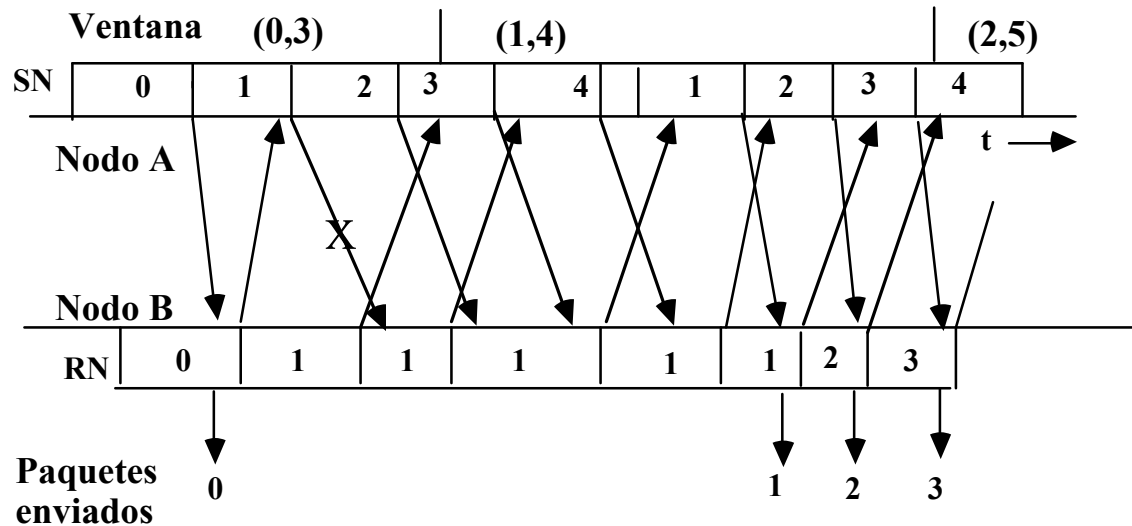
- $RN = 0$;
- Repetir:
 - Cuando llega un paquete correcto, si $SN = RN$:
 - Acceptar el paquete
 - Incrementar $RN = RN + 1$
- En intervalos regulares, enviar un paquete de confirmación con RN:
 - La mayoría de los DLC envían una confirmación cada vez que reciben un paquete desde la otra dirección:
 - Confirmación retardada para ser insertada en el siguiente mensaje de vuelta (*piggybacking*)
- El receptor rechaza todos los paquetes con SN distinto de RN:
 - No obstante, estos paquetes pueden seguir conteniendo números RN útiles (consultar el apartado Trabajos)

Ejemplo de ARQ con Retroceso 7



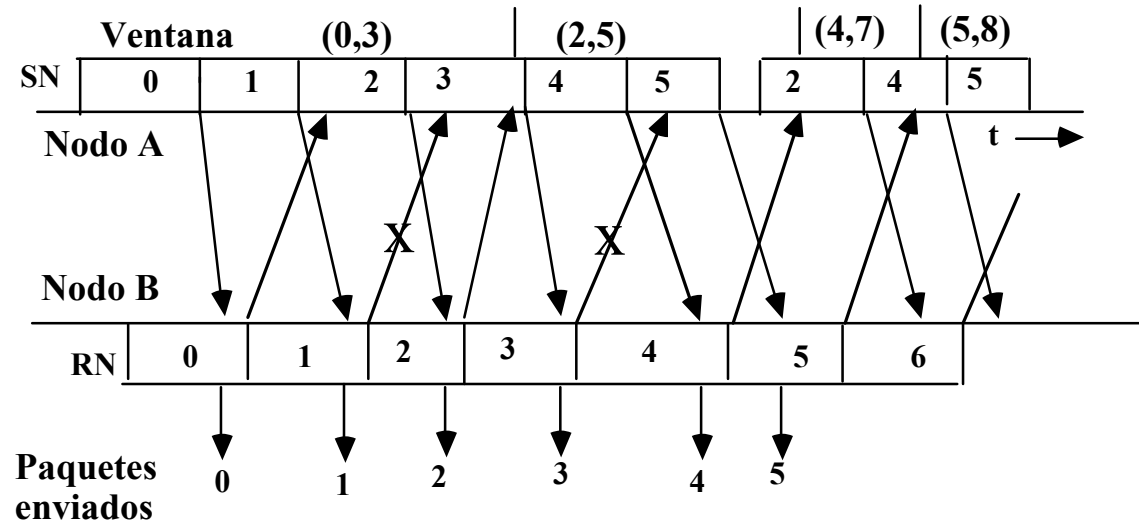
- Obsérvese que el paquete RN-1 debe ser aceptado por B antes de que se pueda iniciar la transmisión en B de una trama que contenga la petición RN

