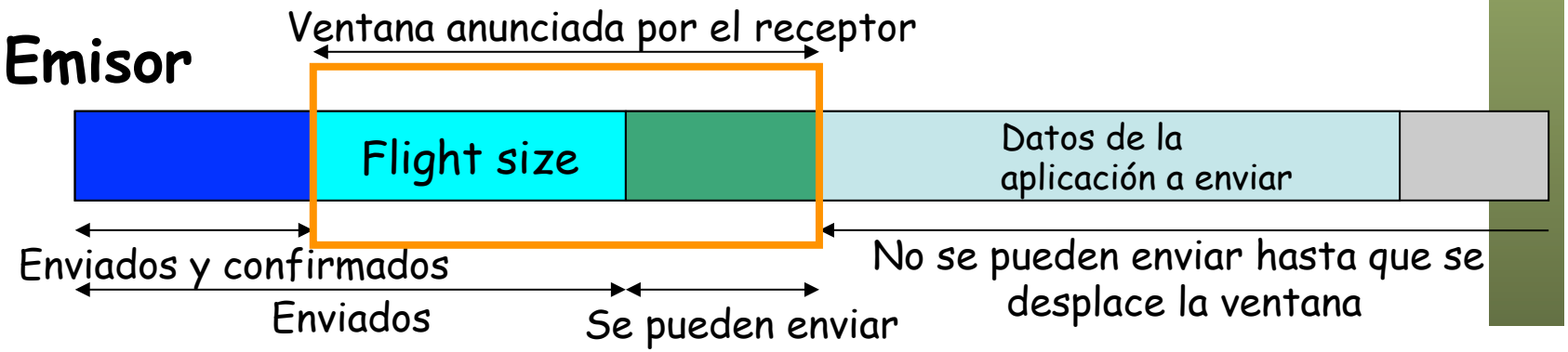
Control de congestión en TCP

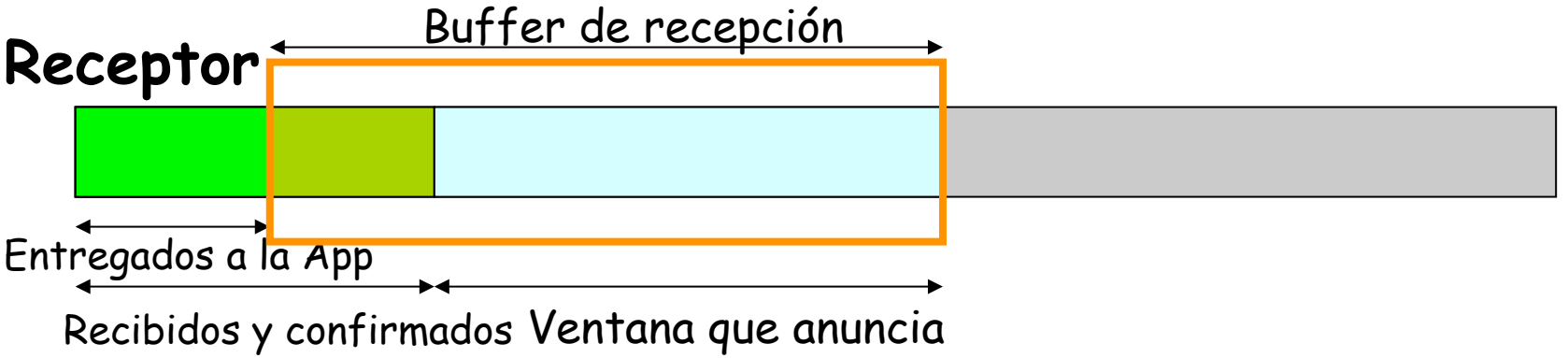
Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Grado en Ingeniería en Tecnologías de Telecomunicación, 4º

Ventana: Básico

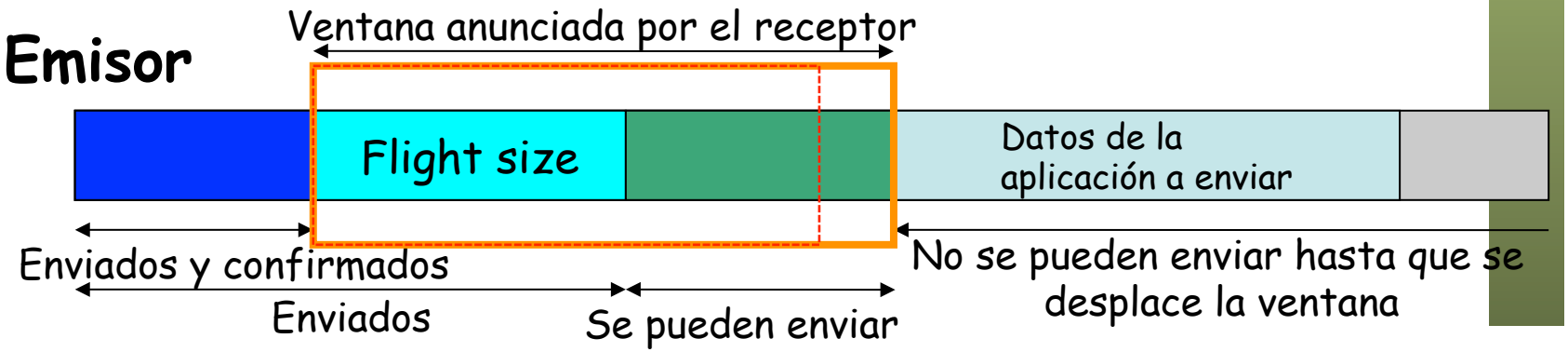
- Por simplicidad analicemos solo un sentido
- La aplicación **receptor lee bytes** del stream
  - La ventana se abre en el emisor (*window update*)
  - Se desliza en el receptor (…)

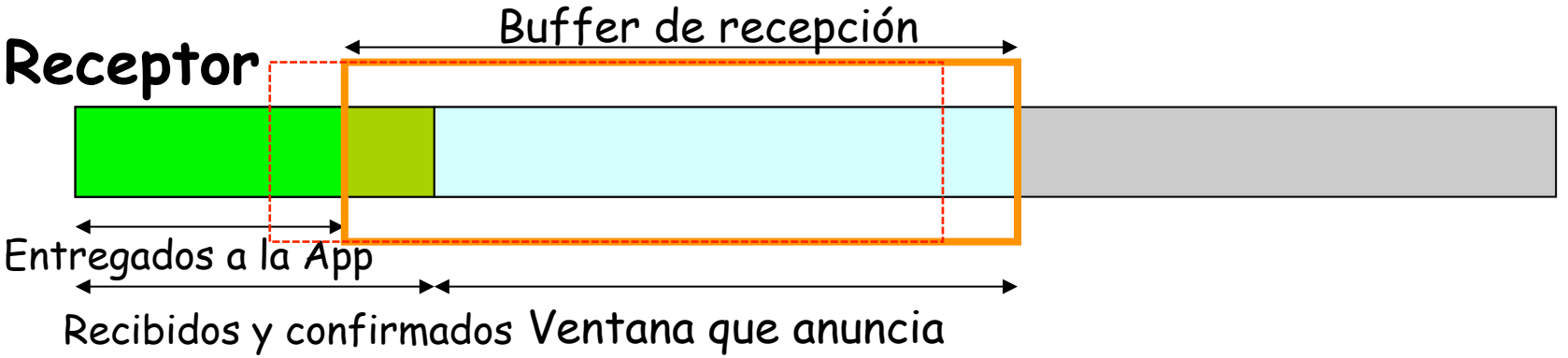




Ventana: Básico

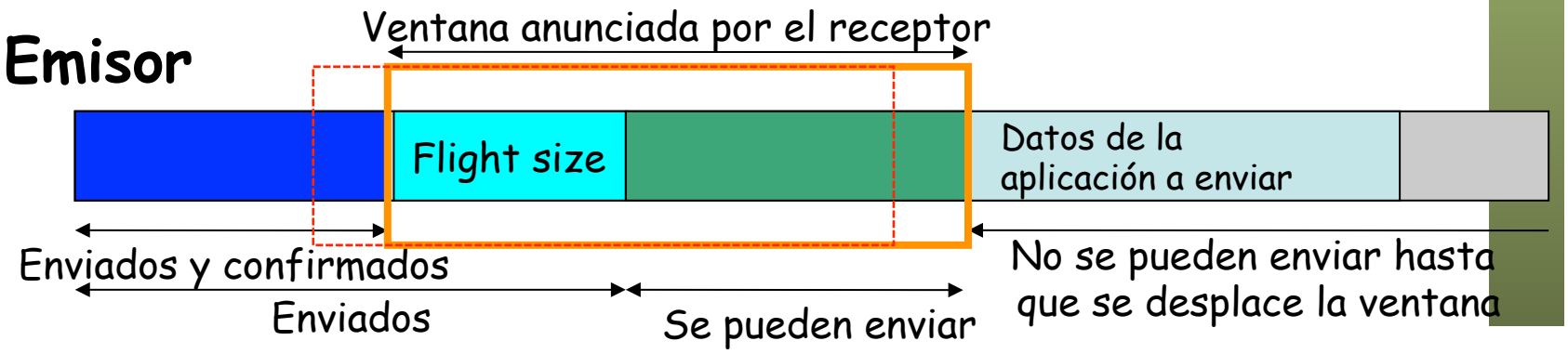
- Por simplicidad analicemos solo un sentido
- La aplicación **receptor lee bytes** del stream
  - La ventana se abre en el emisor (*window update*)
  - Se desliza en el receptor

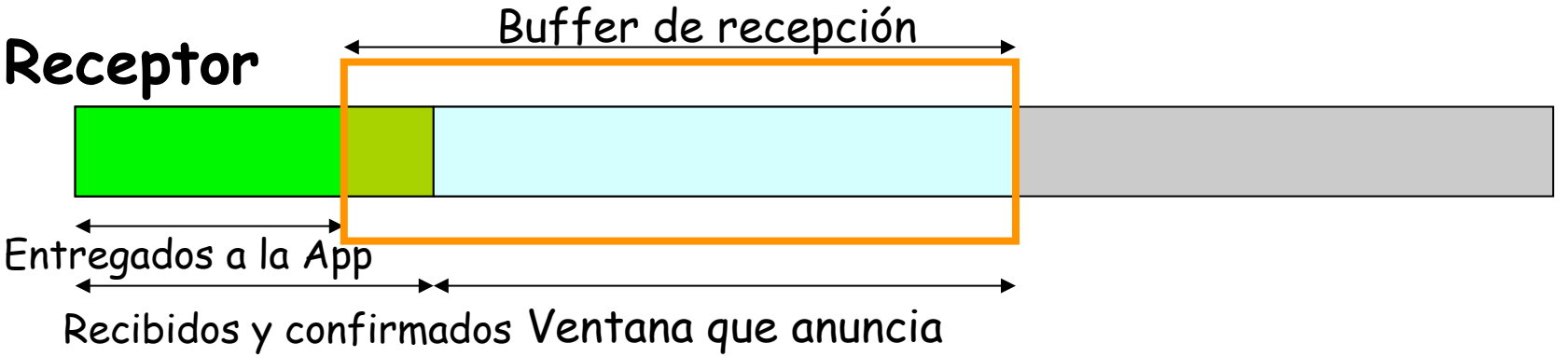




Ventana: Básico

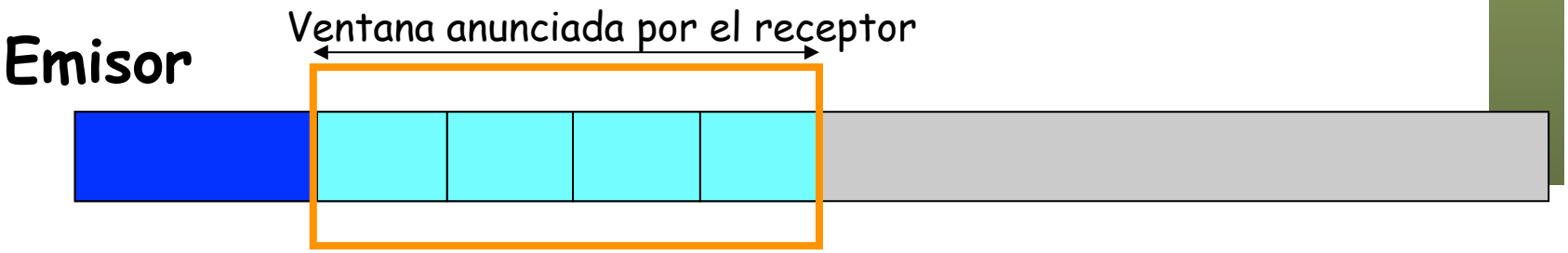
- Por simplicidad analicemos solo un sentido
- **Se reciben más confirmaciones**
- La ventana se desliza en el emisor

