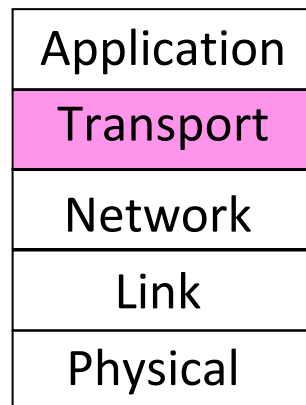


Capítulo 3

Capa de Transporte Transferencia de datos confiable



Metas

- Ejercitaremos los siguientes asuntos:
 1. Entrega de datos confiable
 2. Protocolo de parada y espera
 3. Protocolos de tubería
 4. Control de flujo en la capa de transporte
 5. Control de flujo en TCP

Entrega de datos confiable

- **Situación:** con mecanismos simples como confirmaciones de recepción, temporizadores y retransmisiones quedan lagunas en como se hace la entrega de datos confiables.
- **Por ejemplo, quedan preguntas sin contestar como las siguientes:**
 - ¿Qué se confirma exactamente en un ACK?
 - ¿cuántos paquetes se envían antes de recibir un ACK?
 - ¿Qué hace el receptor cuando se pierde o daña un paquete?
- Para responder preguntas como estas hay que definir un **protocolo de entrega de datos confiable**.
- Veremos que hay varios protocolos de estos, porque hace falta **optimizar el rendimiento según las características de la red**.
 - Con características de la red nos referimos a: latencia, proporción de errores, y capacidad de la red – tasa de datos que puede manejar.

Entrega de datos confiable

- La capa de transporte debe soportar al menos un protocolo para **entrega de datos confiable**.
- Estudiaremos distintos protocolos de entrega de datos confiable.
 - Los protocolos irán desde los más simples a los más complejos.
- **Estos protocolos asumen que el canal puede:**
 - Corromper paquetes
 - Perder paquetes
 - La transferencia de datos es en un sentido, o sea hay un emisor y un receptor.
- El protocolo más simple que vemos es el de **parada y espera**. Luego veremos protocolos más complejos llamados de **tubería**.
- Estos protocolos se pueden usar tanto en capa de transporte como en capa de enlace de datos.
 - Pues entrega confiable de datos es un problema de esas capas.

Preliminares

- Dijimos que la CT se ocupa de uso de temporizadores y retransmisiones de paquetes.
 - Paquetes perdidos deben **retransmitirse**.
- **Sabemos que un paquete no se perdió**
 - porque fue confirmado con un **paquete de confirmación de recepción**.
- **¿Cómo sabemos que un paquete se perdió?**
 - Podemos asumir que si pasa un cierto tiempo y no fue confirmado entonces se perdió y hay que retransmitirlo.
- **Para medir el tiempo:**
 - Usar **temporizadores** (timers)

Preliminares

- **Situación:** Se perdió una confirmación de recepción y se envió el paquete de nuevo.
- **Problema:** El mismo paquete llega dos o más veces al receptor y la capa de transporte la pasa a la capa de aplicación más de una vez.
 - ❑ Esto es inaceptable
 - ❑ ¿Cómo evitar entregar a la capa de aplicación paquetes repetidos?
- **Solución:** asignar **números de secuencia** a los paquetes que salen.
 - ❑ La idea es que dado un número de secuencia de un segmento que acaba de llegar,
 - ❑ el receptor puede usar ese número de secuencia para decidir si el segmento es un duplicado y en ese caso descartarlo.

Metas

- Ejercitaremos los siguientes asuntos:
 1. Entrega de datos confiable
 2. **Protocolo de parada y espera**
 3. Protocolos de tubería
 4. Control de flujo en la capa de transporte
 5. Control de flujo en TCP

Protocolos para entrega de datos confiable

- **Situación:** la latencia es baja.
 - El RTT es bajo comparado con el tiempo de copiar un paquete.
 - Por lo tanto solo se puede mandar un paquete antes que llegue el ACK.
- Un protocolo óptimo para manejar esta situación se llama **parada y espera**.

Protocolo de Parada y Espera

- **Suposición:** el canal de comunicaciones subyacente puede perder paquetes (de datos, de ACKs)
 - Los paquetes tienen N° de secuencias.
 - Con 1 bit es suficiente.
 - Se trabaja con Acks
 - El receptor debe especificar N° de secuencia del paquete siendo confirmado.
 - Se usan retransmisiones de paquetes.
 - Para esto se requiere de uso de temporizadores.

Comportamiento del emisor:

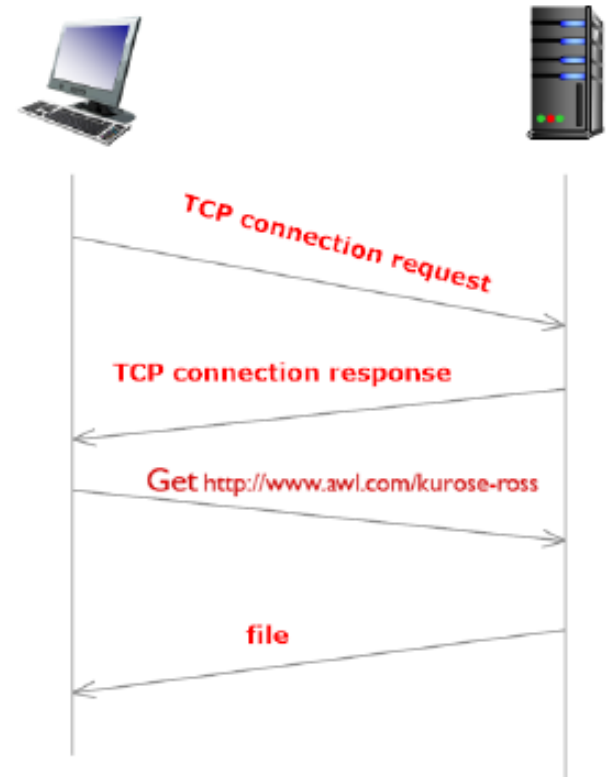
1. El emisor envía paquete P y **para** de enviar.
2. **Espera:** El emisor espera una cantidad “razonable” de tiempo para el ACK
3. Si llega el ACK a tiempo, se envía siguiente paquete. Goto 2.
4. Sino se retransmite paquete P. Goto 2.
- Si hay paquete o ACK demorado pero no perdido:
 - La retransmisión va a ser un duplicado con igual número de secuencia ; luego se descarta en el receptor.

