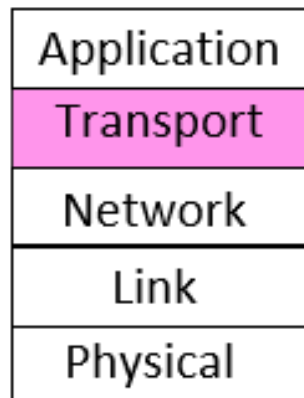


Capítulo 3

Capa de transporte

Evitando segmentos duplicados retrasados



Comparación de segmentos

- **¿Pueden viajar segmentos duplicados de un host emisor a uno receptor?**
 - Sí, por ejemplo, cuando se pierde un ack y un segmento se retransmite.
 - También cuando por congestión un segmento se demora, expira su temporizador y el segmento se retransmite.
- **Exigencia:** No se pueden entregar segmentos duplicados a la capa de aplicación.
- **Consecuencia:** Por lo tanto es necesario saber si un segmento que llega a un host es duplicado o no.

Comparación de segmentos

- **Problema:** ¿Cómo hacer para saber eficientemente si dos segmentos son diferentes o no?
- **Solución inviable:** comparar los segmentos bit a bit.
 - Requiere almacenar todos los segmentos que llegaron previamente.
 - Es muy ineficiente.