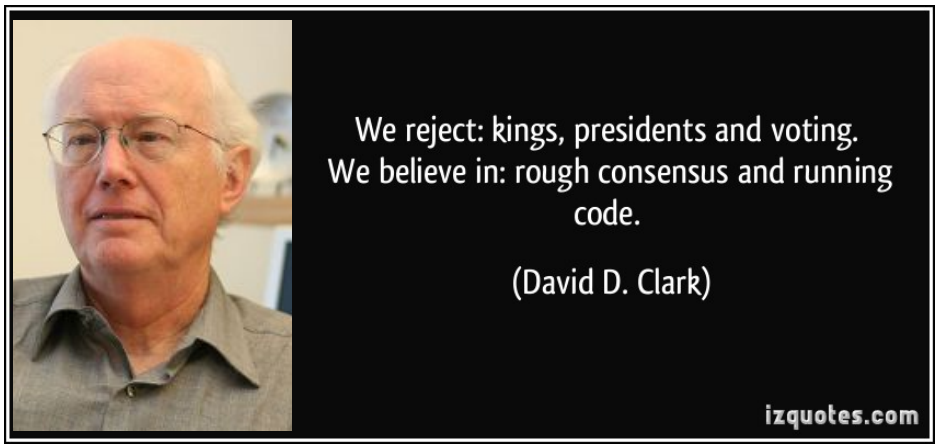




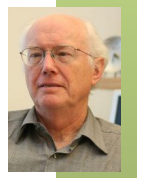
Ventana: SWS

- *Silly Window Syndrome*, David D. Clark, RFC 813
- Ejemplo:
  - Emisor envía la ventana en 4 segmentos
  - (...)

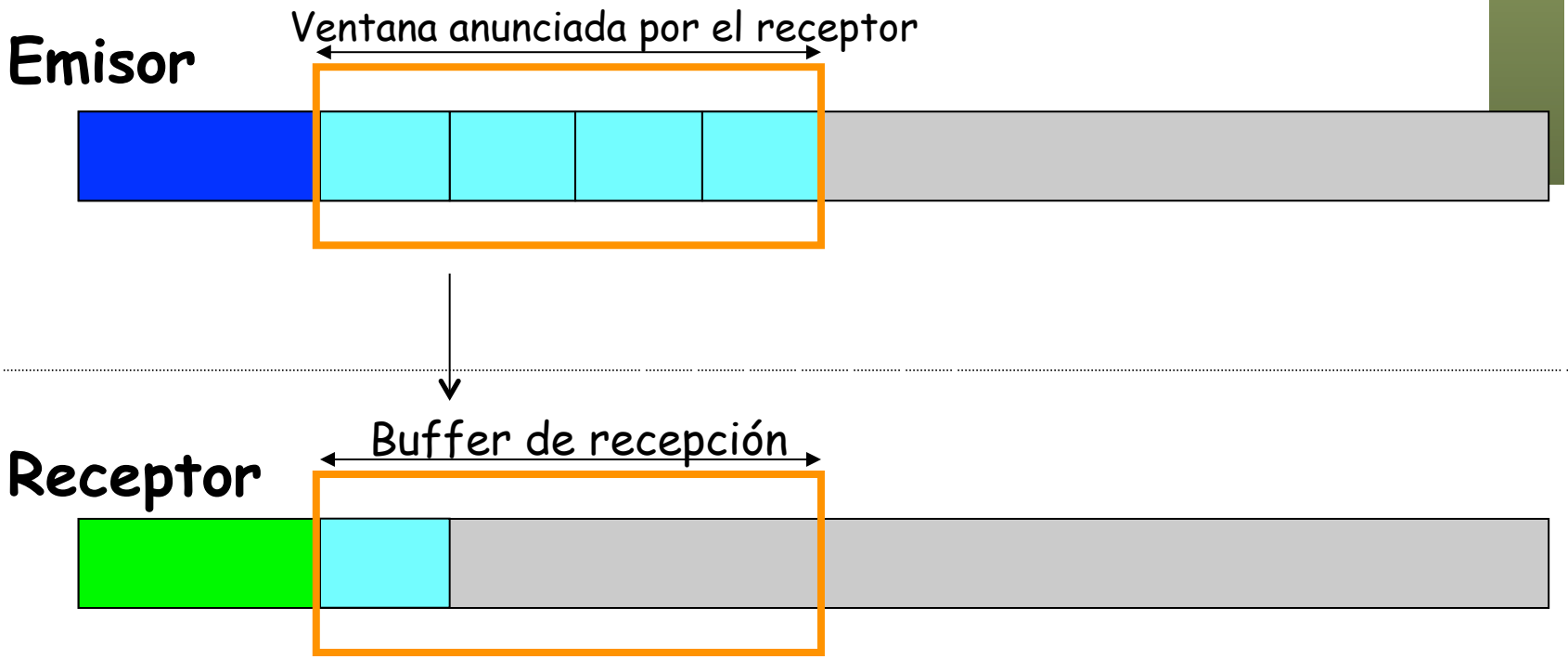




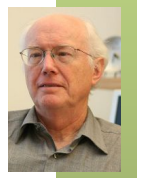
Ventana: SWS



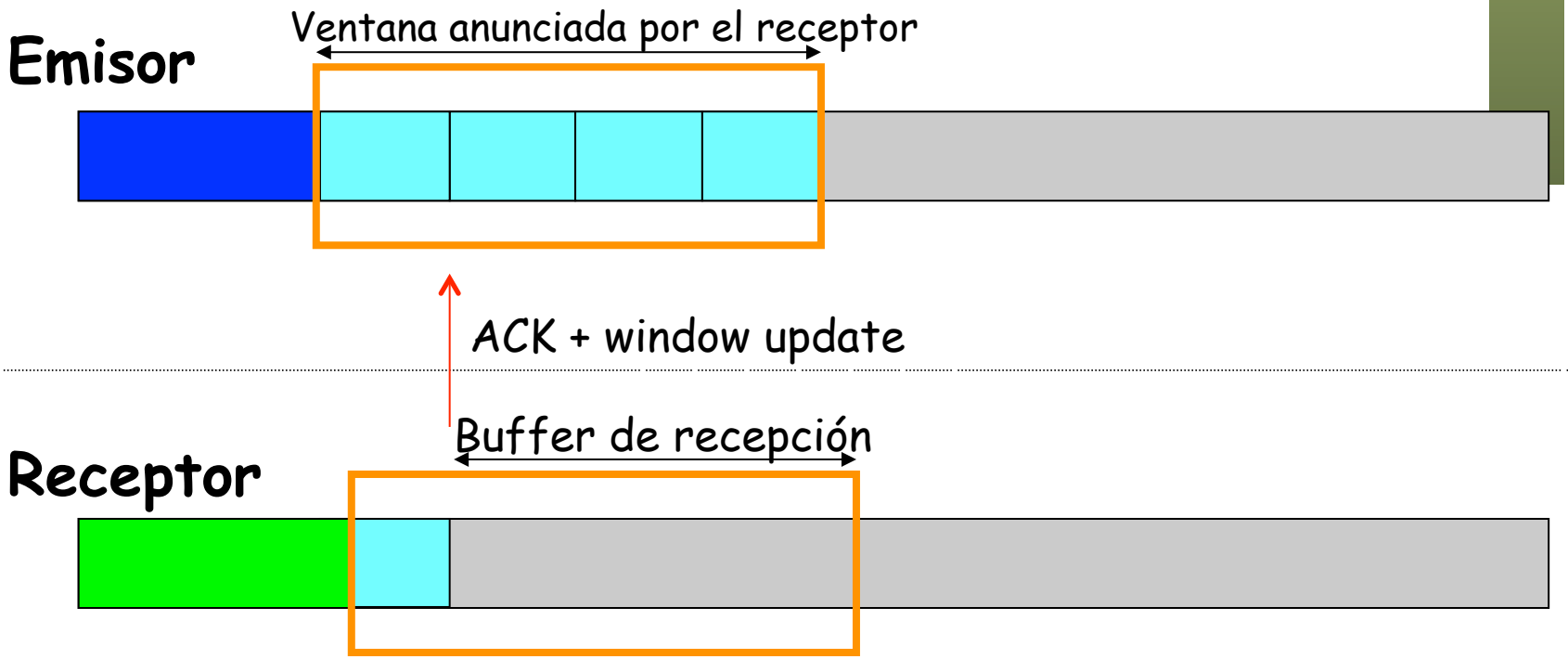
- *Silly Window Syndrome*, David D. Clark, RFC 813
- Ejemplo:
  - Emisor envía la ventana en 4 segmentos
  - Llega el primero (...)



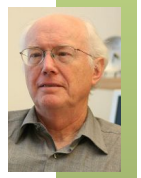
Ventana: SWS



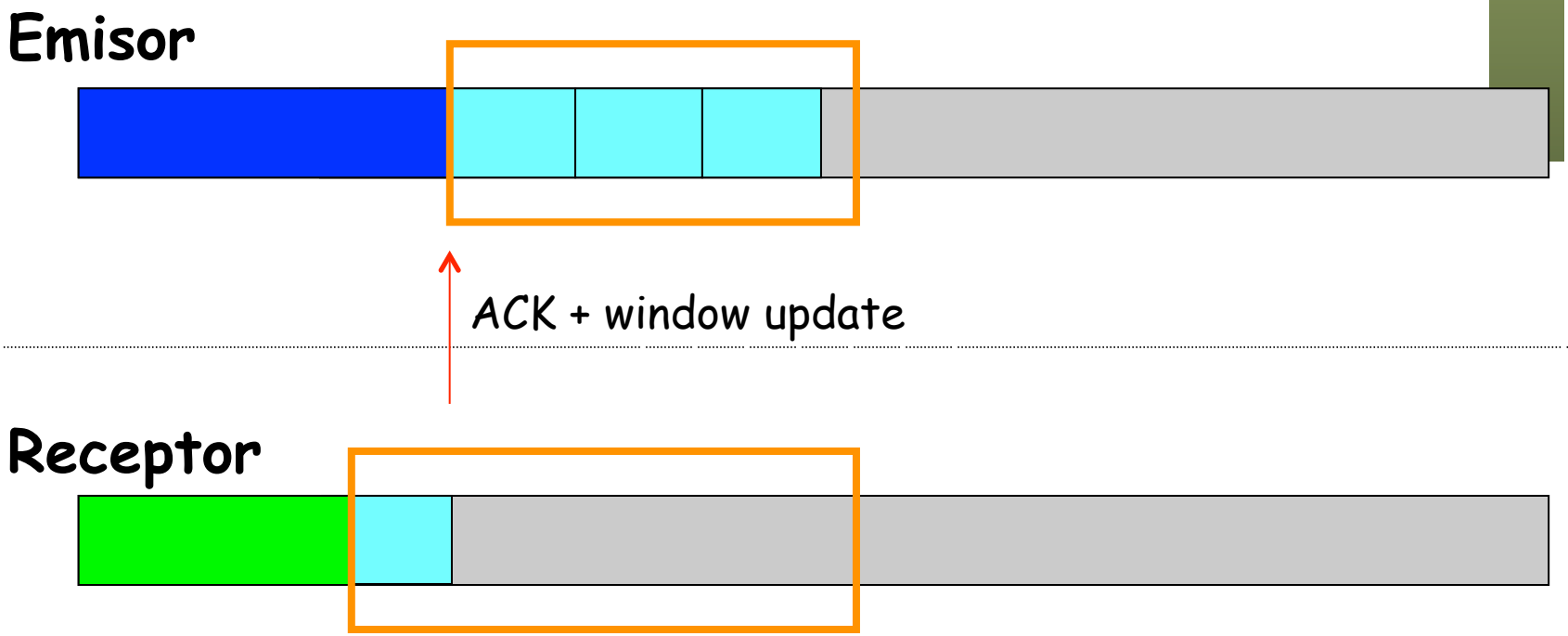
- *Silly Window Syndrome*, David D. Clark, RFC 813
- Ejemplo:
  - Emisor envía la ventana en 4 segmentos
  - Llega el primero y lo confirma, pero la aplicación solo lee parte (...)