Protocolos para entrega de datos confiable

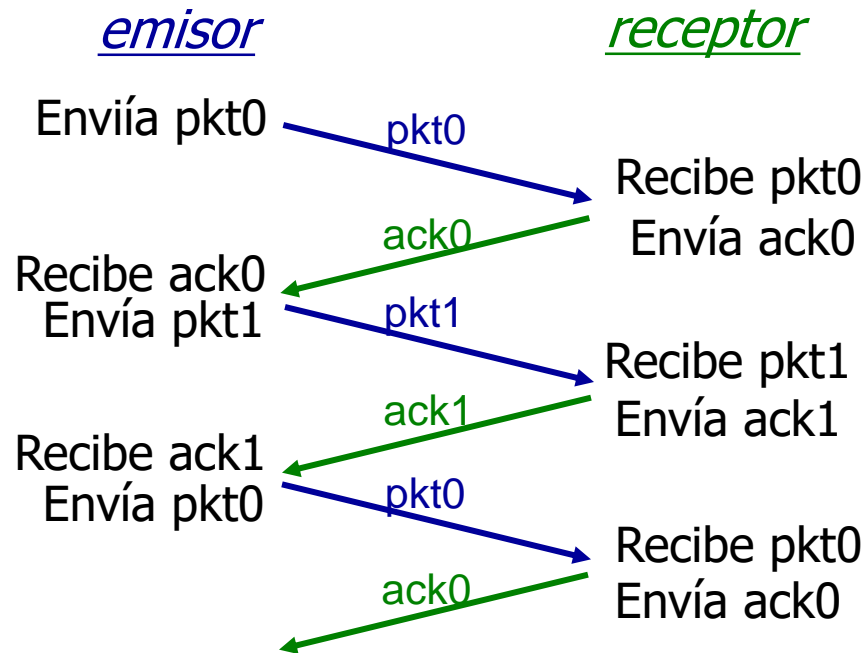
Diagramas de secuencia de mensajes

- Hay máquinas con diferentes roles
- Mensaje entre par de máquinas
- Se describe de arriba hacia abajo secuencia de esos mensajes

Ejemplo: figura a la derecha

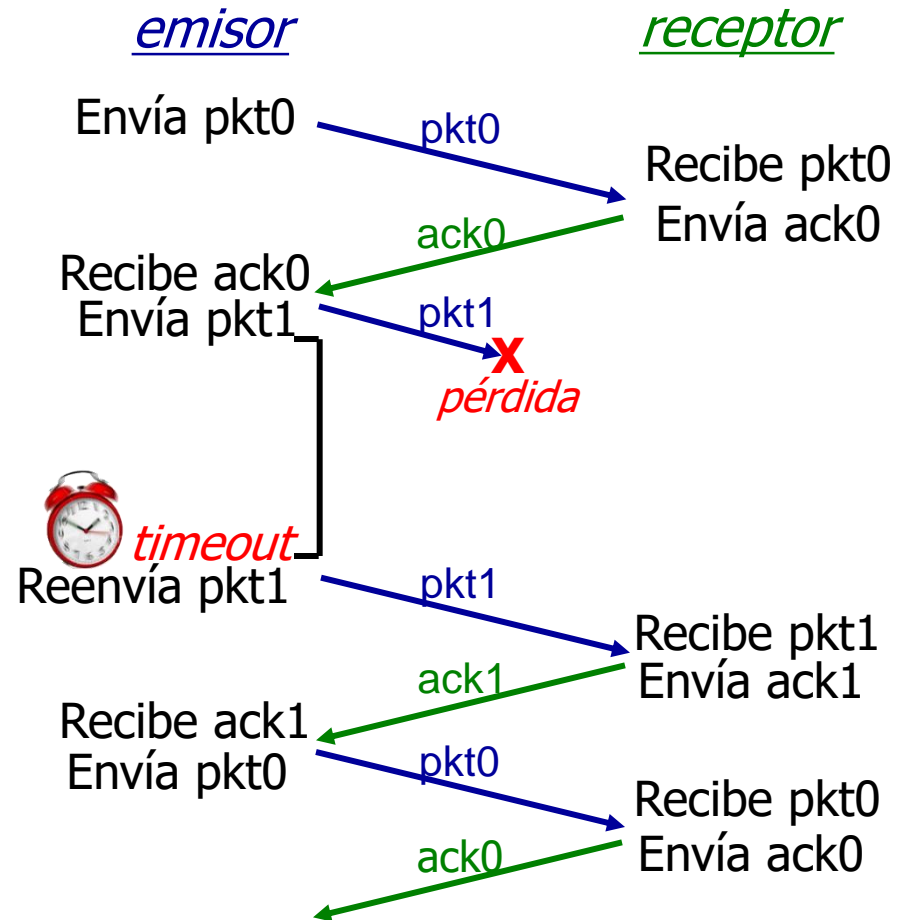


Parada y Espera en Acción



(a) Sin pérdida

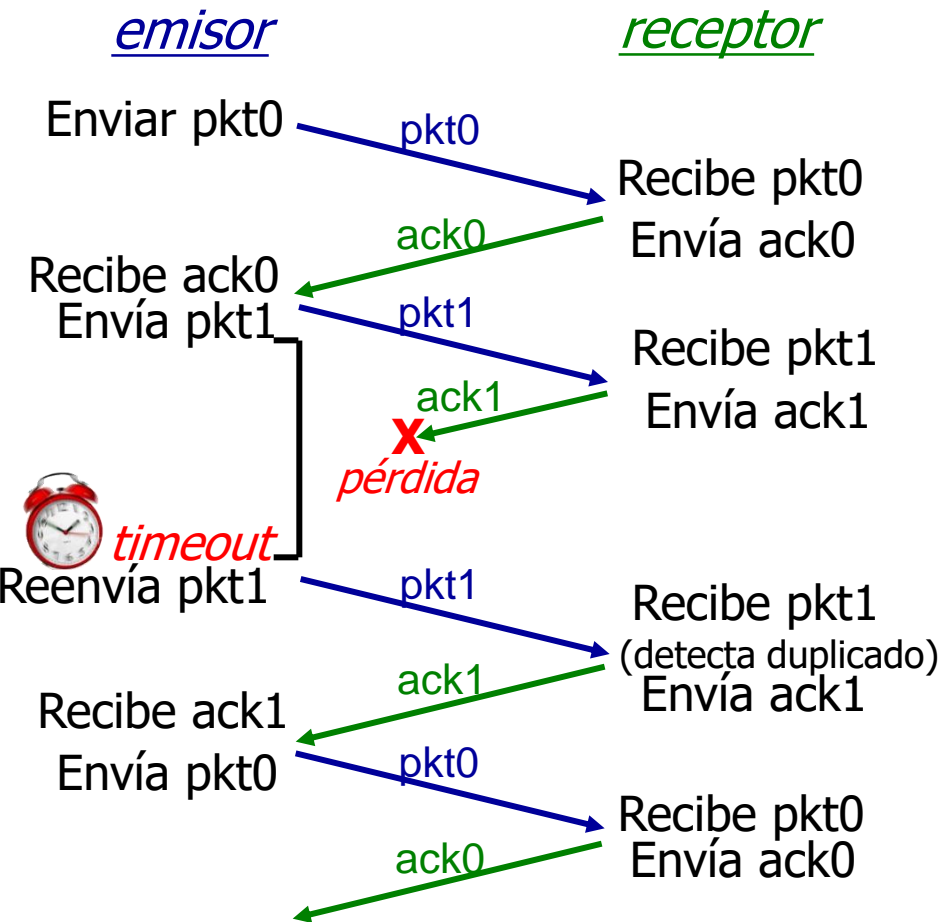
- ¿Qué pasa si se pierde el pkt1?