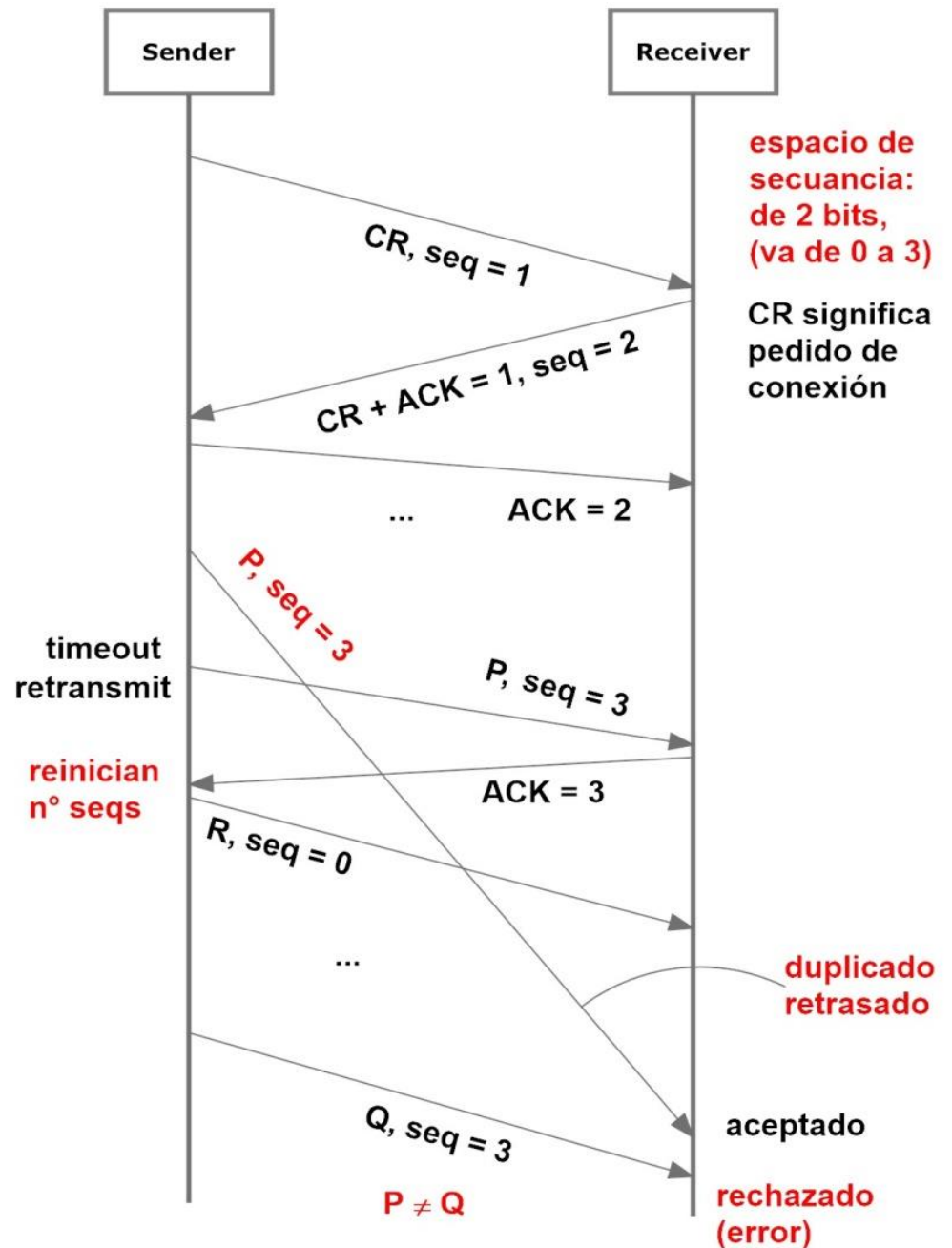
Comparación de segmentos

- **Solución (mejor):** Numerar los segmentos con **números de secuencia.**
 - Entonces paquetes con n° de secuencia diferentes son distintos.
 - Esta idea funcionaría bien si tenemos n° de secuencia de tamaño arbitrario.
 - O números de secuencia lo suficientemente largos como para estar seguros que no se van a reutilizar.

Comparación de segmentos

- Los números de secuencia no pueden ser de tamaño arbitrario
 - porque queremos que los segmentos tengan longitud máxima.
 - Por lo tanto el espacio de números de secuencia es finito;
 - porque queremos que el número de secuencia sea un campo del encabezado con longitud fija.
- Veremos que la idea de solo usar espacio de secuencia finito y numerar segmentos con n° de secuencia no funciona siempre bien.