RETRANSMISIÓN DEBIDA A ERRORES REALIZADA POR UN ARQ CON RETROCESO 4



- Observe que el valor del tiempo de espera aquí es el tiempo que se tarda en enviar toda una ventana de paquetes
- Observe también que, tras un error, hay que retransmitir toda la ventana

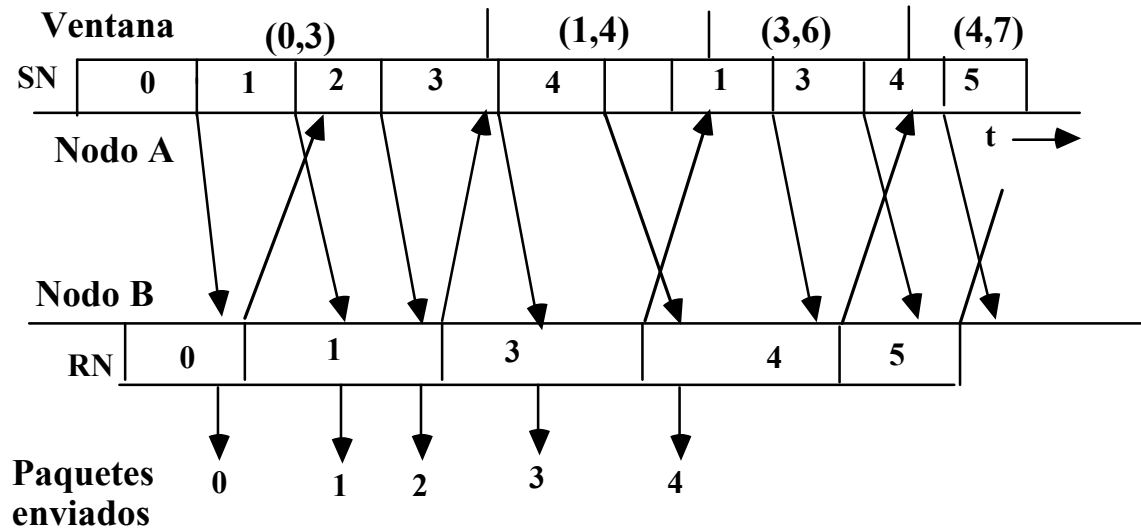
RETRANSMISIÓN POR ERRORES DE *FEEDBACK* REALIZADA POR UN ARQ CON RETROCESO 4



- Cuando se produce un error en dirección inversa, la confirmación aún puede llegar a tiempo. Es el caso anterior, en el que un paquete con RN=2 que va de A a B llega a tiempo para evitar la retransmisión del paquete 0
- El paquete 2 se retransmite, porque RN=4 no llegó a tiempo, pero sí lo hizo para evitar la retransmisión del paquete 3
 - ¿Era realmente necesaria la retransmisión de los paquetes 4 y 5?