(b) Pérdida de paquete

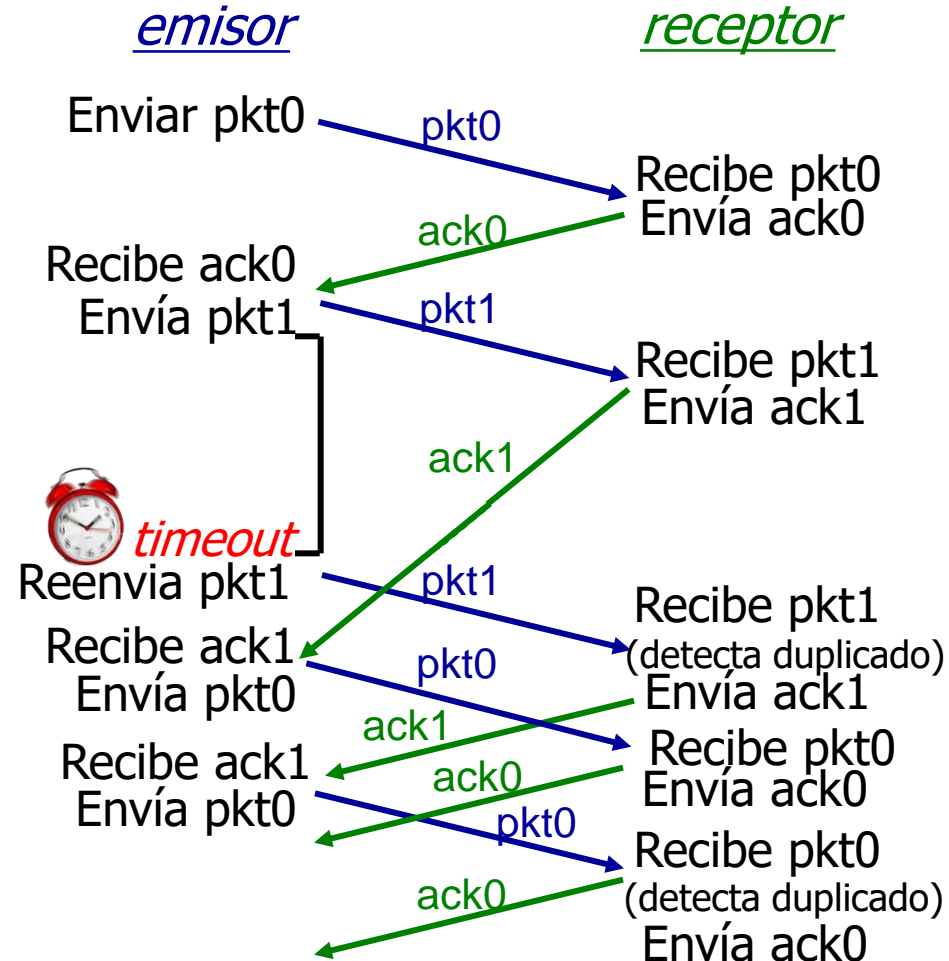
Parada y Espera en Acción

- ¿Qué pasa si se pierde el ack1?



(c) Pérdida de ACK

- ¿Qué pasa si el ack1 se demora?



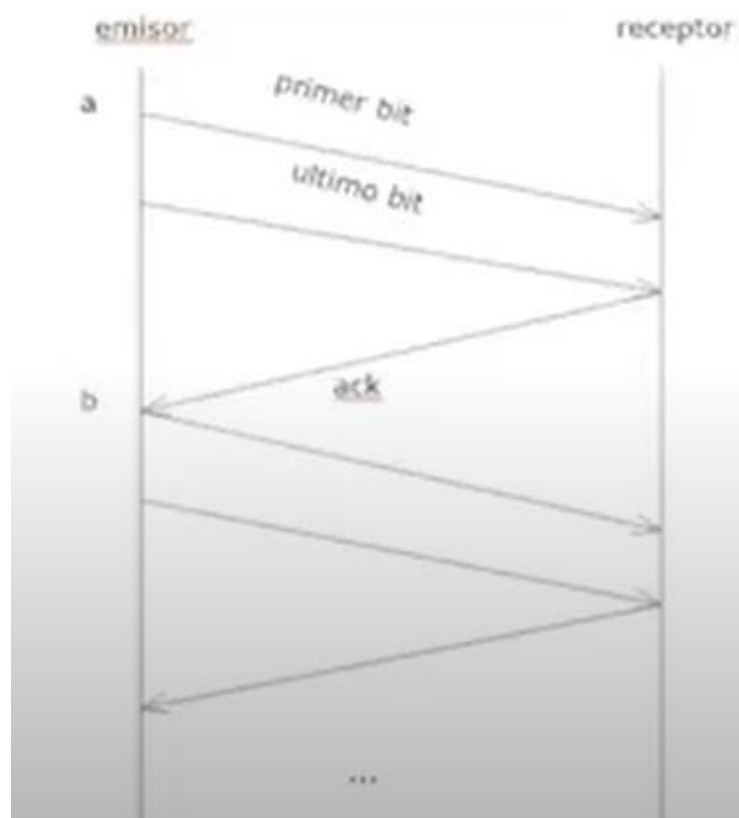
(d) Timeout prematuro/ ACK demorado

Protocolo de parada y espera

- ¿Cómo evaluar un protocolo para comunicación confiable?
- Podemos **simplificar un poco las cosas** y además hacer un **análisis de mejor caso**.
- **Para simplificar asumir que:**
 - Hay un canal de comunicación que une el emisor con el receptor.
 - Por ejemplo, un cable.
- **Para el mejor caso asumir que:**
 - La transmisión entre emisor y receptor es sin errores (no se pierden paquetes, no se demoran paquetes, no se alteran paquetes en curso).

Protocolo de parada y espera

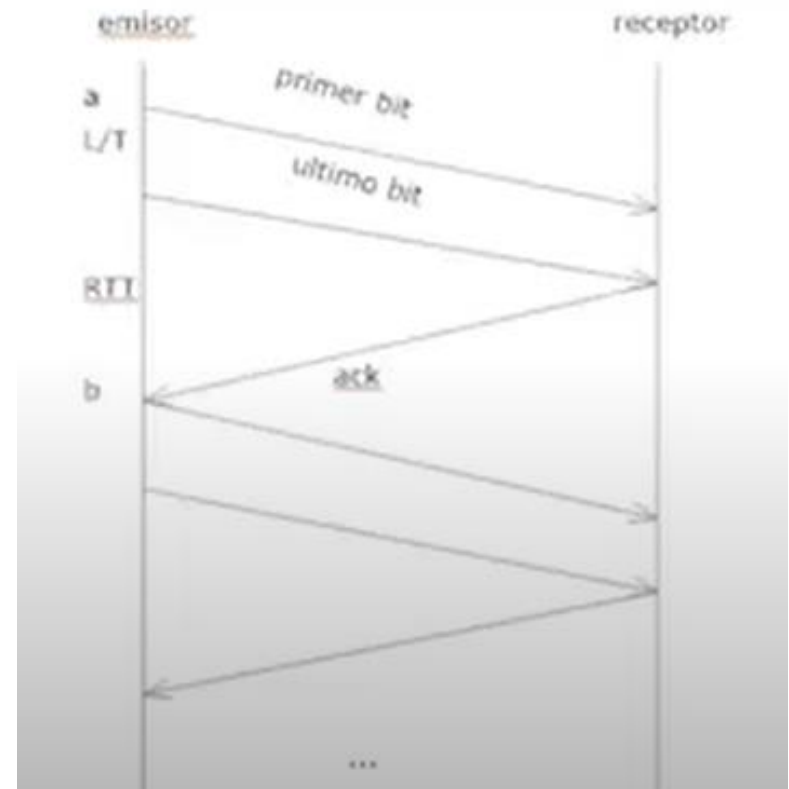
- Si L es la longitud de los segmentos y T es la tasa de transmisión del canal (en bits por segundo).
 - ¿Cómo se calcula tiempo de transmisión del mensaje?
- **RTT** es el tiempo entre la salida del último bit del mensaje y la llegada del primer bit del ack.
 - Se puede calcular a partir de D , la **demora de propagación del emisor al receptor**.
 - ¿Cuánto sería usando D ?