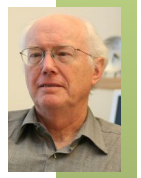
Ventana: SWS



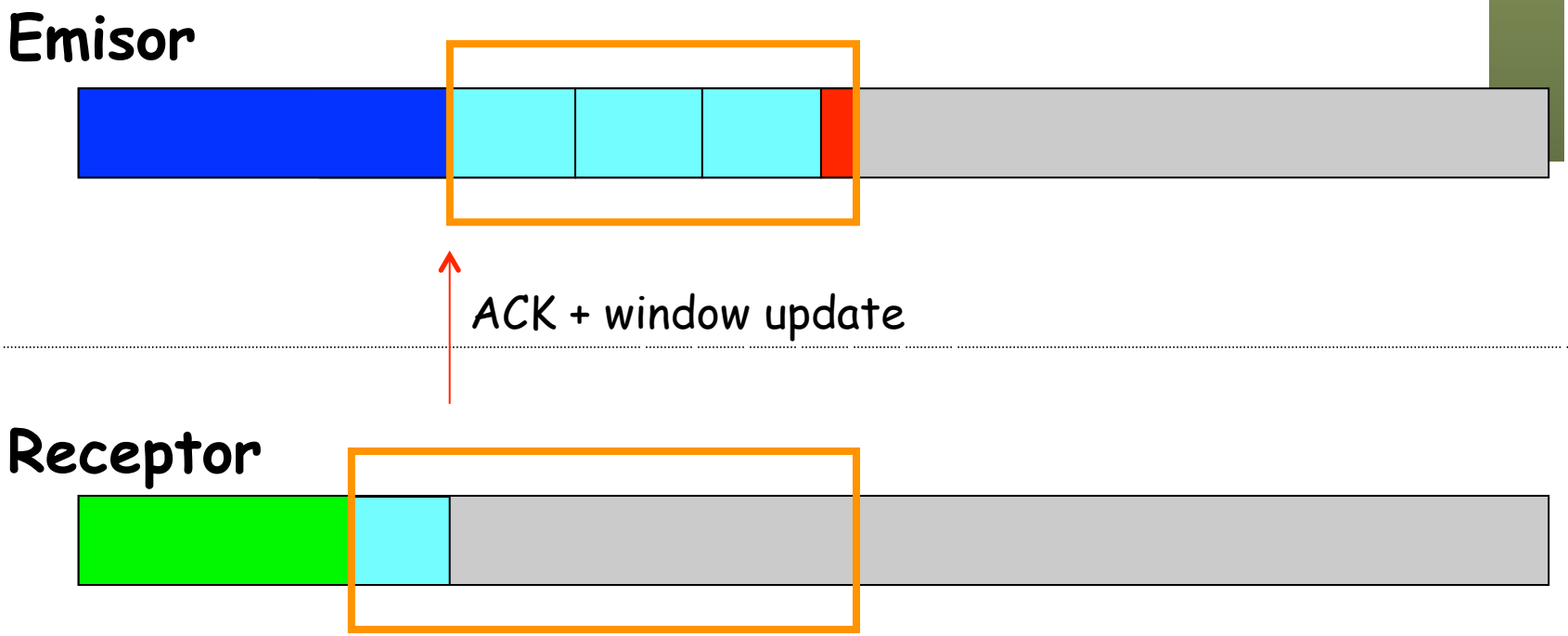
- *Silly Window Syndrome*, David D. Clark, RFC 813
- Ejemplo:
  - Emisor envía la ventana en 4 segmentos
  - Llega el primero y lo confirma, pero la aplicación solo lee parte
  - Se desplaza y cierra la ventana en el emisor (...)



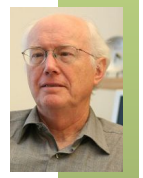
Ventana: SWS



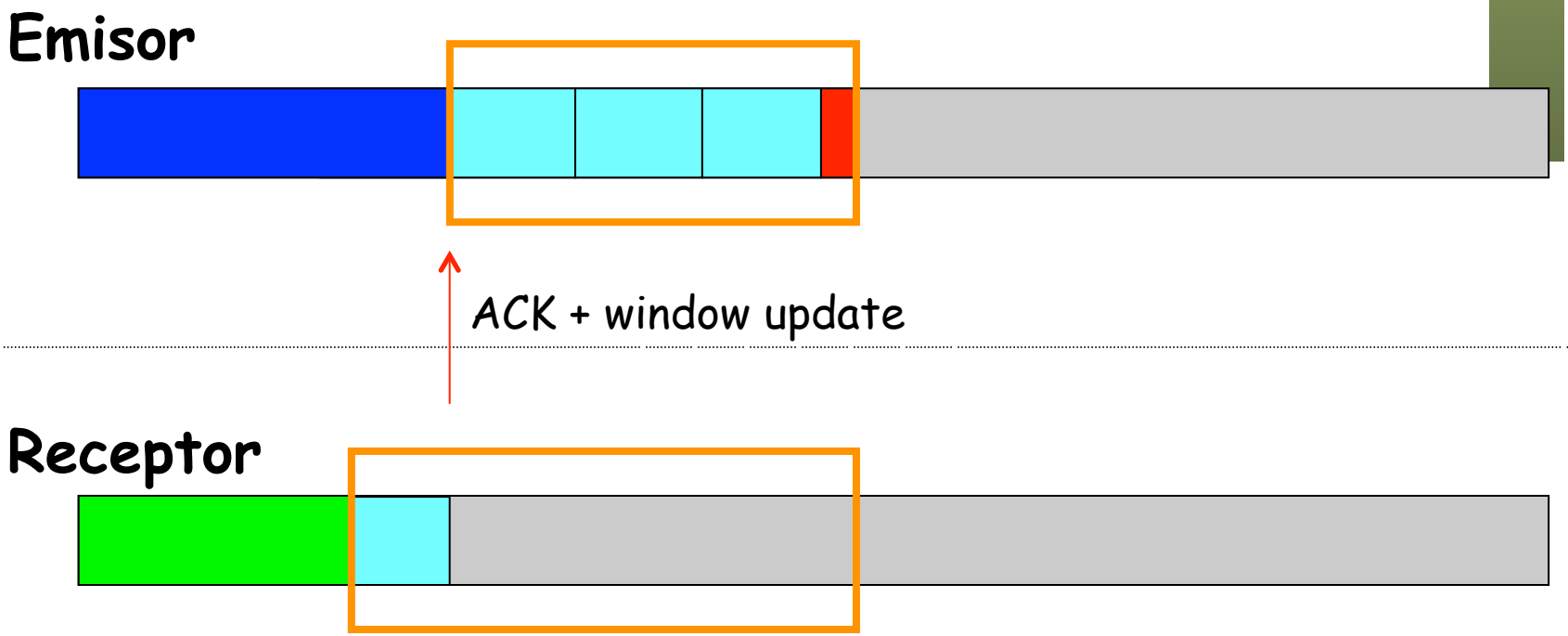
- *Silly Window Syndrome*, David D. Clark, RFC 813
- Ejemplo:
  - Emisor envía la ventana en 4 segmentos
  - Llega el primero y lo confirma, pero la aplicación solo lee parte
  - Se desplaza y cierra la ventana en el emisor
  - Ahora puede enviar un segmento nuevo, pero es pequeño



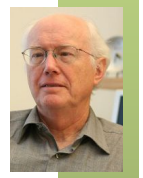
Ventana: SWS



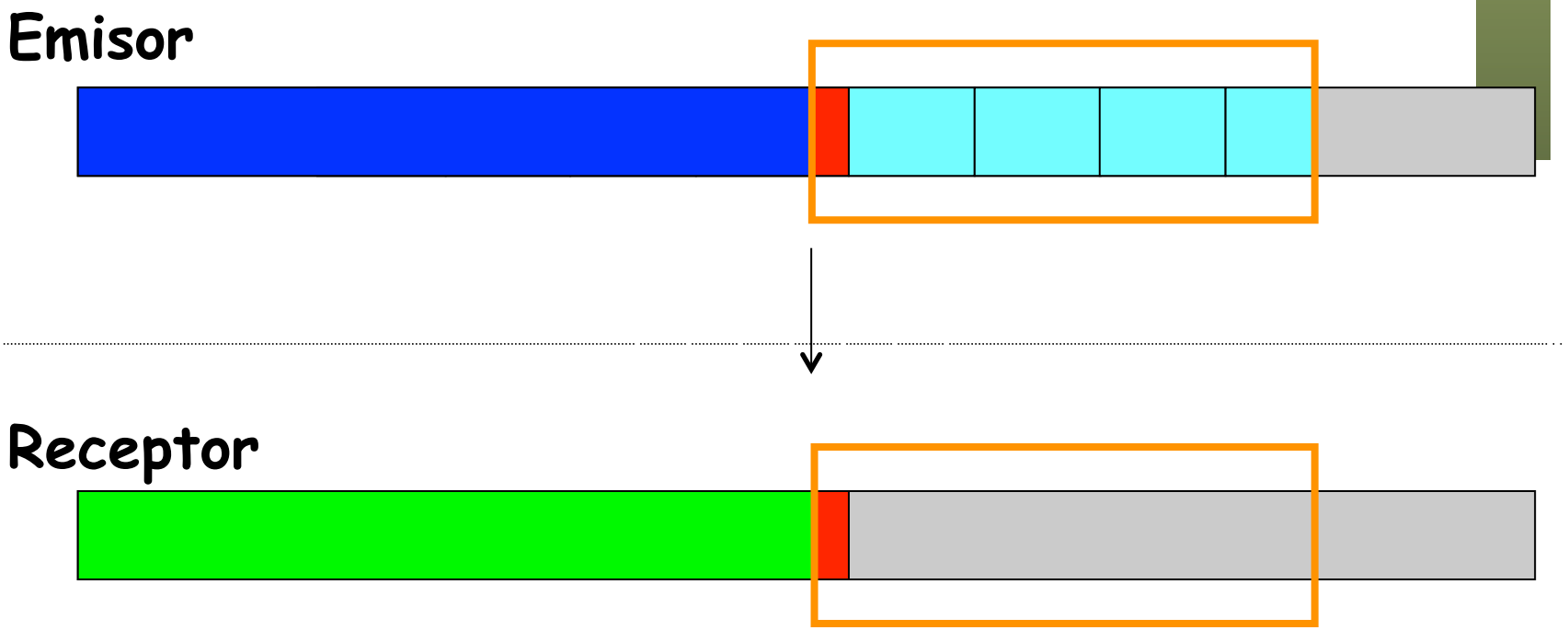
- Ejemplo:
  - Tiempo más tarde (...)