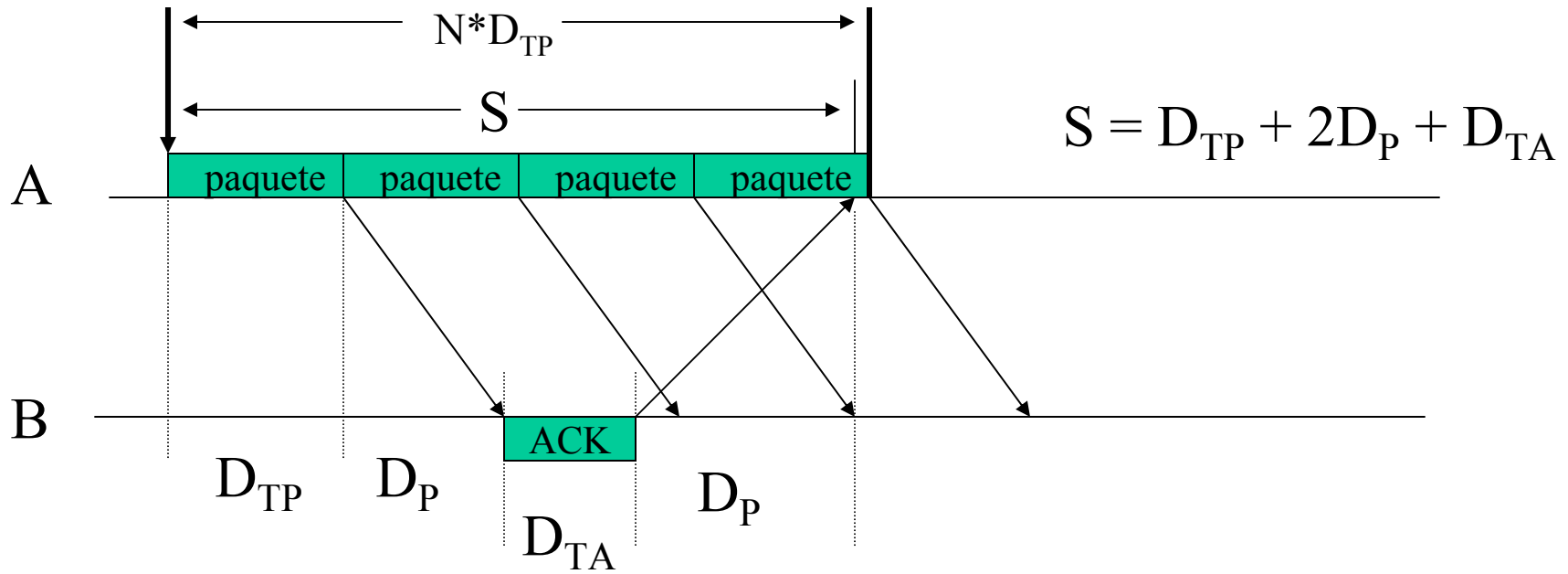
Estrictamente hablando, no, ya que la ventana permite la transmisión de los paquetes 6 y 7 antes que el resto de las retransmisiones. No obstante, esto depende de la implementación

EFECTO DE LAS TRAMAS LARGAS



- Las tramas largas en la dirección de *feedback* ralentizan las confirmaciones
 - Esto hace que un emisor con tramas cortas deba esperar o retroceder
- Obsérvese, una vez más, que la retransmisión de los paquetes 3 y 4 no era estrictamente necesaria, ya que el emisor podría haber enviado paquetes nuevos dentro de la ventana:
 - Una vez más, esto depende de la implementación