Protocolo de parada y espera

- Tiempo de transmisión del segmento: L/R
- $RTT = 2 * D$
- Utilización del canal U_{sender}

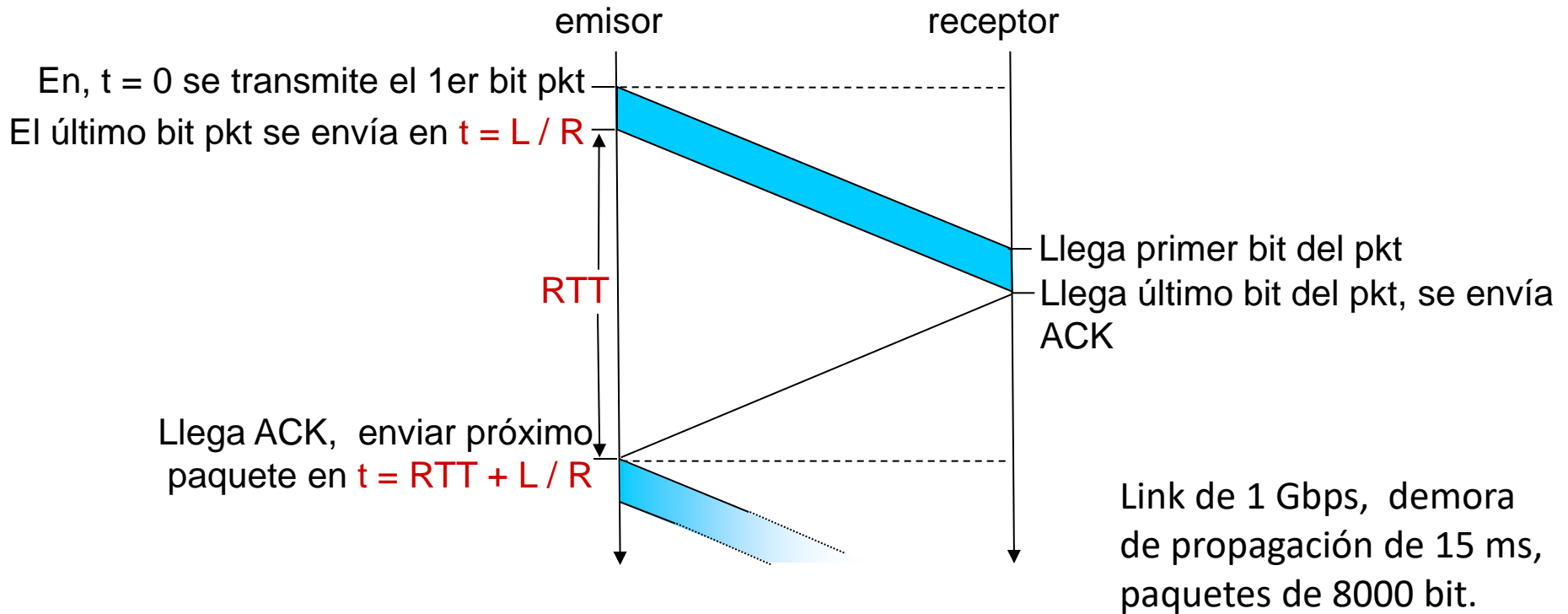
$$U_{\text{sender}} = \frac{L/R}{RTT + L/R}$$



Desempeño de Parada y Espera

- **Ejemplo:** link de 1 Gbps, demora de propagación de 15 ms, paquetes de 8000 bit:
- $D_{envío}$ es la demora en enviar un paquete.
- U_{sender} : **utilización** – es la fracción del tiempo en que el emisor está ocupado enviando.
- RTT es tiempo de ida y vuelta de un bit: RTT = 30 msec.
- El protocolo de red limita el uso de recursos físicos.

Desempeño de Parada y Espera



$$D_{\text{envío}} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits/sec}} = 8 \text{ microsecs}$$

Suposición: RTT fijo

$$U_{\text{sender}} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

Metas

- Ejercitaremos los siguientes asuntos:
 1. Entrega de datos confiable
 2. Protocolo de parada y espera
 3. **Protocolos de tubería**
 4. Control de flujo en la capa de transporte
 5. Control de flujo en TCP

Protocolos de tubería

- **Situación:** la latencia es alta.
 - El RTT es muy alto comparado con el tiempo de copiar un paquete.
 - Por lo tanto se pueden mandar varios paquetes antes que llegue el ACK del primer paquete enviado.
- Un protocolo adecuado para aprovechar esta situación se llama **protocolo de tubería**.

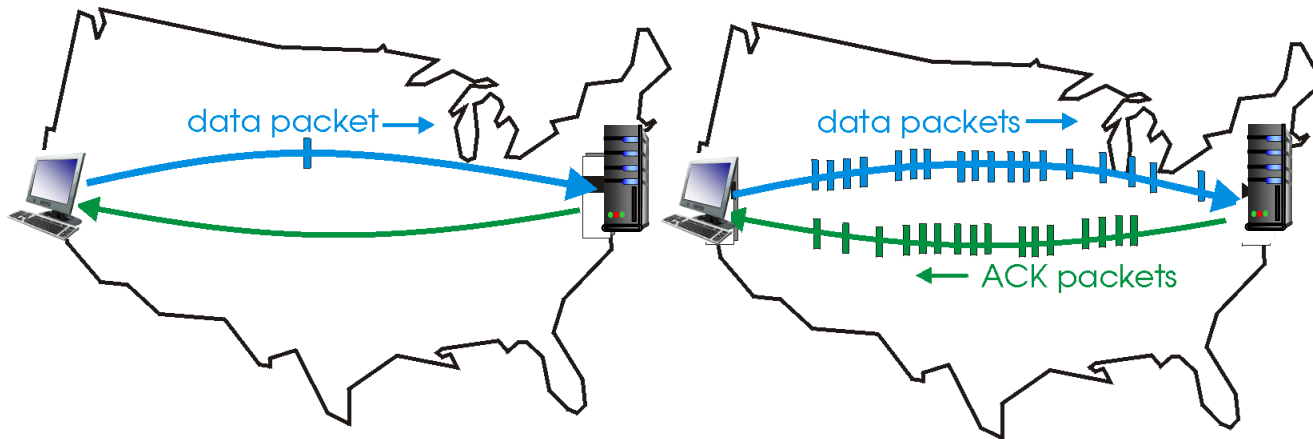
Protocolos de transferencia de datos confiable

- **Protocolos de tubería que estudiaremos**
 - Protocolo Retroceso-N
 - Protocolo de Repetición Selectiva

Protocolos de tubería

Tubería: el emisor puede enviar múltiples paquetes al vuelo a ser confirmados

- El rango de números de secuencia debe ser incrementado usando palabras de más de un bit.
- Hay que usar búferes en el emisor.