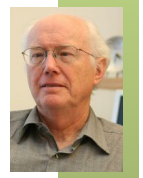
Ventana: SWS



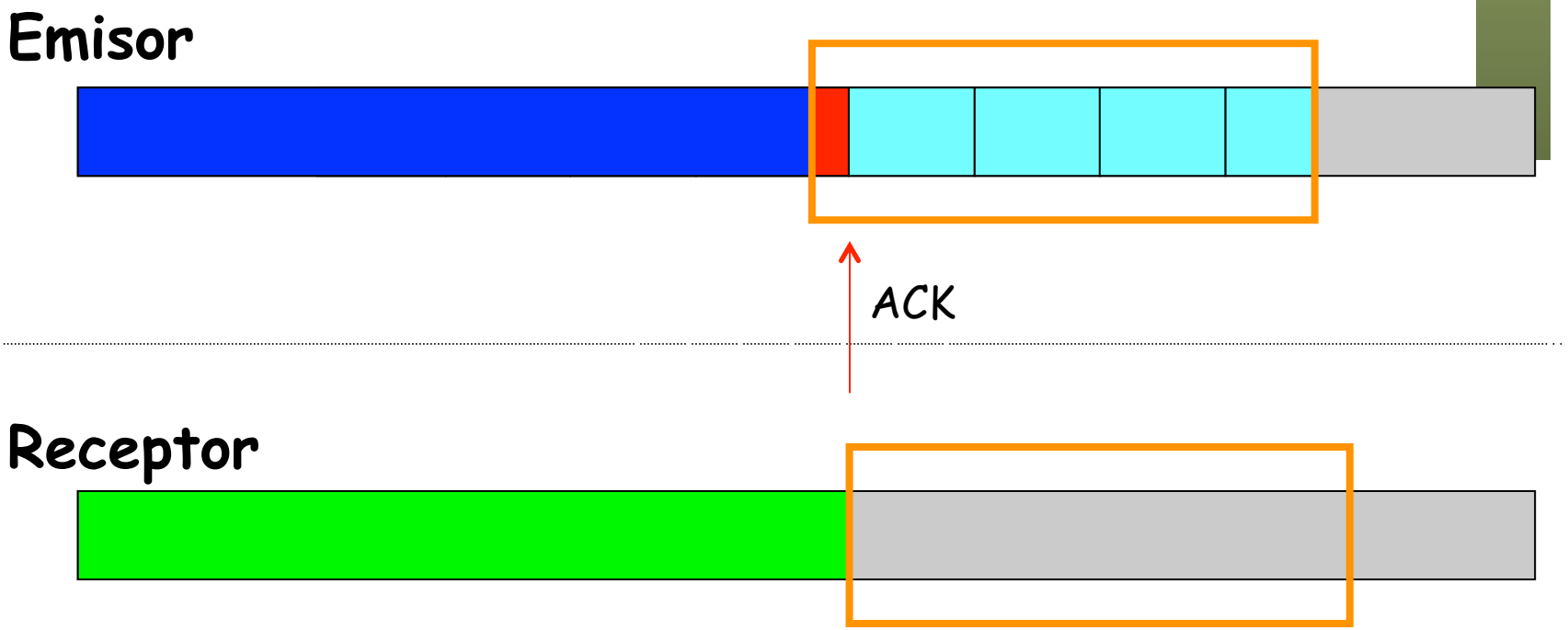
- Ejemplo:
  - Tiempo más tarde ese segmento pequeño llega (...)



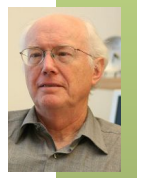
Ventana: SWS



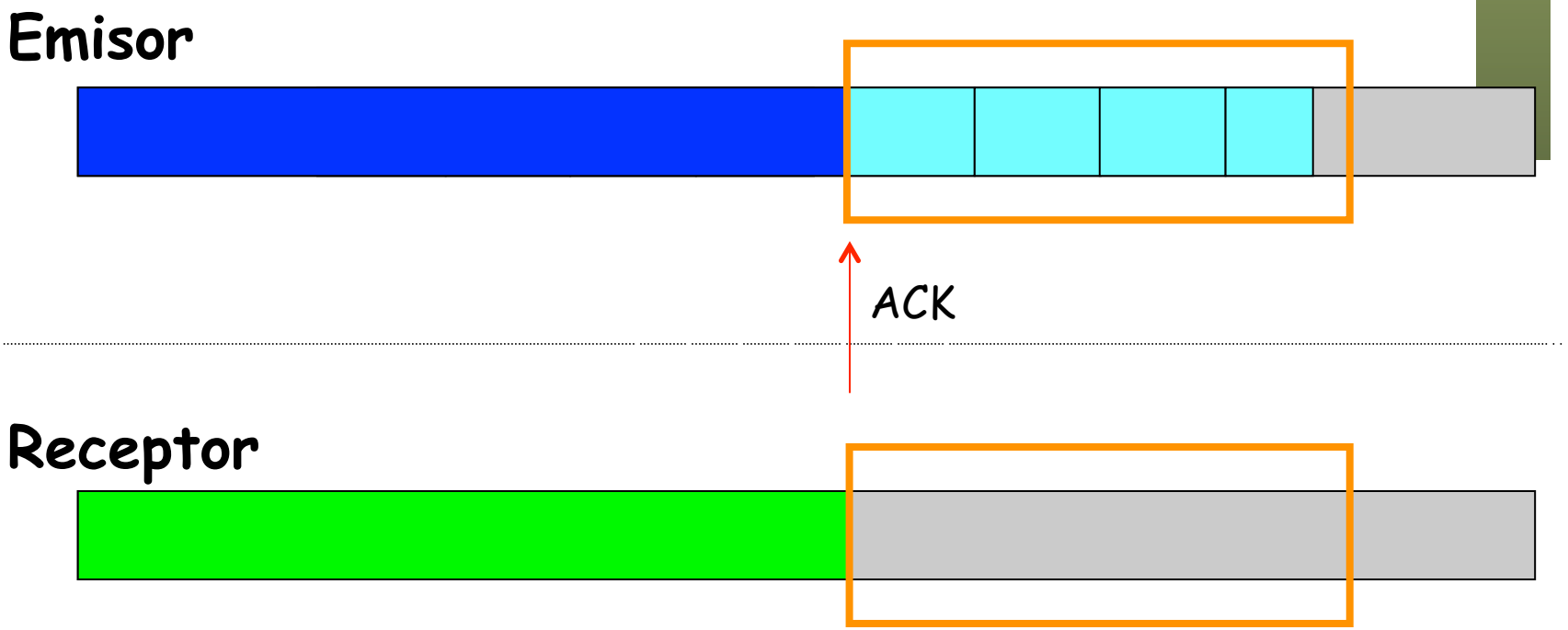
- Ejemplo:
  - Tiempo más tarde ese segmento pequeño llega y se confirma
  - (...)



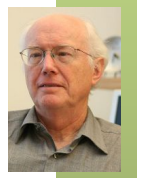
Ventana: SWS



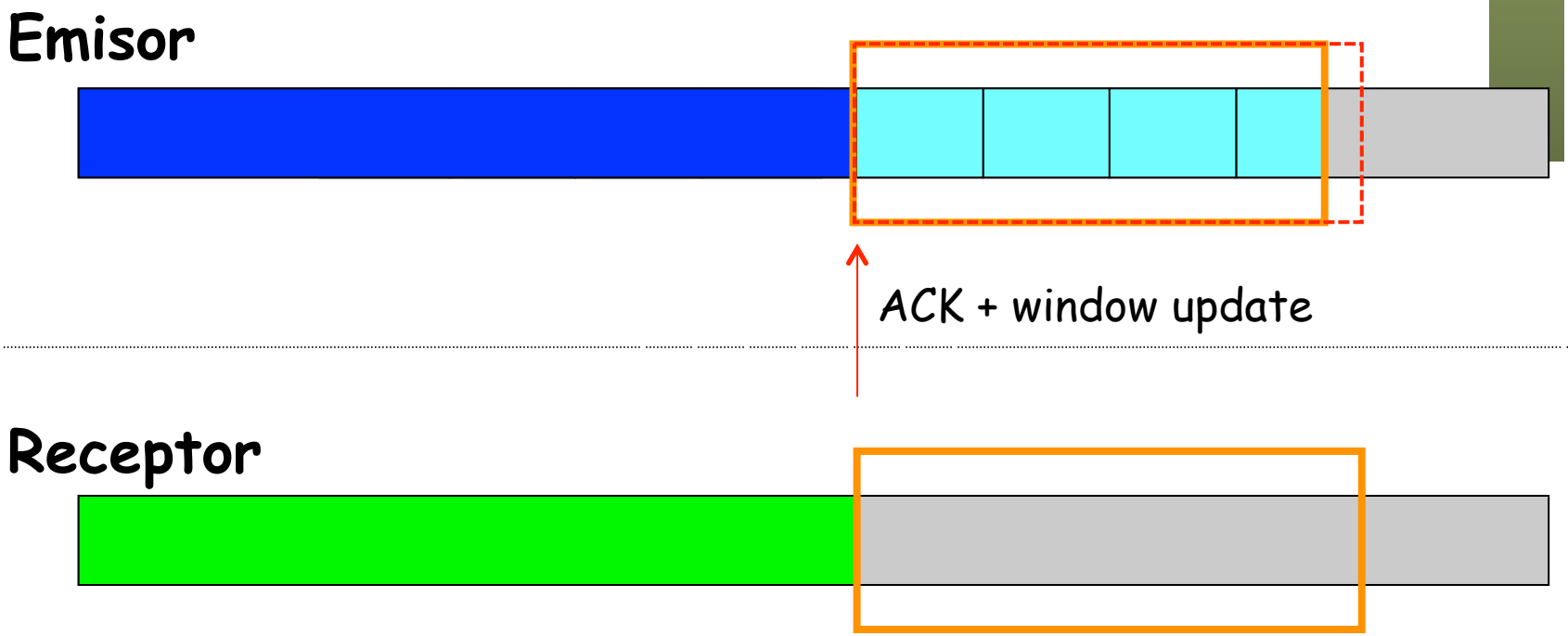
- Ejemplo:
  - Tiempo más tarde ese segmento pequeño llega y se confirma
  - Eso desplaza la ventana solo ese poco
  - Y ahora, si hay nuevos datos, solo se pueden enviar en un segmento pequeño
  - (...)



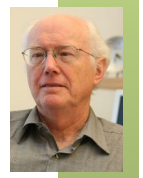
Ventana: SWS



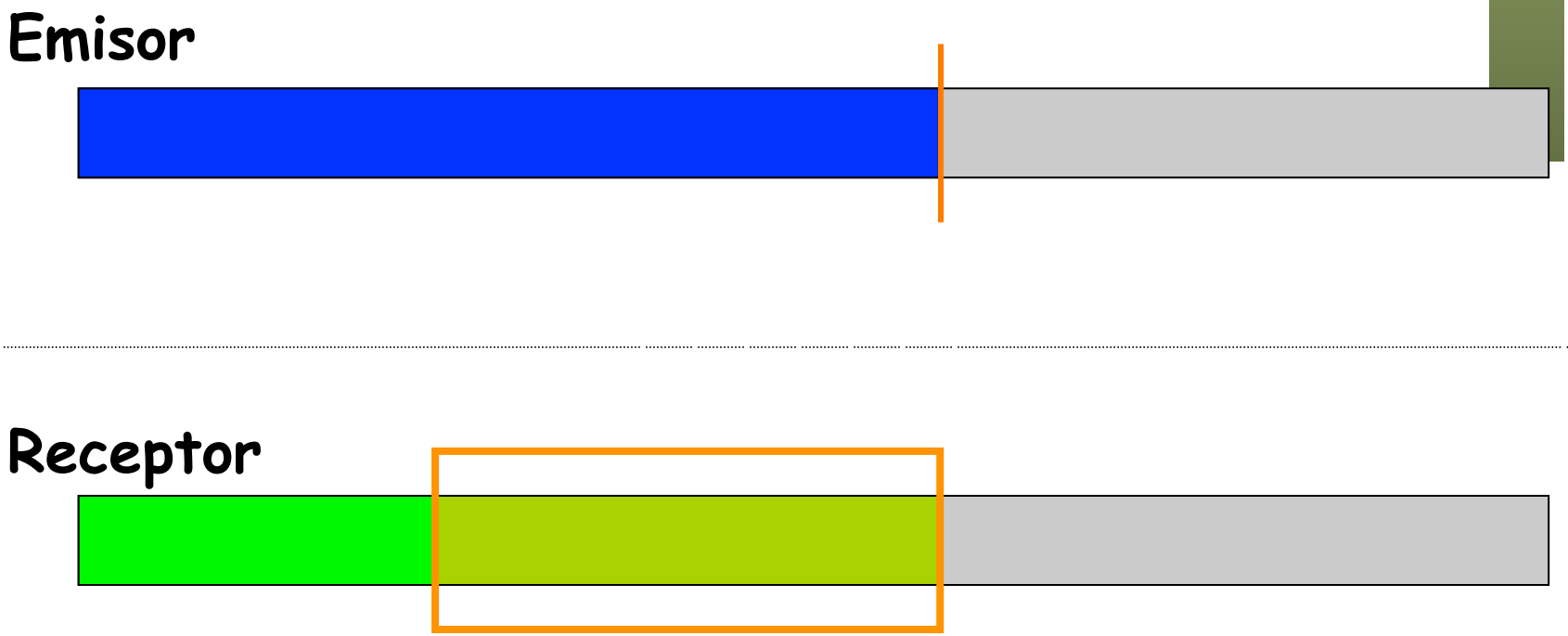
- Ejemplo:
  - Tiempo más tarde ese segmento pequeño llega y se confirma
  - Eso desplaza la ventana solo ese poco
  - Y ahora, si hay nuevos datos, solo se pueden enviar en un segmento pequeño
  - Para evitar esto, el anuncio de ventana en la confirmación podría reducirla para que no deje hueco hasta que sea al menos del MSS