Eficacia del Retroceso N



- Queremos elegir un N lo suficientemente grande como para permitir la transmisión continua mientras espera una confirmación para el primer paquete de la ventana:

$$N > S / D_{TP}$$

- Sin errores, la eficacia del Retroceso N es:

$$E = \min\{1, N \cdot D_{TP} / S\}$$

Eficacia del Retroceso N con transmisión de errores

Análisis aproximado

Suponemos que: $N = \left\lceil \frac{S}{D_{TP}} \right\rceil$ $TO = N * D_{TP}$

- Cuando se produce un error, hay que retransmitir toda la ventana de N paquetes

Sea X = el número de paquetes enviados por cada transmisión correcta

$$E[X] = 1*(1-P) + (X+N)*P$$

$$= 1 + N*P/(1-P)$$

$$\text{Eficacia} = 1/E[X]$$

Requisitos del Retroceso N

- El funcionamiento del Retroceso N está garantizado, independientemente de qué paquetes se seleccionen para repetir, si:
 - 1) El sistema se inicializa adecuadamente
 - 2) No se producen fallos en la detección de errores
 - 3) Los paquetes viajan en orden FCFS
 - 4) Existe una probabilidad positiva de recepción correcta
 - 5) El emisor reenvía ocasionalmente $S_{n_{\min}}$ (p.ej., en el t. de espera)
 - 6) El receptor envía el RN de vez en cuando

Notas sobre el Retroceso N

- No requiere almacenamiento de paquetes en el *buffer* del receptor
- El emisor debe almacenar en el *buffer* hasta N paquetes mientras espera su confirmación
- El emisor debe reenviar toda la ventana en caso de error
- Los paquetes se pueden numerar en módulo M, donde $M > N$:
 - Porque se pueden enviar simultáneamente N paquetes como máximo
- El receptor sólo puede aceptar los paquetes en orden:
 - El receptor debe enviar los paquetes ordenados a la capa superior
 - No puede aceptar el paquete $i+1$ antes que el paquete i
 - Con esto se elimina la necesidad de almacenar en el *buffer*
 - Y se introduce la necesidad de reenviar toda la ventana en caso de error
- El mayor problema del Retroceso N es esta necesidad de reenviar toda la ventana en caso de error, lo cual se debe al hecho de que el receptor sólo pueda aceptar los paquetes ordenados

Protocolo de repetición selectiva (SRP)

- La repetición selectiva intenta retransmitir únicamente los paquetes que se han perdido (debido a errores):
 - El receptor debe poder aceptar paquetes desordenados
 - Puesto que el receptor debe entregar los paquetes en orden a la capa superior, debe poder almacenar en *buffer* algunos paquetes
- Peticiones de retransmisión:
 - Implícitas:

El receptor confirma todos los paquetes correctos; los paquetes que no se han confirmado antes del tiempo de espera se dan por perdidos o con errores

Obsérvese que se debe utilizar este enfoque para asegurarse de que finalmente se reciben todos los paquetes
 - Explícitas:

Un NAK explícito (rechazo selectivo) puede solicitar la retransmisión de un solo paquete

Este enfoque puede acelerar la retransmisión, pero no es estrictamente necesario
 - En la práctica, se emplean uno o ambos enfoques

Reglas del SRP

- Protocolo de ventanas similar al del Retroceso N:
 - Tamaño W de ventana
- Los paquetes se numeran en módulo M , donde $M \geq 2W$
- El emisor puede transmitir paquetes nuevos siempre que su número sea W en todos los paquetes sin confirmar
- El emisor retransmite los paquetes no confirmados tras un tiempo de espera:
 - O después de un NAK (en caso de utilizar NAK)
- El receptor confirma todos los paquetes correctos
- El receptor almacena los paquetes correctos hasta que puedan enviarse ordenados a la capa superior

Necesidad de almacenar en *buffer*

- El emisor debe almacenar en el *buffer* todos los paquetes hasta que sean confirmados:
 - Es posible almacenar hasta W paquetes sin confirmar
- El receptor debe almacenar los paquetes hasta que pueda enviarlos en orden:
 - es decir, hasta que se hayan recibido todos los paquetes de menor número
 - Necesidad de enviar los paquetes en orden a la capa superior
 - Puede que sea necesario almacenar hasta W paquetes (en caso de que se pierda el primer paquete de una ventana)
- Implicación del tamaño del *buffer* = W
 - Número de paquetes no confirmados en el emisor $\leq W$
Buffer limitado en el emisor
 - El número de paquetes no confirmados en el receptor no puede diferir en más de W
Buffer limitado en el receptor (necesidad de enviar paquetes ordenados)
 - Los paquetes deben numerarse en módulo $M \geq 2W$ (utilizando $\log_2(M)$ *bits*)