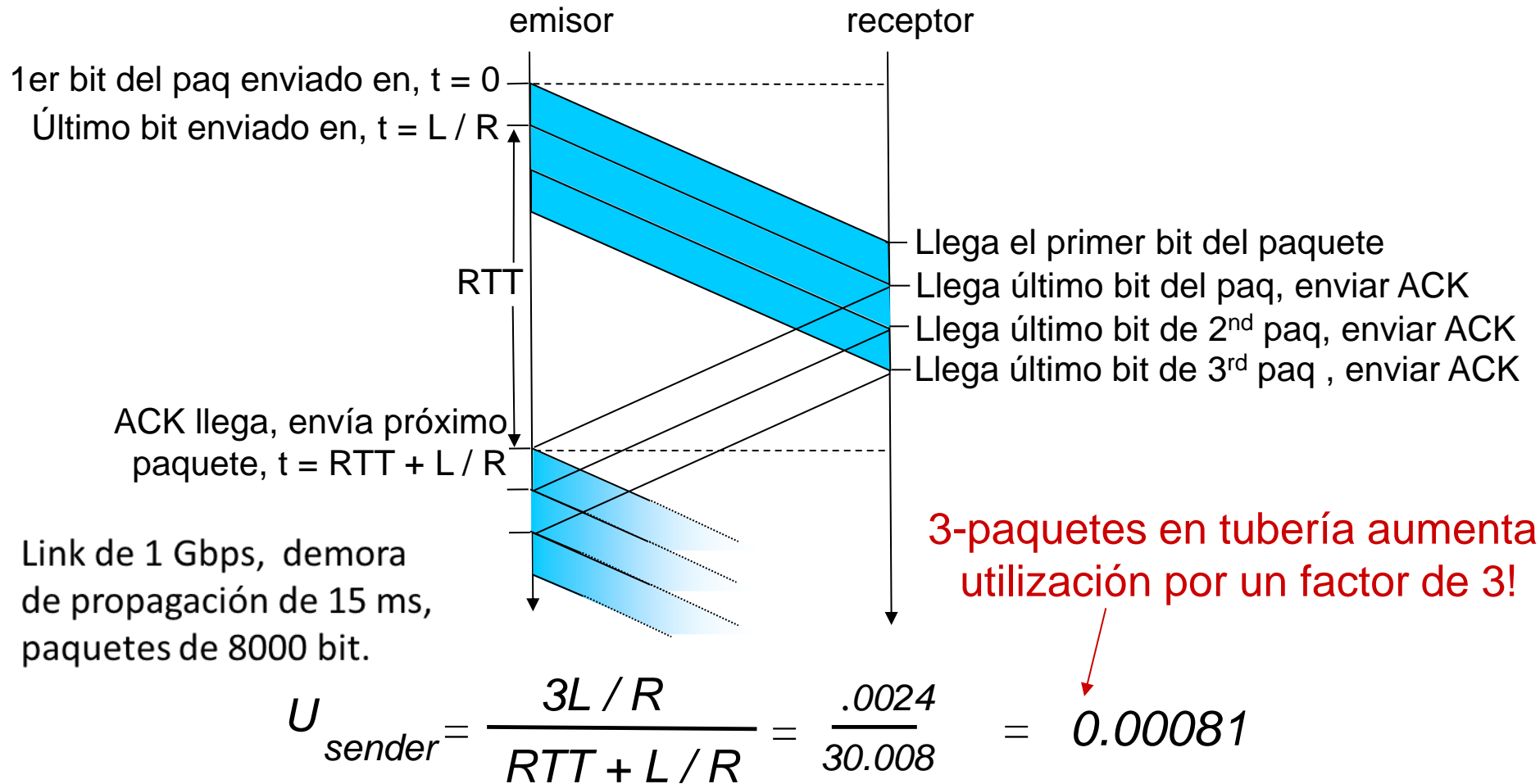


(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

- Hay dos formas genéricas de protocolos de tubería:
 - *retroceso N* y *repetición selectiva*

Tubería: utilización incrementada



Suposición: el RTT es fijo y no varía (p.ej: dos hosts unidos por cable).

Uso de búferes en el emisor

- La **ET emisora** **debe** manejar **búferes para los mensajes de salida**.
- **Esto es necesario porque:**
 - puede hacer falta retransmitirlos
- **¿Cómo se usan búferes en el emisor?**
 - El emisor almacena en búfer todas los segmentos hasta que se confirma su recepción.

Protocolos de transferencia de datos confiable

- **Protocolos de tubería**
 - **Protocolo Retroceso-N**
 - Protocolo de Repetición Selectiva

Protocolos de tubería

■ Situación:

- La latencia es grande.
- La proporción de errores o pérdida de paquetes es muy baja.
- Rara vez se demoran los paquetes.

■ Solución: si estos problemas rara vez se dan; entonces se puede hacer el código del receptor más **sencillo y eficiente**, usando la solución más fácil a estos problemas.

- Veremos el protocolo de retroceso N que la adopta.

■ Nota: si los problemas ocurren con cierta frecuencia, será necesario complicar el código del receptor para que se manejen eficientemente esos problemas.

Retroceso N

- Si un paquete T a la mitad de una serie larga se daña o pierde:
 - La CT receptora debe entregar paquetes a la capa de aplicación en secuencia.
 - Por lo que no se pueden entregar a la capa de aplicación los paquetes que llegaron **bien** después de T .
- Problema: ¿qué debe hacerse con los paquetes correctos que le siguen a un paquete que se perdió?

Retroceso N

Solución: Con retroceso N el receptor descarta todos los paquetes subsecuentes al paquete perdido, sin enviar ack para los paquetes descartados.

Retroceso N

■ Comportamiento del receptor:

- Receptor envía **ack acumulativo**
 - mayor número de secuencia tal que todos los segmentos anteriores se recibieron bien.
- Asumir que el receptor recibió un paquete n .
- Si n está en orden (todos los paquetes anteriores llegaron) y está correcto (sin errores):
 - manda ack para n y entrega parte de datos de paquete n a capa superior.
- Sino:
 - el receptor descarta el paquete n y manda ACK del paquete más reciente recibido en orden.

Retroceso N en el emisor

- **Comportamiento del emisor:**

- El emisor tiene un solo temporizador para el paquete más viejo no confirmado.
- Al expirar el temporizador (del segmento más viejo no confirmado),
 - retransmite todos los segmentos no confirmados.
- Si llega ACK nuevo y hay segmentos enviados no confirmados,
 - el temporizador es reiniciado.
- Si llega ACK nuevo y no hay segmentos sin confirmar,
 - el temporizador es detenido.