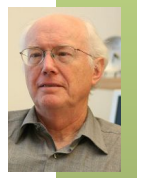
Ventana: SWS



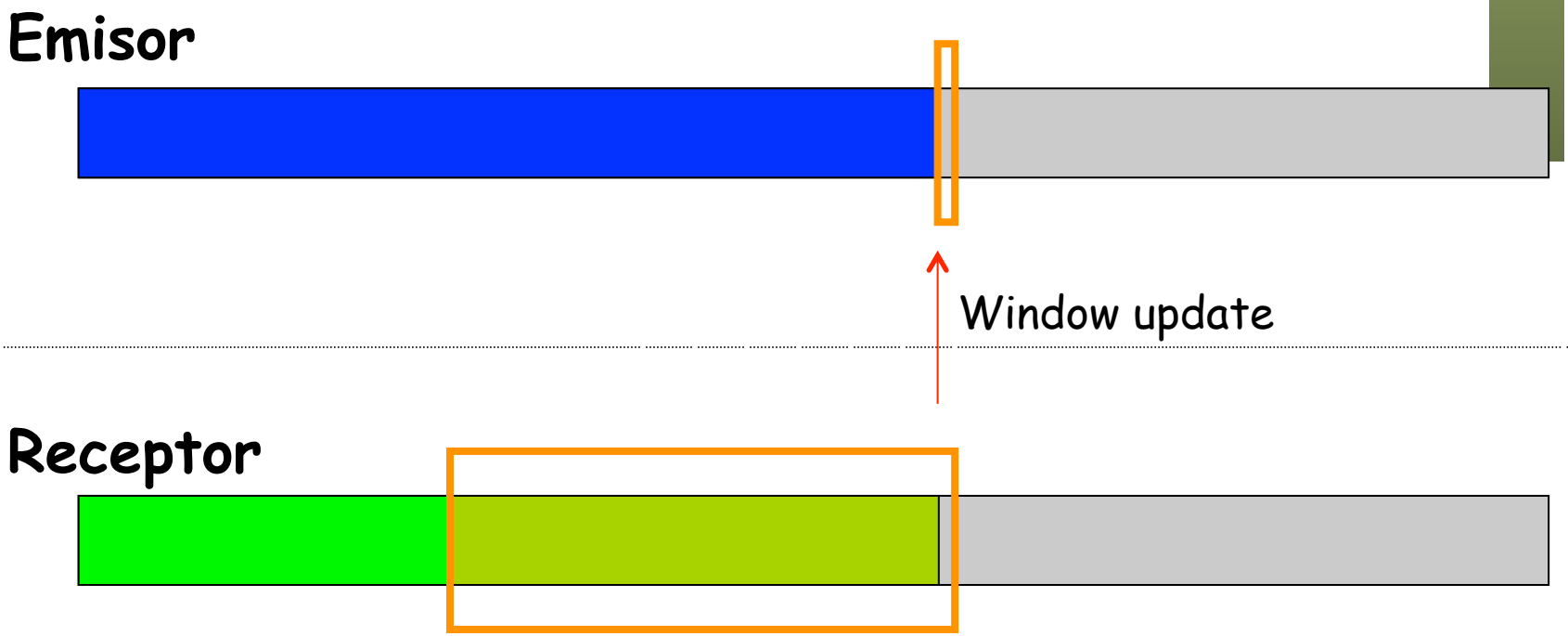
- Otro ejemplo:
  - El receptor está muy ocupado y la ventana cerrada
  - Receptor solo lee del buffer una pequeña cantidad de bytes (...)



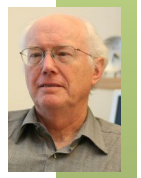
Ventana: SWS



- Otro ejemplo:
  - El receptor está muy ocupado y la ventana cerrada
  - Receptor solo lee del buffer una pequeña cantidad de bytes
  - Abre la ventana solo en esa cantidad con lo que solo se puede enviar un paquete pequeño
  - (...)

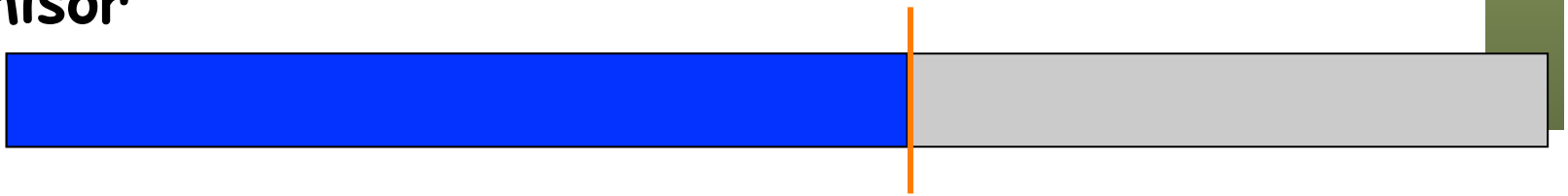


Ventana: SWS



- Otro ejemplo:
  - El receptor está muy ocupado y la ventana cerrada
  - Receptor solo lee del buffer una pequeña cantidad de bytes
  - Abre la ventana solo en esa cantidad con lo que solo se puede enviar un paquete pequeño
  - Para evitar esto puede retener el window update hasta poder abrir la ventana en todo un MSS

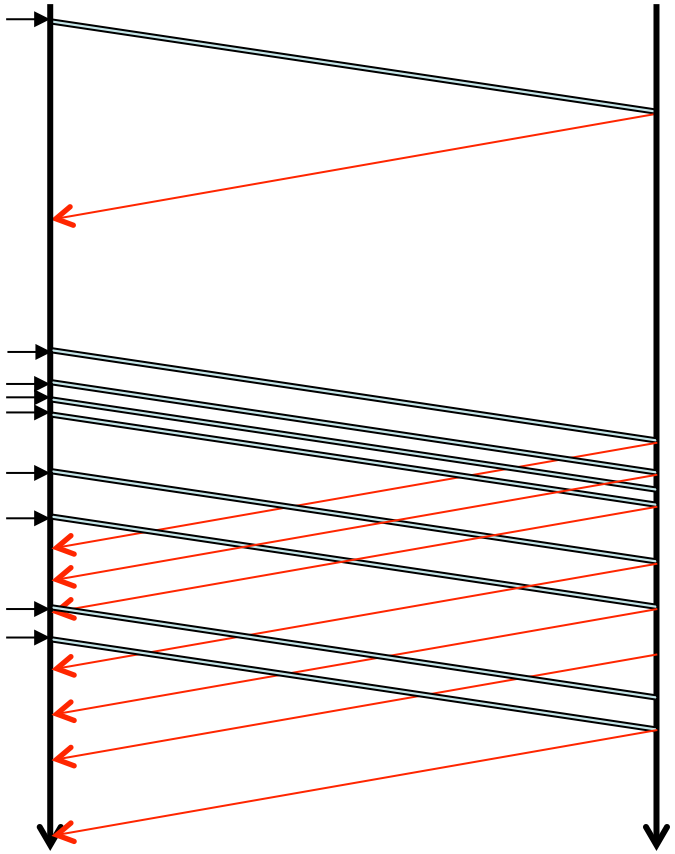
Emisor



Receptor

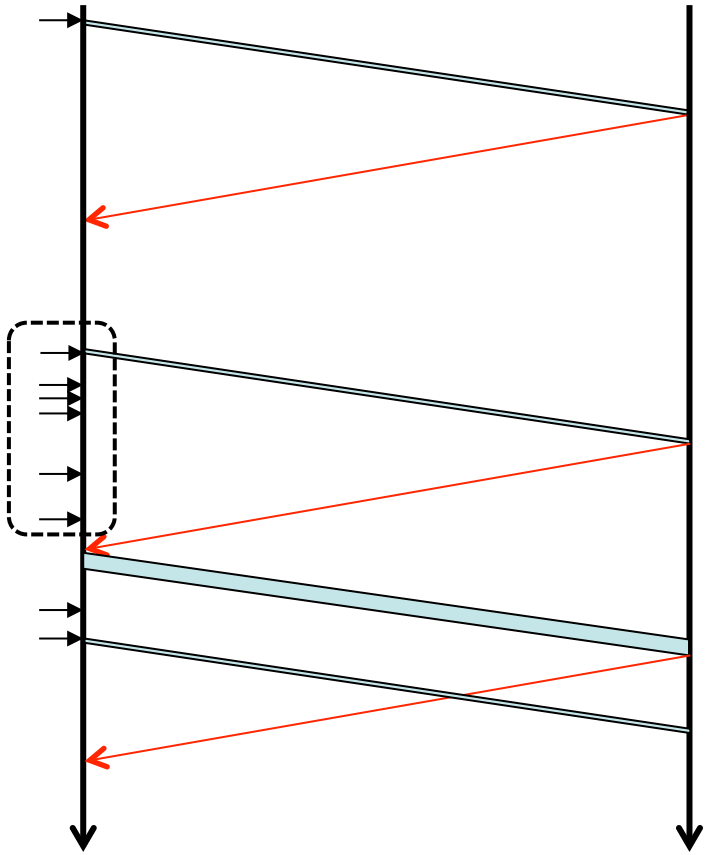
Algoritmo de Nagle

- Fuentes lentas o que generan datos en bloques pequeños
- Dan lugar a una gran cantidad de paquetes pequeños
- Mucha cabecera y pocos datos (1 byte de datos TCP + 20 de cabecera IP + 20 de cabecera TCP en el caso peor)
- (…)



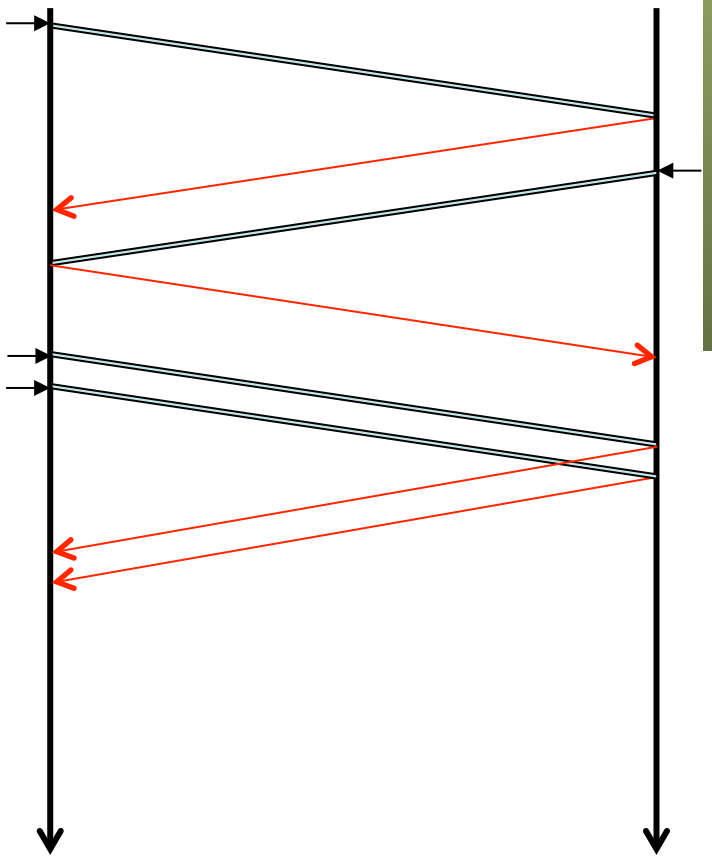
Algoritmo de Nagle

- Mientras haya datos en vuelo sin confirmar acumular datos
- Una vez que están confirmados todos se puede enviar lo acumulado
- O si se alcanza el MSS
- *Self-clocking*: si los ACKs llegan pronto se envían paquetes que pueden ser pequeños
- Ahorras capacidad en los enlaces
- Aumenta el retardo para la aplicación por el buffering
- No afecta si se envían bloques grandes
- Se puede desactivar si compensa el menor RTT frente al *overhead*