EFICACIA

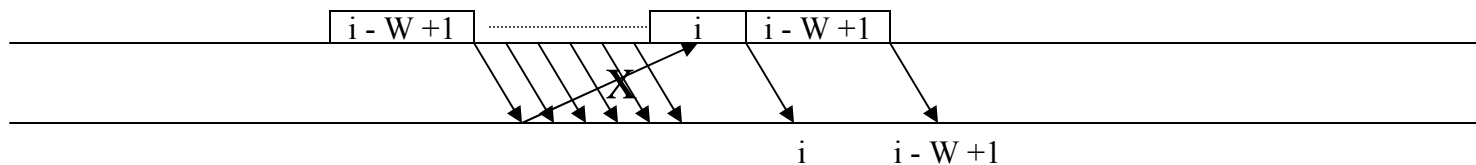
- En un SRP ideal, sólo se retransmiten los paquetes con errores:
 - Lo ideal no es realista, ya que, en ocasiones, los paquetes se deben retransmitir porque la ventana ha caducado. No obstante, si el tamaño de la ventana es mucho mayor que el valor del tiempo de espera, esto es poco probable
- Con un SRP ideal, la eficacia = $1 - P$
 - P = probabilidad de un error en paquetes
- Obsérvese la diferencia con el Retroceso N , donde:

$$\text{eficacia (Retroceso } N) = 1 / (1 + N * P / (1 - P))$$

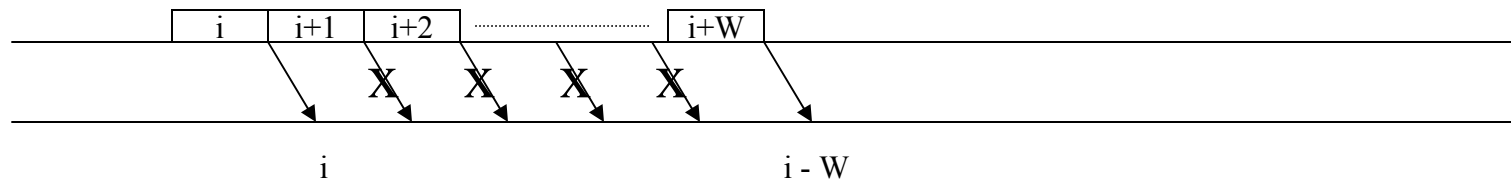
- Cuando el tamaño de la ventana es pequeño el rendimiento es parecido, pero con una ventana grande, el SRP es mucho mejor:
 - A medida que aumentan las tasas de transmisión, se necesitan ventanas más grandes y, en consecuencia, un mayor uso del SRP

¿Por qué se numeran los paquetes en módulo $2W$?

- Examinemos el intervalo de paquetes que pueden seguir al paquete i en el receptor:



El paquete i puede ir seguido del primer paquete de la ventana ($i-W+1$) si requiere retransmisión