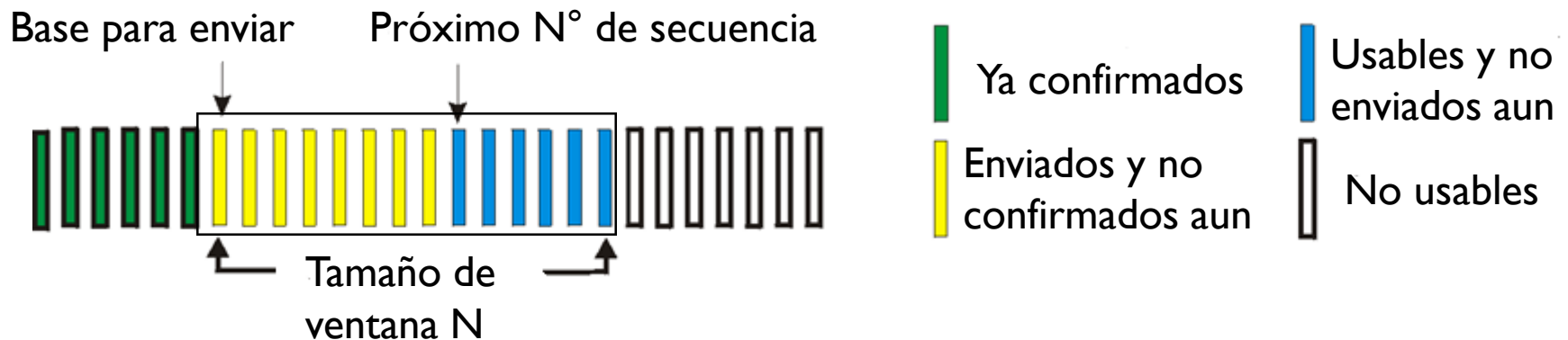
Retroceso N en el emisor

■ Asumimos en el emisor:

- Todos los búferes son del **mismo tamaño**
- Voy a tener una cantidad de búferes fija en el emisor
 - Asumiendo que el RTT es fijo esta cantidad representaría, la cantidad de segmentos a enviar por ráfaga para aprovechar el canal al máximo.
- Para referirnos a los números de secuencia de esos búferes usamos el concepto de **ventana del emisor**.

Retroceso-N: en el emisor

- La “ventana” permite hasta N paquetes consecutivos sin confirmar
- **ventana emisora** = tramas enviadas sin ack positivo o tramas listas para ser enviadas.



- *timeout(n)*: retransmite paquete n y todos los paquetes de mayor N° de secuencia en la ventana.

Retroceso N

- Si mando un paquete de confirmación solamente, ¿qué número de secuencia debe tener?
- Enviar ACK con N° de secuencia más alto tal que los N° de secuencia anteriores fueron recibidos.
 - A esto se le llama **ACK acumulativo**.
- Si se pierde un segmento llegan bien varios de los siguientes, para estos se generan **ACKs duplicados**.
- Para los números de secuencia, el receptor maneja variable *expectedSeqnum* que es el número de secuencia más chico que no llegó aun.

Retroceso-N en acción

Ventana window (N=4)

0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8

emisor

Envía pkt0
Envía pkt1
Envía pkt2
Envía pkt3
(espera)

rcv ack0, envía pkt4
rcv ack1, envía pkt5

ignore duplicate ACK



pkt 2 expira

Envía pkt2
Envía pkt3
Envía pkt4
Envía pkt5

receptor

Loss = pérdida

Recibe pkt0, envía ack0
Recibe pkt1, envía ack1

Recibe pkt3, descarta,
(re)envía ack1

Recibe pkt4, descarta,
(re)envía ack1

Recibe pkt5, descarta,
(re)envía ack1

rcv pkt2, entrega, send ack2
rcv pkt3, entrega, send ack3
rcv pkt4, entrega, send ack4
rcv pkt5, entrega, send ack5

Retroceso N

- ¿Si el espacio de secuencia es de $\text{MAX_SEQ} + 1$ números de secuencia (estos comienzan desde 0), se puede hacer la ventana emisora de tamaño $\text{MAX_SEQ} + 1$?
- La respuesta es no (Justificación en las 2 filmillas siguientes).
- **Conclusión:** El tamaño de la ventana emisora no puede superar MAX_SEQ cuando hay $\text{MAX_SEQ} + 1$ números de secuencia.

Retroceso N

- Considere la siguiente situación con $\text{MAX_SEQ} = 7$.
 1. El emisor envía paquetes 0 a 7.
 2. Llega al emisor una confirmación de recepción, superpuesta para paquete 7.
 3. El emisor envía otros 8 paquetes, con los números de secuencia 0 a 7.
 4. Ahora llega otra confirmación de recepción superpuesta para el paquete 7.
 5. ¡No se sabe si item 4. es un reenvío de ACK o uno nuevo!

Retroceso N

- ¿llegaron con éxito los 8 paquetes que correspondían al segundo bloque o se perdieron (contando como pérdidas los rechazos siguientes a un error)?
 - En ambos casos el receptor podría estar enviando el paquete 7 como confirmación de recepción.
 - El emisor no tiene manera de saberlo.
- **Por lo tanto:** el tamaño de la ventana emisora no puede superar MAX_SEQ cuando hay $\text{MAX_SEQ} + 1$ números de secuencia.

Retroceso N

- **Problema:** ¿Cómo evitar que haya más de *MAX_SEQ* paquetes sin ack pendientes?
- **Solución:** prohibir a la CR que moleste con más trabajo.
- **Implementación:** Usar *enable_network_layer* y *disable_network_layer*.

Retroceso N

- ¿Cuál es el problema principal de retroceso N?
- El uso ineficiente del canal frente a segmentos perdidos o demorados.

Protocolos de transferencia de datos confiable

- **Protocolos de tubería**
 - Protocolo Retroceso-N
 - **Protocolo de Repetición Selectiva**

Retroceso N

■ Situación:

- La latencia es grande.
- La proporción de errores o pérdida de paquetes es importante.
- Los paquetes se pueden demorar.

■ Solución: va a ser necesario hacer que el código del receptor maneje eficientemente los problemas de la red,

- por más que esto signifique complicar el código del receptor.
- Veremos el protocolo de repetición selectiva adopta este enfoque.

Repetición Selectiva

- ¿Qué ocurre si un paquete T a la mitad de una serie larga se pierde?
- La CT receptora debe entregar paquetes a la capa de aplicación en secuencia.
 - Por lo que no se pueden entregar a la capa de aplicación los paquetes que llegaron **bien** después de T .
- **Problema:** ¿qué debe hacerse con los paquetes correctos que le siguen a un paquete que se perdió?

Repetición Selectiva

- **Solución (Repetición Selectiva):**
 - Los paquetes en buen estado recibidos después de un **paquete dañado E** se **almacenan en búfer**.
 - Cuando el paquete E llega correctamente, el receptor entrega a la capa de aplicación, en secuencia, todos los paquetes posibles que ha almacenado en el búfer.

Repetición Selectiva

- **Mecanismo común de retransmisiones:**
 - El temporizador de E termina y el emisor lo manda de nuevo.
- **Una solución mejor:**
 - Uso de una ack negativa (NAK) por el receptor.
 - Así se estimula la retransmisión de paquetes antes que los temporizadores terminen y así se mejora el rendimiento.

Repetición Selectiva

- El receptor confirma individualmente todos los paquetes recibidos correctamente.
 - Hay búferes para paquetes según se necesiten para su entrega eventual en orden a la capa de aplicación.
- El emisor solo reenvía paquetes para los cuales el ACK no fue recibido o se recibió un NAK.
 - Hay un temporizador del emisor para cada paquete no confirmado.