Duplicado retrasado dentro de una conexión que genera problema

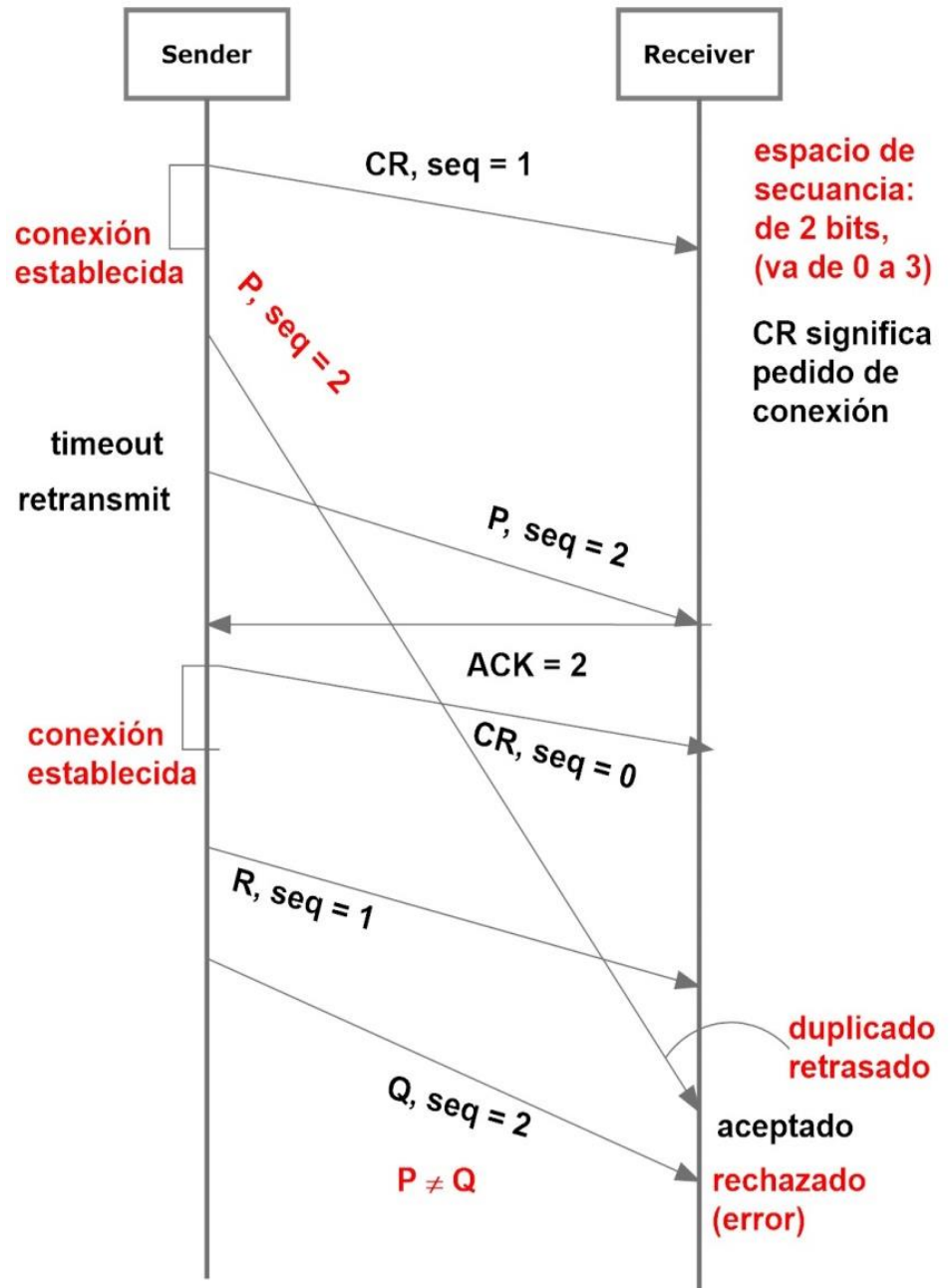


Problema de duplicados retrasados: dentro de una conexión

- **En el ejemplo anterior:**
 - Los números de secuencia alcanzan el máximo 3 vuelven a reiniciarse y aumentan hasta alcanzar el 3 de nuevo.
- **La capa de aplicación acepta un duplicado retrasado porque**
 - el espacio de secuencia se repite.
 - El duplicado retrasado permanece demasiado en la red.

Problema de duplicados retrasados: entre conexiones

Por ejemplo, antes de establecer la segunda conexión se libera la primera.



Problema de duplicados retrasados: entre conexiones

- En el ejemplo anterior:
- La segunda conexión arranca desde un n° de secuencia que permite llegar rápido al n° de secuencia 2.
- El duplicado retrasado con n° de secuencia 2 permanece demasiado tiempo en la red.

Duplicados retrasados

- ¿Cuál es el común denominador de los ejemplos anteriores?

Duplicados retrasados

- **Sucede la siguiente situación:** un segmento S con n° de secuencia x queda demorado debido a que la red está congestionada.
 - El temporizador de retransmisiones asociado a S expira y se retransmite S .
 - El protocolo de enrutamiento cambia las rutas y la retransmisión de S llega rápido a destino.
 - Pero aun quedó en la red un **duplicado retrasado** de S (de n° de secuencia x).
 - Ese duplicado retrasado de S más adelante llega a destino generando problemas.
 - Este tipo de problemas es tan serio que debe ser evitado.

¿Cómo encarar problemas de duplicados retrasados?

- **Idea:** *Asegurar que ningún paquete viva más allá de T sec. (**tiempo de vida de paquete**)*
 - Esto se refiere a paquetes de datos, retransmisiones de ellos y a confirmaciones de recepción.
 - Eliminar paquetes viejos que andan dando vueltas por ahí.
- Veremos que esta idea hace la solución de los problemas de duplicados retrasados manejables.

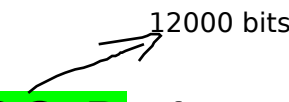
Problema de duplicados retrasados dentro de una conexión

- Para resolver el problema de duplicados retrasados dentro de una conexión:
 - Asumiendo que T es el tiempo de vida de paquete, el origen etiqueta los segmentos con n° de secuencia que no van a reutilizarse dentro de T sec.

Problema de duplicados retrasados dentro de una conexión

- Para lograr que al regresar al principio de los n° de secuencia, los segmentos viejos con el mismo n° de secuencia hayan desaparecido hace mucho tiempo:
- El espacio de secuencia debe ser lo suficientemente grande para garantizar esto.