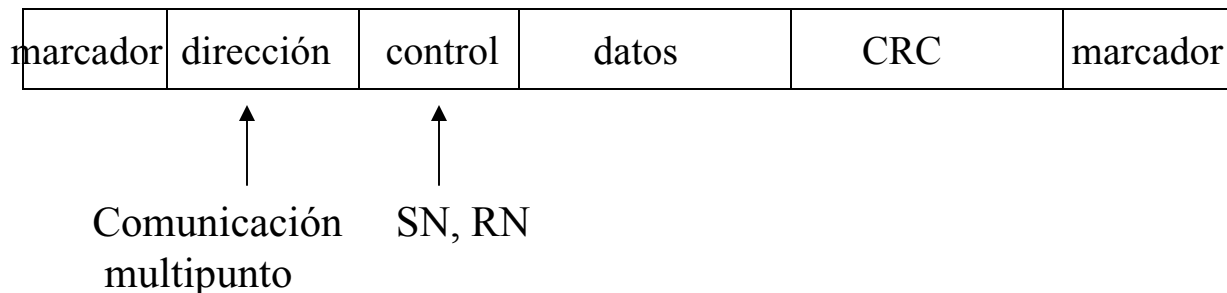
El paquete i puede ir seguido del último paquete de la ventana ($i+W$) si se pierden todas las confirmaciones entre i y $i+W$

- El receptor debe distinguir entre paquetes $i-W+1 \dots i+W$
 - Estos paquetes $2W$ se pueden distinguir empleando la numeración en mod $2W$

DLC ESTÁNDAR

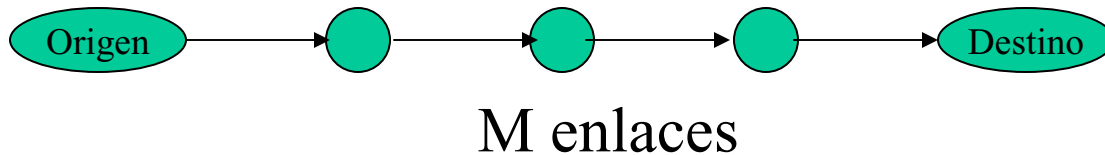
- HDLC, LAPB (X.25) y SDLC son prácticamente iguales:
 - HDLC/ SDLC desarrollado por IBM para redes SNA de IBM
 - LAPB desarrollado para redes X.25
- Todos utilizan un entramado orientado a *bits* con el marcador = 01111110
- Todos utilizan un CRC de 16 *bits* para la detección de errores
- Todos utilizan ARQ con Retroceso N con un N = 7 ó 127 (opcional)

Paquete SDLC



- Los protocolos más antiguos (empleados para módems, p.ej.: xmodem) utilizaban el sistema de Parada y espera y comprobaciones de sumas sencillas

Tamaño óptimo de paquetes basado en el efecto de canalización (*pipelining*)



- Se debe recibir el paquete completo antes de reenviarlo al siguiente nodo
- Retardo para el envío de N paquetes por M enlaces (retardo de canalización)

$$D = N \cdot D_{TP} + (M-1) \cdot D_{TP}$$

- Cada paquete contiene K *bits* de datos y una cabecera de H *bits*
 - CRC, marcadores, SN, etc.
 - Tamaño total del paquete: $K+H$ *bits*
- Para transmitir un mensaje de L *bits* necesitamos L/K paquetes
- Tiempo para transmitir el mensaje a través de M enlaces:

R = tasa de datos

$$D = \frac{L}{K} \left(\frac{K+H}{R} \right) + (M-1) \left(\frac{K+H}{R} \right)$$

Tamaño óptimo del paquete

Retardo de transmisión

Retardo de canalización

$$D = \frac{L}{K} \left(\frac{K+H}{R} \right) + (M-1) \left(\frac{K+H}{R} \right)$$

- Los paquetes pequeños reducen el retardo de canalización, pero aumentan el de transmisión debido a las cabeceras adicionales
- Los paquetes grandes reducen la sobrecarga en las cabeceras, pero aumentan el retardo de canalización

- Tamaño óptimo del paquete: $K_{opt} = \sqrt{\frac{LH}{M-1}}$
- El enfoque puede ser adecuado para redes multisalto de alta velocidad
- Un enfoque alternativo puede optimizar el tamaño de los paquetes para minimizar las retransmisiones de capa de enlace por errores
 - Los paquetes grandes son más propensos a contener errores de transmisión