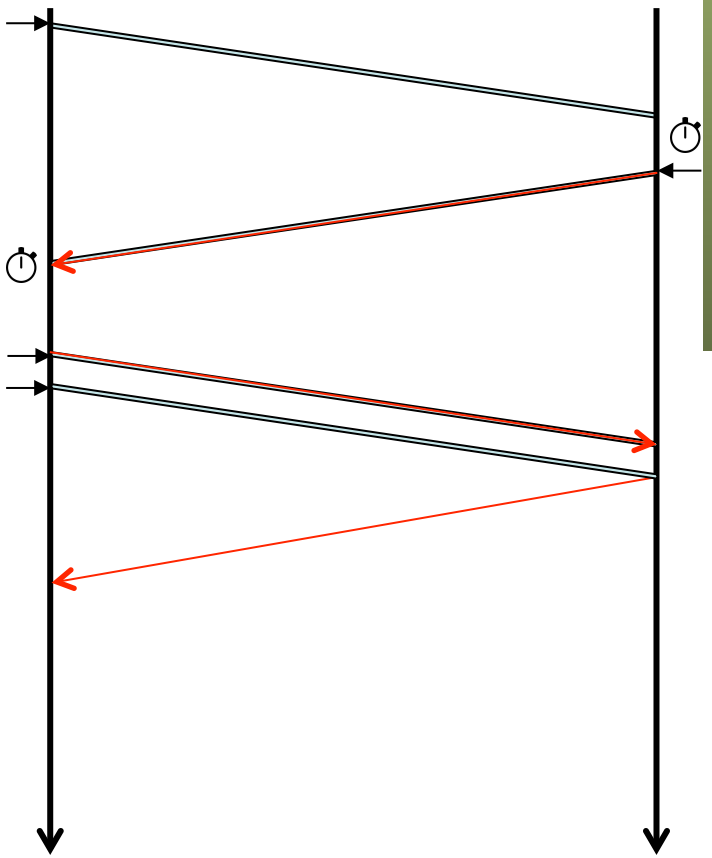
Delayed ACK

- Se envía un ACK por cada paquete de datos
- Los ACK son acumulativos, uno posterior repite la confirmación del anterior, luego no hacen falta tantos
- (...)



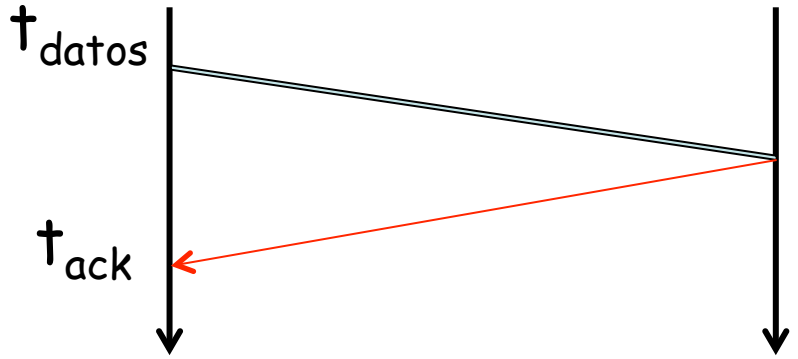
Delayed ACK

- Con *delayed ACK* al recibir datos espera antes de mandar ACK
- Así tal vez pueda hacerles *piggybacking* con datos en sentido contrario
- O reciba más datos y confirme todos de una vez
- Confirma recibe 2xRMSS bytes
- O si caduca un timer
- Timer debe ser < 500ms
- Típico 200ms o dinámico
- (Implementaciones con valores mucho menores)



RTO

- Retransmission TimeOut; bastantes cambios desde la RFC 793
- Timeout es el último recurso y vacía la red de segmentos
- Se basa en estimar el RTT con el tiempo entre datos y ACK
- Incorporarlo a medidas anteriores mediante un EWMA (Exponentially Weighted Moving Average)
- Y aumentar ese valor por un factor
- Acotado por un mínimo y máximo (típico entre 1s y 4min)
- Inicio (SYNs) RTO = 3s
- Al expirar: retx y RTO'=2xRTO (exponential backoff)



$t_{ack} - t_{datos} = RTT$

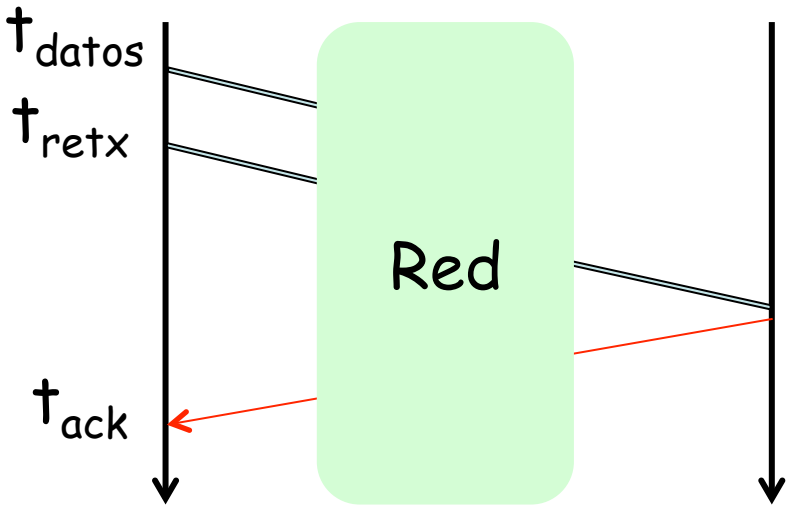
$SRTT' = \alpha\ SRTT + (1-\alpha)\ RTT$

$RTO = \beta\ SRTT'$

α = 0.8 a 0.9 (*smoothing factor*)
β = 1.3 a 2 (*delay variance factor*)
SRTT = "RTT suavizado"

RTO: Algoritmo de Karn

- Si hay retransmisión no se sabe a qué paquete corresponde el ACK
- Ignorar medida de RTT para paquetes que se han transmitido más de una vez
- El timer con *backoff* debido a la retransmisión se emplea para la próxima transmisión
- Recalcular el RTO solo cuando llegue un ACK sin retransmisión
- Si el segmento TCP emplea la opción timestamp sí se recalcula



Phil Karn

RTO: Nuevo cálculo

- V. Jacobson introduce estimación de la varianza
- Versión actual recomendada: RFC 6298 "Computing TCP's Retransmission Timer" (2011)
- Inicialmente RTO = 1s (SHOULD)
- Ante la primera medida de RTT:
  - SRTT = RTT
  - RTTVAR' = RTT / 2
  - RTO = SRTT + 4 RTTVAR' = RTT + 2 RTT = 3 RTT
- RTO mínimo de 1s (SHOULD, hay implementaciones a 200ms)
- Puede haber RTO máximo si es de al menos 60s
- Emplea algoritmo de Karn
- Si 4xRTTVAR'=0 usar la resolución del reloj empleado en timer

$$RTTVAR' = (1-\beta)\, RTTVAR + \beta\, (SRTT - RTT)$$

$$SRTT' = (1-\alpha)\, SRTT + \alpha\, RTT$$

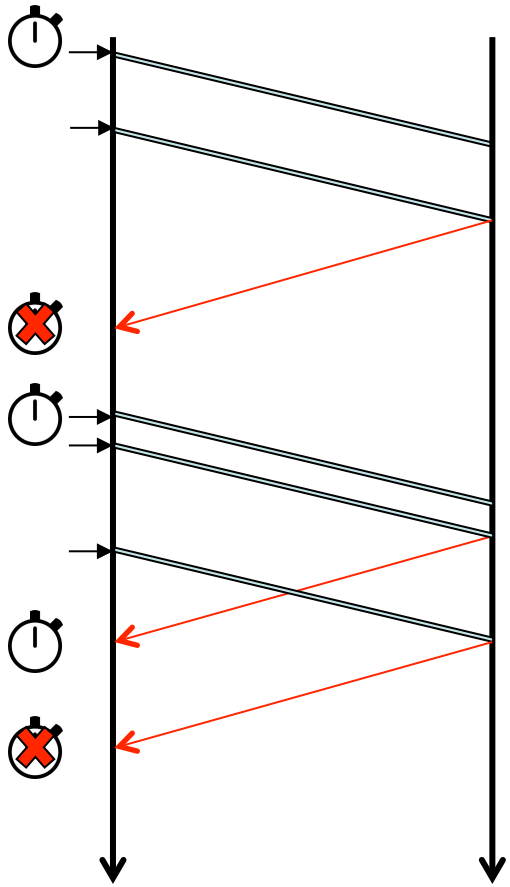
$$RTO = SRTT' + 4\, RTTVAR'$$

Ojo, en este orden

$\alpha = 1/8$
$\beta = 1/4$

RTO: ¿Por paquete?

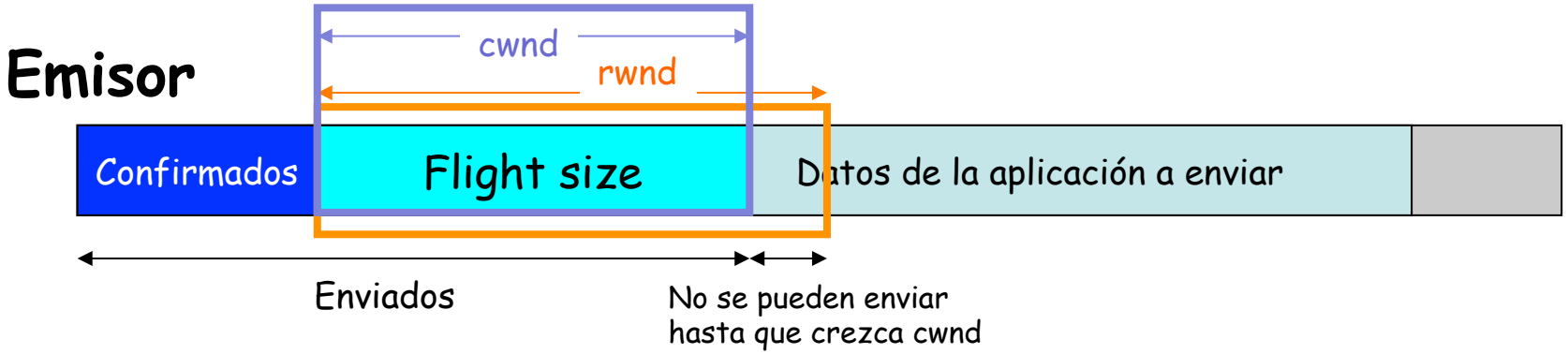
- ¿Un timer por cada paquete de datos enviado? ¿O solo un timer activo a la vez?
- Recomendación para un solo timer (RFC 6298)
  - Si se envía un paquete de datos y no hay timer activo activarlo
  - Cuando se confirman todos los datos enviados desactivar timer
  - Cuando un ACK confirma datos reiniciar el timer con nuevo RTO