Repetición Selectiva

- **Ventana del emisor**
 - Contiene N N° de secuencias consecutivos
 - Limita N° de secuencias a enviar a paquetes no confirmados.
- **Tipos de paquetes que puede haber en la ventana del emisor:**
 - Paquetes enviados y confirmados porque antes hay paquetes no confirmados
 - Paquetes enviados y no confirmados
 - Paquetes listos para enviarse en búfer

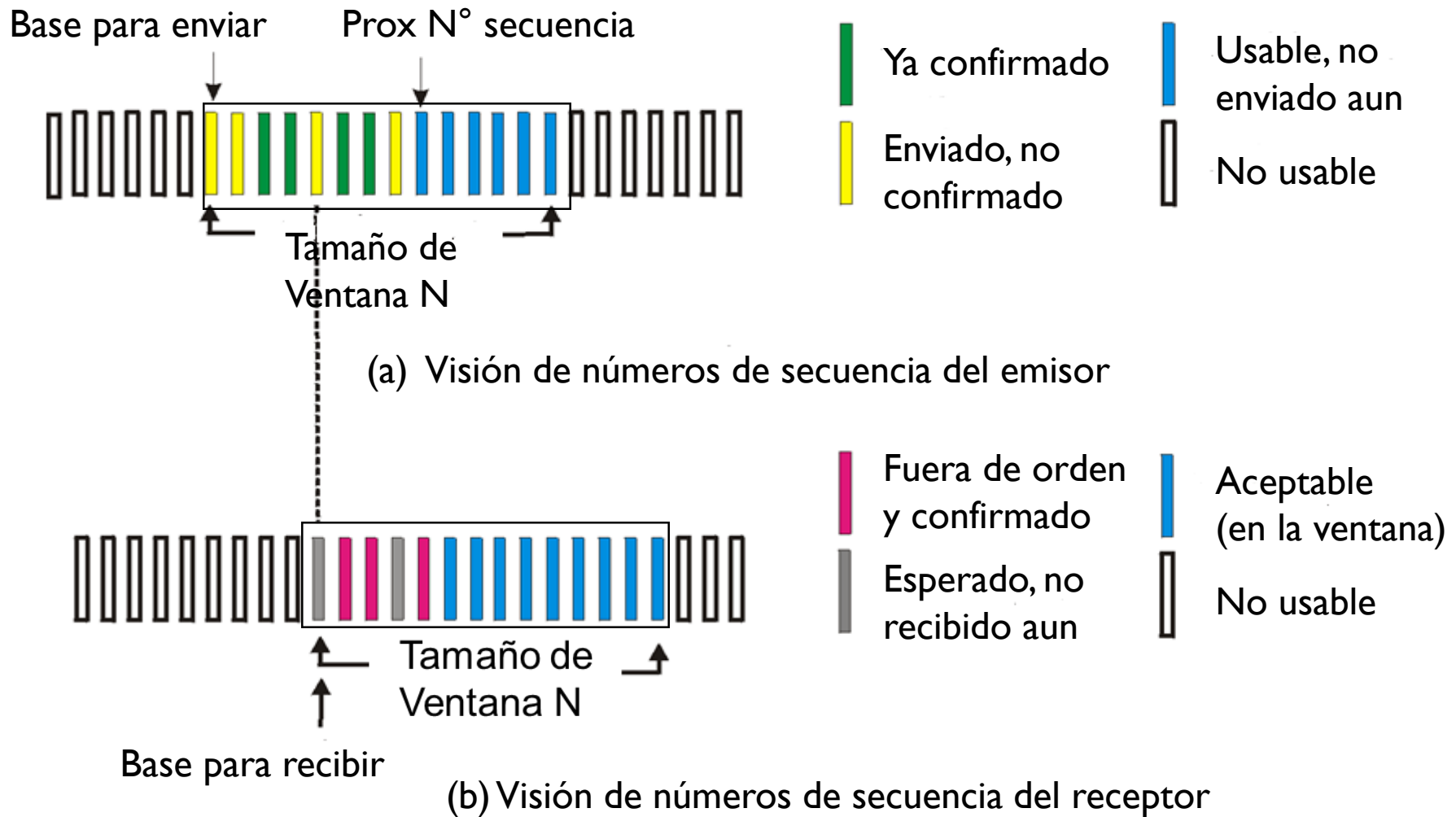
Repetición Selectiva

- **Situación:** Es necesario almacenar en búfer paquetes
 - porque puede perderse un paquete y llegar otros a continuación del mismo y en repetición selectiva estos se almacenan.
- **Problema:** ¿Cómo representar el conjunto de paquetes que puede almacenar en búfer el receptor?
- **Solución:** Usar *intervalos de números de secuencia* dentro del *espacio de números de secuencia*.
 - Un intervalo de esos recibe el nombre de **ventana corrediza**.

Repetición Selectiva

- **Tipos de paquetes que puede haber en la ventana del receptor:**
 - Paquetes esperados y no recibidos
 - Paquetes recibidos fuera de orden
 - Paquetes aceptables en la ventana que no han llegado aun
- **Se mantiene en búfer un paquete aceptado por la ventana receptora**
 - hasta que todos los que le preceden hayan sido pasados a la capa de aplicación.

Repetición Selectiva: ventanas del emisor y del receptor



Repetición Selectiva

- **Algunos detalles de repetición selectiva:**
 - tamaño de ventana emisora comienza en 0 y crece hasta MAX_SEQ.
 - El receptor tiene un búfer para cada N° de secuencia en su ventana.
 - **¿Qué se hace cuando llega un paquete?**
 - Cuando llega un paquete, su número de secuencia es revisado para ver si cae dentro de la ventana.
 - De ser así, y no ha sido recibido aun, se acepta y almacena.

Repetición selectiva

Emisor

Datos vienen de arriba:

- Si el próximo N° secuencia a enviar de la ventana está disponible, almacenar y enviar paquete

timeout(n):

- Reenviar paquete n , reiniciar timer

ACK(n) en [sendbase, sendbase+N]:

- marcar paquete n como recibido
- Si n es paquete más pequeño no confirmado, **avanzar base de ventana** al siguiente N° secuencia no confirmado.

Receptor

pkt n en [base rcv, base rcv +N-1]

- Enviar ACK(n)
- Fuera de orden: almacenarlo
- En orden: entregar (también entregar paquetes en bufer en orden), avanzar ventana al siguiente paquete que no ha sido recibido aun.

pkt n en [base rcv-N, base rcv-1]

- Enviar ACK(n)

Sino:

- ignorar

Súposiciones: N es tamaño de ventana del receptor. Algoritmo sin NAKs

Repetición selectiva en acción

ventana emisor (N=4)

0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8

emisor

Envía pkt0

Envía pkt1

Envía pkt2

Envía pkt3

(espera)

rcv ack0, envía pkt4

rcv ack1, envía pkt5

Registra que ack3 llegó



pkt 2 timeout

Envía pkt2

Registra que ack4 llegó

Registra que ack5 llegó

receptor

Recibe pkt0, envía ack0

Recibe pkt1, envía ack1

Recibe pkt3, almacena,
envía ack3

Recibe pkt4, almacena,
envía ack4

Recibe pkt5, almacena,
envía ack5

rcv pkt2; entrega pkt2,
pkt3, pkt4, pkt5; envía ack2

Q: ¿Qué pasa cuando llega ack2?