Ejercicio

- Asumimos que el tiempo de vida máximo de segmento es de 30 sec. Se transmiten segmentos de tamaño máximo de 1500 B. La tasa de transferencia es de 10 Mbps.  12000 bits
10.000.000 bits
- ¿Cuán grande debe ser el espacio de secuencia? (i.e. ¿de cuántos bits debe ser?).

x Sec puedo mandar 10.000.000 bits

en 30 Sec puedo mandar 300.000.000 bits

833.33 paquetes por segundo

25000 paquetes en 30 sec

15 bits es decir 16 bits MAL

como 2^{14} es menor a 25mil y 2^{15} es mayor uso este ultimo

Ejercicio

- Necesitamos una cantidad de números de secuencia que no se puedan repetir en 30 segundos cuando mandamos continuamente segmentos de 1500B a 10Mbps.
- Necesitamos responder cuántos segmentos de 1500 B se pueden mandar continuamente durante 30 seg a 10Mbps.
- La **cantidad de números de secuencia** deberá ser mayor a esa cantidad de segmentos.
 - Tendrá que ser una potencia de 2 porque el numero de secuencia es un campo del encabezado del segmento.

Ejercicio

- ¿cuántos bits se pueden transferir en 30 sec?

Ejercicio

- Segmentos son de 1500 B = 12000 b.
- **¿Cuánto demora enviar un segmento?**
10.000.000 b ----- 1000 msec
12000 b ----- x
- X = 1,2 msec.
- **¿Cuántos segmentos se pueden mandar en 30 sec de manera continua?**

Ejercicio

1,2 msec ----- 1 segmento

30000 msec ----- y

- $Y = 25000 < 2^{15}$ (próxima potencia de 2)
- Basta tomar espacio de secuencia de 15 b.

¿Cómo evitar que duplicado retrasado que pasa de una conexión a otra genere problemas?

- Como al establecer una conexión se usan segmentos, una conexión debería tener un **N° inicial de secuencia** con el que comienza a operar.

¿Cómo evitar que duplicado retrasado que pasa de una conexión a otra genere problema?

- ***Idea de solución***: hay que escoger como número inicial de secuencia de la conexión nueva un n° de secuencia que haga imposible o improbable que el duplicado retrasado de n° de secuencia x genere problemas.
 - Además se mantiene dentro de una conexión que el origen etiqueta los segmentos con n° de secuencia que no van a reutilizarse dentro de T sec (tiempo de vida del paquete).

¿Cómo evitar que duplicado retrasado que pasa de una conexión a otra genere problema?

- **Implementación 1** (en libro de Comer): al crear una nueva conexión cada extremo genera un n° de secuencia de 32 bits aleatorio que pasa a ser el número inicial de secuencia para los datos enviados.
 - Alguna implementación de TCP usa esta solución.
- **¿Por qué tiende a funcionar?**
 - La probabilidad de que un paquete duplicado retrasado genere problemas en una conexión siguiente es baja debido a la elección aleatoria del número inicial de secuencia de la conexión siguiente.

¿Cómo evitar que duplicado retrasado que pasa de una conexión a otra genere problema?

- **Implementación 2** (en libro de Tanenbaum): vincular n° de secuencia de algún modo al tiempo y para medir el tiempo usar un reloj.
 - Cada host tiene un **reloj de hora del día**.
 - Los relojes de los hosts **no necesitan ser sincronizados**;
 - se supone que **cada reloj es un contador binario que se incrementa a si mismo en intervalos uniformes**.
 - **El reloj continua operando aun ante la caída del host**
- Cuando **se establece una conexión los k bits de orden mayor del reloj = número inicial de secuencia.**

¿Cómo evitar que duplicado retrasado que pasa de una conexión a otra genere problema?

- ¿Por qué funciona?
- Si el reloj se mueve más rápido que la asignación de números de secuencia a los paquetes que se envían (esta es una suposición razonable),
 - entonces el número inicial de secuencia de una nueva conexión va a ser mayor al n° de secuencia de cualquier duplicado retrasado de la conexión previa.
 - *Por eso me gusta más esta solución que la anterior.*

¿Cómo evitar que duplicado retrasado que pasa de una conexión a otra genere problema?

- Tanenbaum da un ejemplo de una implementación de TCP donde:
 - Se usa un **esquema basado en reloj** con un pulso de reloj cada 4 μ sec.
 - El tiempo de vida máximo de un paquete es de 120 sec.