Control de congestión en TCP

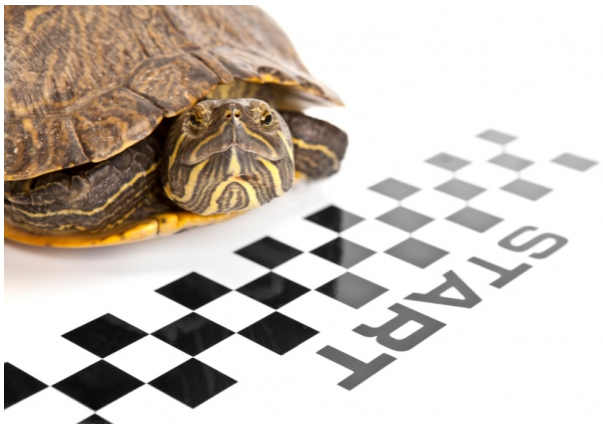
- Última versión RFC 5681 “TCP Congestion Control”
- Fundamentalmente 4 algoritmos:
  - Slow start
  - Congestion avoidance
  - Fast retransmit
  - Fast recovery

- Se puede ser más conservador pero no más agresivo enviando
- Los datos enviados sin confirmar (*flight size*) deben ser menos del mínimo entre:
  - *rwnd* : ventana de control de flujo anunciada por receptor
  - *cwnd* : ventana de congestión calculada por el emisor
- *ssthresh* : umbral entre slow start y congestion avoidance



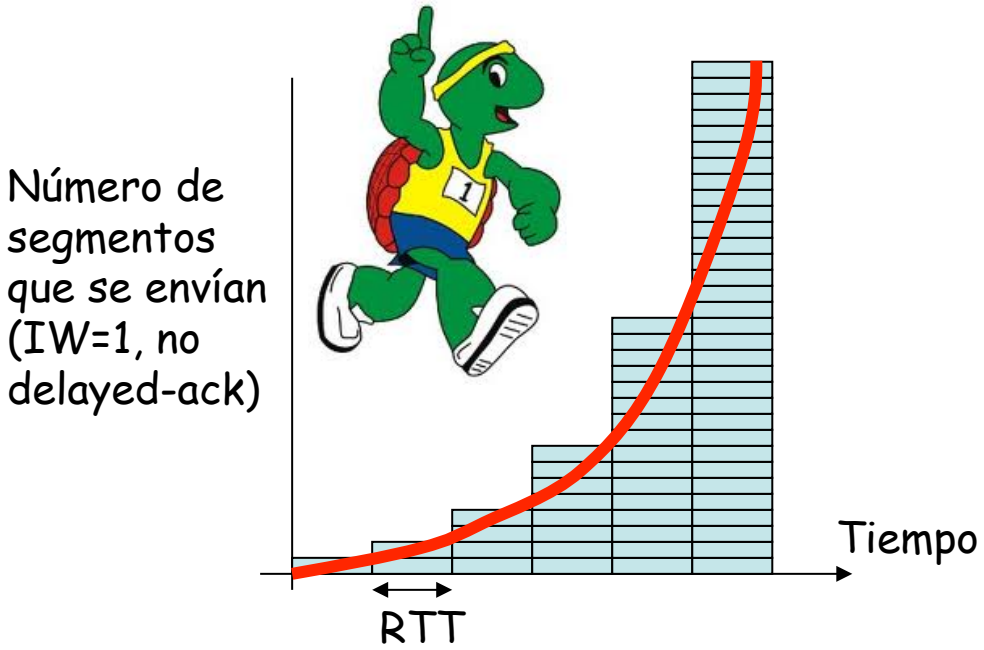
Slow start

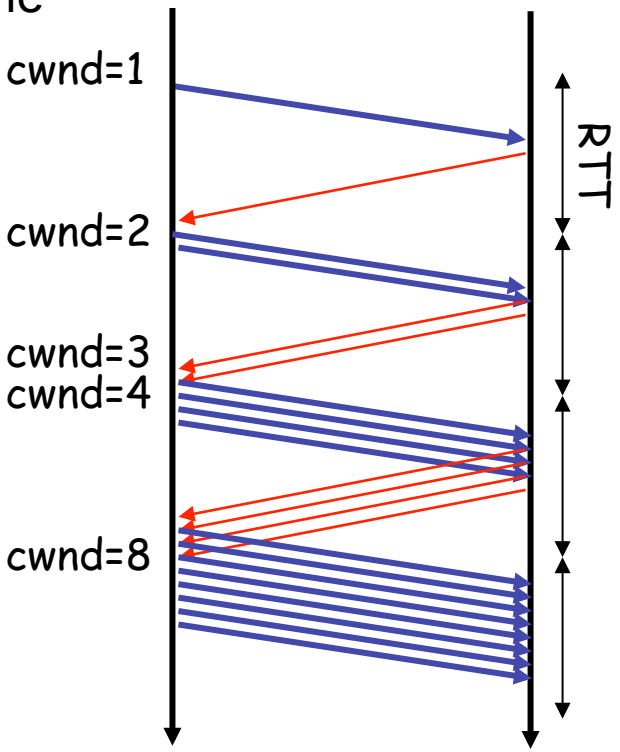
- Inicialmente TCP no conoce las condiciones de la red
- Comienza enviando en modo muy conservador (1 paquete?)
- Si ve que los datos llegan (recibe ACK) prueba a enviar más de golpe (tener más “en vuelo”)
- IW = *Initial Window*, valor inicial máximo de cwnd (MUST)
  - Tradicionalmente IW = 1 ó 2 segmentos pero cambiado en 2002:
  - Si (SMSS > 2190) : IW = 2xSMSS y no más de 2 segmentos
  - Si (2190 >= SMSS > 1095) : IW = 3xSMSS y no más de 3 sgmts.
  - Si (1095 >= SMSS) : IW = 4xSMSS y no más de 4 segmentos
  - Con Ethernet end2end SMSS puede ser 1460 ➔ IW=3 segmentos
  - Evita un *delayed ack* con un primer segmento único
- ssthresh comienza en valor muy grande



Slow start

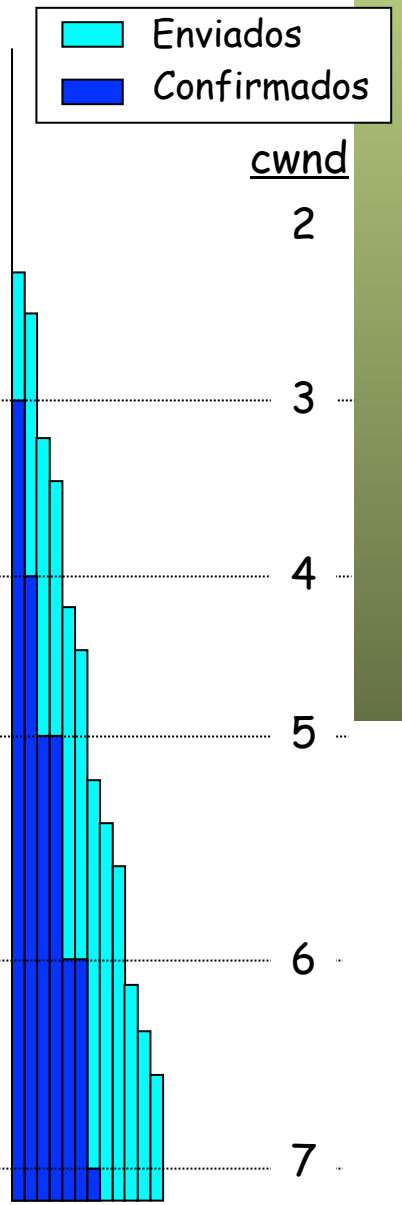
- TCP incrementa cwnd en como mucho SMSS por cada ACK que confirma nuevos datos
- Hasta que alcance/supere ssthresh o se detecte una pérdida
- Tradicionalmente se incrementaba en SMSS
- Ahora se recomienda incrementar en mín(bytes_ack'ed,SMSS)
- Esto protege contra "ACK Division"
- RFC 6928 (experimental 2013) propone IW=10





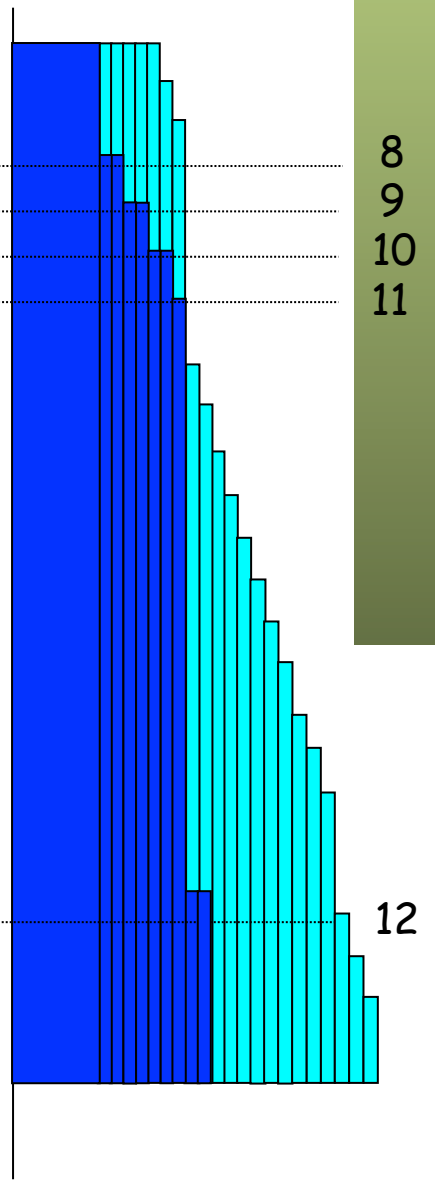
Ejemplo de *slow start*

```
1.1.1.11.2823 > 1.1.1.12.1509: S 0:0(0) win 32120 <mss 1460>
1.1.1.12.1509 > 1.1.1.11.2823: S 0:0(0) ack 1 win 32120  <mss 1460>
1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 1 win 32120

1.1.1.12.1509 > 1.1.1.11.2823: P 1:1449(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 1449:2897(1448) ack 1 win 32120

1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 1449 win 31856
1.1.1.12.1509 > 1.1.1.11.2823: P 2897:4345(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 4345:5793(1448) ack 1 win 32120

1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 2897 win 31856
1.1.1.12.1509 > 1.1.1.11.2823: P 5793:7241(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 7241:8689(1448) ack 1 win 32120

1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 5793 win 31856
1.1.1.12.1509 > 1.1.1.11.2823: P 8689:10137(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 10137:11585(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 11585:13033(1448) ack 1 win 32120

1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 8689 win 31856
1.1.1.12.1509 > 1.1.1.11.2823: P 13033:14481(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 14481:15929(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 15929:17377(1448) ack 1 win 32120

1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 10137 win 31856
```