Dilema de repetición selectiva

Ejemplo:

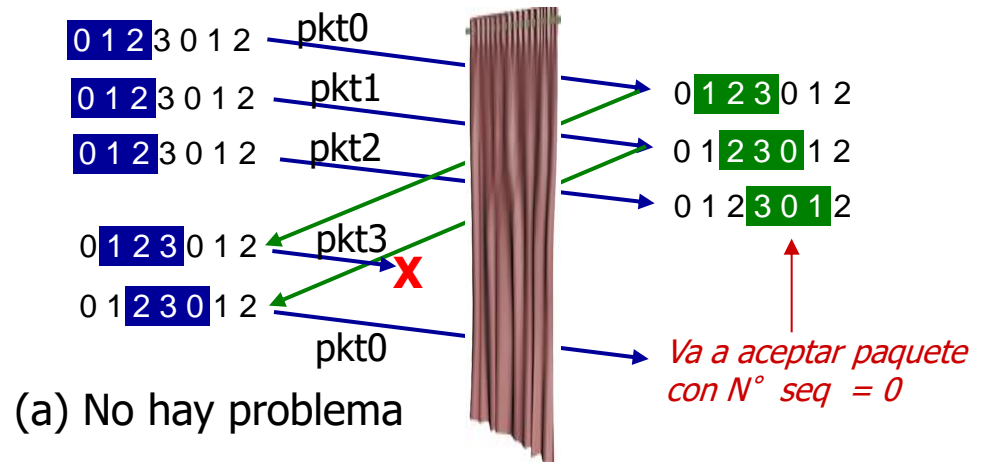
- Espacio de secuencia: 0, 1, 2, 3.
- Tamaño ventana = 3 en emisor y receptor

- El receptor no ve la diferencia en 2 escenarios.
- Datos duplicados aceptados como nuevos en (b)

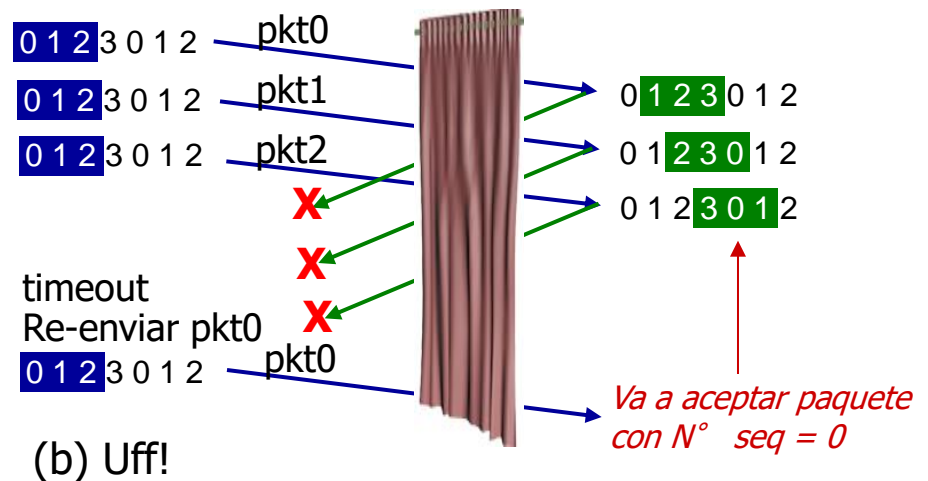
Q: ¿Qué relación entre tamaño de N° secs y tamaño de ventana en receptor debe haber para evitar el problema en (b)?

Ventana emisor
(después de recibir)

Ventana receptor
(después de recibir)



*El receptor no puede ver el lado del emisor.
¡El comportamiento del receptor es idéntico en ambos casos !
¡Algo está muy mal!*



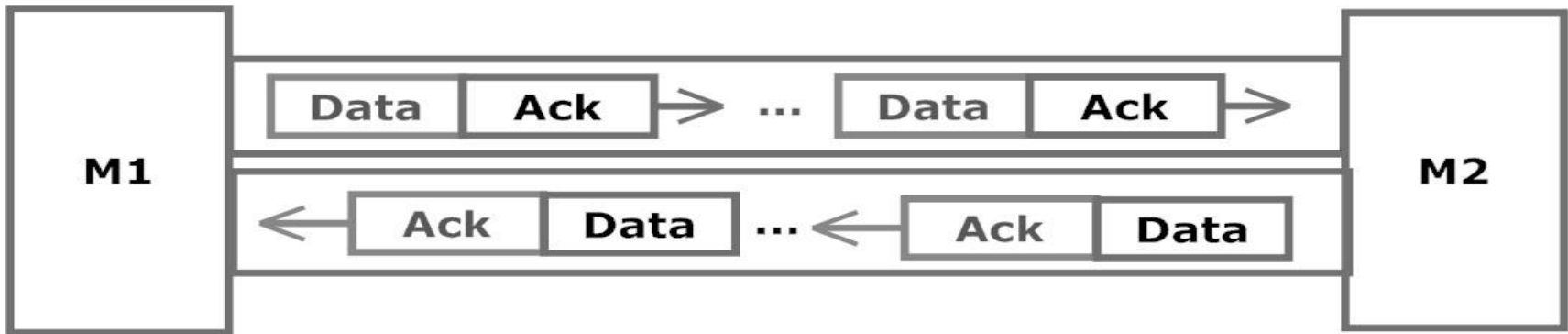
Repetición Selectiva

- **Regla para el tamaño de la ventana receptora:**
 - Tamaño de ventana receptora = $(MAX_SEQ + 1)/2$.
 - Con tamaños mayores de ventana receptora no funciona.

Repetición Selectiva

- En encabezado de paquete hay N° de secuencia de k bits.
- **Problema: ¿Cómo transmitir datos entre dos máquinas y en ambas direcciones eficientemente?**

Repetición Selectiva



- **Solución: llevar a caballito (piggybacking).**
 - cuando llega un segmento S con datos, el receptor se aguanta y espera hasta que la capa de aplicación le pasa el siguiente paquete P .
 - La confirmación de recepción de S se anexa a P en un segmento de salida (usando el **campo ack** en el encabezado del segmento de salida).

Repetición Selectiva

- **Problema:** ¿Cómo extender repetición selectiva para tener flujos de datos entre 2 máquinas en las dos direcciones?
- **Solución:**
 - Se usa llevar a caballito.
 - La capa de transporte para mandar un ack, debe esperar por un paquete al cual superponer un ack.

Repetición Selectiva

- Problema: ¿Cómo evitar retrasar demasiado envío de confirmaciones de recepción por no tener tráfico de regreso?
- Solución: método que usa temporizador auxiliar
 - tras llegar un paquete de datos en secuencia, se arranca un temporizador auxiliar mediante *start_ack_timer*.
 - Si no se ha presentado tráfico de regreso antes de que termine este temporizador, se envía un paquete de ack independiente.

Repetición Selectiva

- tiempo de temporizador auxiliar \ll tiempo de temporizador de retransmisiones.
 - \ll significa mucho menor.
- ¿Por qué?
 - para asegurarse que la ack de un paquete correctamente recibido llegue antes que el emisor termine su temporización y retransmita el paquete.