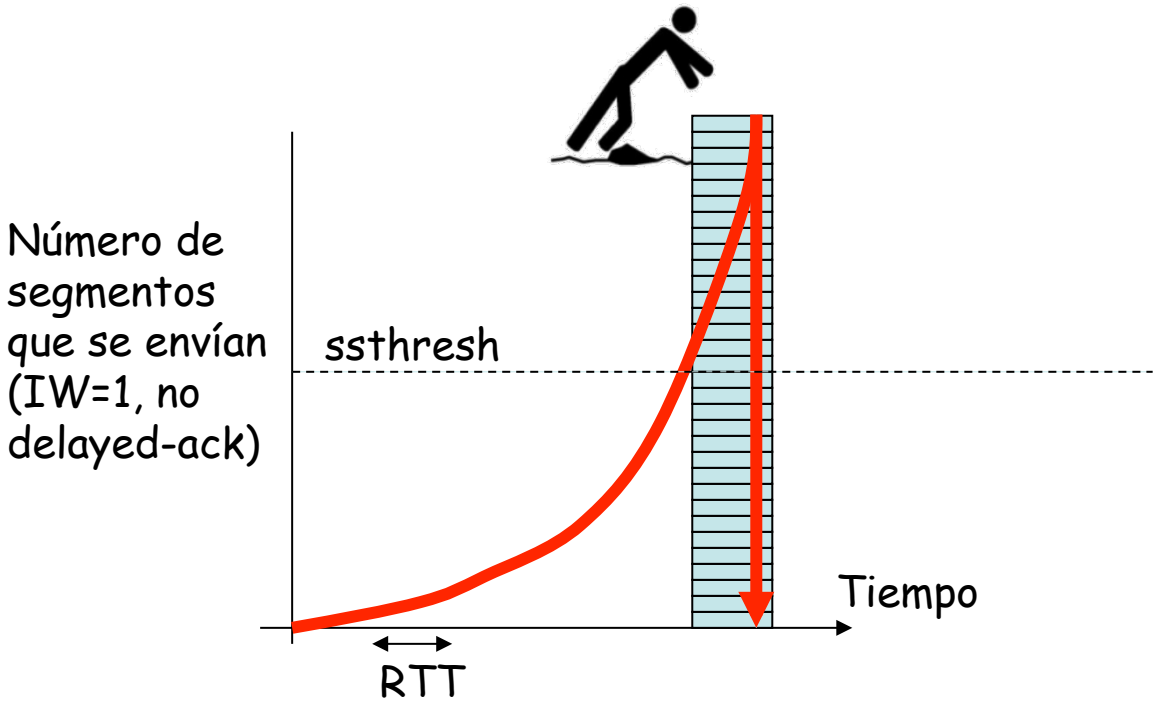
Ejemplo de *slow start*

```
1.1.1.12.1509 > 1.1.1.11.2823: P 17377:18825(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 18825:20001(1176) ack 1 win 32120
1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 13033 win 31856
1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 15929 win 31856
1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 18825 win 31856
1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 20001 win 31856

1.1.1.12.1509 > 1.1.1.11.2823: P 20001:21449(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 21449:22897(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 22897:24345(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 24345:25793(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 25793:27241(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 27241:28689(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 28689:30137(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 30137:31585(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 31585:33033(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 33033:34481(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: P 34481:35929(1448) ack 1 win 32120

1.1.1.11.2823 > 1.1.1.12.1509: . 1:1(0) ack 22897 win 31856
1.1.1.12.1509 > 1.1.1.11.2823: . 35929:37377(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: . 37377:38825(1448) ack 1 win 32120
1.1.1.12.1509 > 1.1.1.11.2823: . 38825:40273(1448) ack 1 win 32120
```



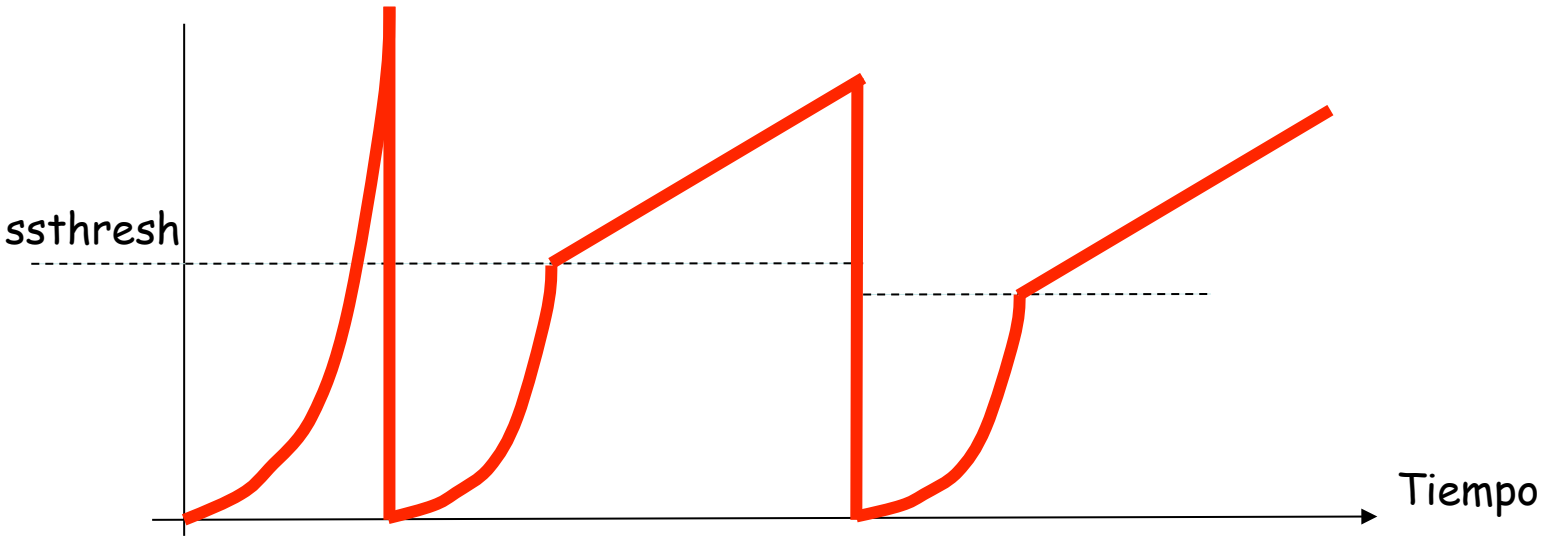
Pérdidas por timeout

- Al detectar pérdidas por timeout:
  - cwnd = SMSS (independiente de IW)
  - Si los datos no se había retransmitido ya por timeout:
    ssthresh = máx(FlightSize / 2, 2xSMSS)
  - Si ya se habían retransmitido no se cambia ssthresh



Congestion avoidance

- Cuando cwnd es mayor que ssthresh
- Se incrementa cwnd en 1 SMSS por cada RTT o en mín(bytes_ack'ed,SMSS)

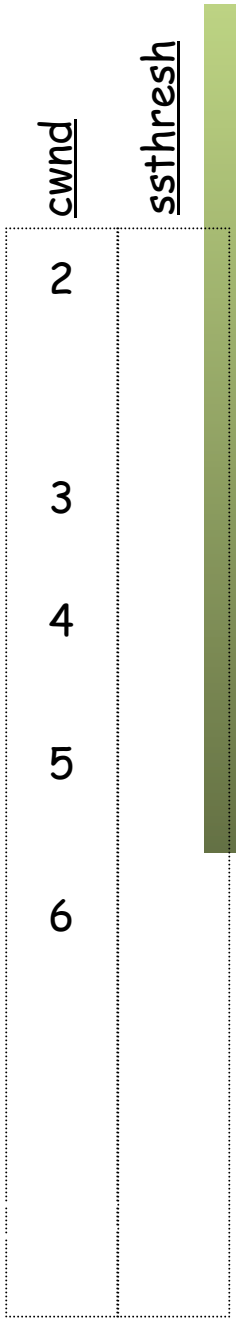


Congestion avoidance

- Cuando cwnd es mayor que ssthresh
- Se incrementa cwnd en 1 SMSS por cada RTT o en mín(bytes_ack'ed,SMSS)

Ejemplo Tahoe (I)

```
753.246362 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: S 0:0(0) win 32120
753.246536 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: S 0:0(0) ack 1

753.468594 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1:205(204) ack 1
753.468750 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 205:409(204) ack 1

754.291021 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 205
754.291137 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 409:613(204) ack 1
754.291257 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 613:817(204) ack 1
754.746127 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 409
754.746234 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 817:1021(204) ack 1
754.746353 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1021:1225(204) ack 1
755.832827 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 817
755.832948 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1225:1429(204) ack 1
755.833066 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1429:1633(204) ack 1
755.833182 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1633:1837(204) ack 1
756.327987 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 1225
756.328105 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1837:2041(204) ack 1
756.328220 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 2041:2245(204) ack 1
756.328333 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 2245:2449(204) ack 1

(...)
```



Ejemplo Tahoe (I)

```
753.246362 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: S 0:0(0) win 32120
753.246536 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: S 0:0(0) ack 1

753.468594 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1:205(204) ack 1
753.468750 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 205:409(204) ack 1

754.291021 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 205
754.291137 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 409:613(204) ack 1
754.291257 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 613:817(204) ack 1
754.746127 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 409
754.746234 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 817:1021(204) ack 1
754.746353 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1021:1225(204) ack 1
755.832827 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 817
755.832948 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1225:1429(204) ack 1
755.833066 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1429:1633(204) ack 1
755.833182 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1633:1837(204) ack 1
756.327987 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 1225
756.328105 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1837:2041(204) ack 1
756.328220 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 2041:2245(204) ack 1
756.328333 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 2245:2449(204) ack 1

760.697798 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1225:1429(204) ack 1

761.796804 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 1429
```

cwnd	ssthresh
2	
3	
4	
5	
6	
1	3
2	

Ejemplo Tahoe (II)

```
761.796932 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1429:1633(204) ack 1
761.797054 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1633:1837(204) ack 1
762.838579 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 1837

762.838691 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 1837:2041(204) ack 1
762.838807 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 2041:2245(204) ack 1
762.838923 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 2245:2449(204) ack 1

764.101234 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 2245

764.827096 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 2449

764.827200 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 2449:2653(204) ack 1
764.827315 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 2653:2857(204) ack 1
764.827428 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 2857:3061(204) ack 1
764.827541 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 3061:3265(204) ack 1
766.320277 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 2857

766.320406 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 3265:3469(204) ack 1
766.320523 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 3469:3673(204) ack 1
766.758919 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 3265

766.759024 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 3673:3877(204) ack 1
766.759141 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 3877:4081(204) ack 1
767.859805 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 3673
```

cwnd	ssthresh
3	3
4	
4 (1)	
4 (2)	
4 (3)	
4 (4)	

Ejemplo Tahoe (III)

```
767.859926 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 4081:4285(204) ack 1
767.860043 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 4285:4489(204) ack 1
768.359065 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 4081

768.359188 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 4489:4693(204) ack 1
768.359305 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 4693:4897(204) ack 1
768.359418 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 4897:5101(204) ack 1
768.899165 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 4489

768.899311 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 5101:5305(204) ack 1
768.899429 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 5305:5509(204) ack 1
770.122698 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 4897

770.122858 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 5509:5713(204) ack 1
770.122975 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 5713:5917(204) ack 1
770.614382 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 5305

770.614528 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 5917:6121(204) ack 1
770.614644 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 6121:6325(204) ack 1
771.347724 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 5509

771.347856 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 6325:6529(204) ack 1
771.659393 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 5917
```

cwnd	ssthresh
	3
5	
5 (1)	
5 (2)	
5 (3)	
5 (4)	
5 (5)	

Ejemplo Tahoe (y IV)

```
771.659497 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 6529:6733(204) ack 1
771.659613 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 6733:6937(204) ack 1
772.159445 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 6325

772.159599 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 6937:7141(204) ack 1
772.159715 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 7141:7345(204) ack 1
772.159829 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 7345:7549(204) ack 1
772.699491 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 6733

772.699660 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 7549:7753(204) ack 1
772.699781 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 7753:7957(204) ack 1
1773.397932 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 6937

773.398089 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 7957:8161(204) ack 1
773.919587 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 7345

773.919734 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 8161:8365(204) ack 1
773.919851 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 8365:8569(204) ack 1
774.409619 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 7753

774.409770 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 8569:8773(204) ack 1
774.409886 eth0 > 1.1.1.61.3014 > 10.1.11.1.1069: P 8773:8977(204) ack 1
774.909675 eth0 < 10.1.11.1.1069 > 1.1.1.61.3014: . 1:1(0) ack 8161
```

cwnd	ssthresh
	3
6	
6 (1)	
6 (2)	
6 (3)	
6 (4)	
6 (5)	

Congestion avoidance

- Cuando cwnd es mayor que ssthresh
- Se incrementa cwnd en 1 SMSS por cada RTT o en mín(bytes_ack'ed,SMSS)
- **Opción 1**
  - Contar los bytes confirmados
  - Cuando alcanza el valor de cwnd se incrementa la ventana
  - Requiere un contador adicional
- **Opción 2**
  - Con cada ACK que confirma nuevos datos actualizar
    cwnd += SMSS x SMSS / cwnd
  - No requiere contador adicional
  - Es una aproximación

Resumen

- TCP Tahoe
  - Slow start
  - Congestion avoidance
- Algoritmo de Nagle
- Silly Window Syndrome
- Cálculo del RTO