

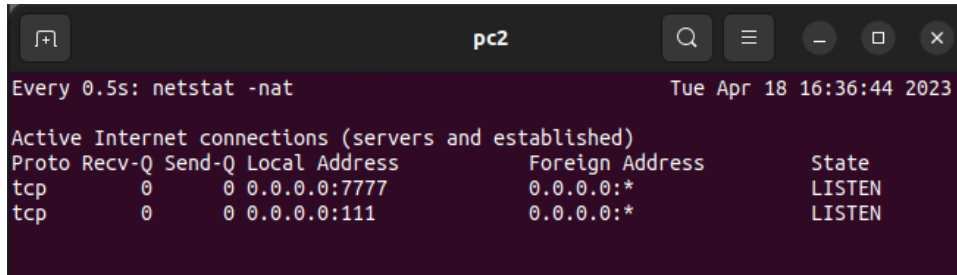
Práctica 4: Control de Congestión en TCP

1. Estados de una conexión TCP

1.1. Establecimiento de la conexión

Cuestiones 1-4.

Primero lanzamos el tcpdump en r1, luego el servidor en pc2.

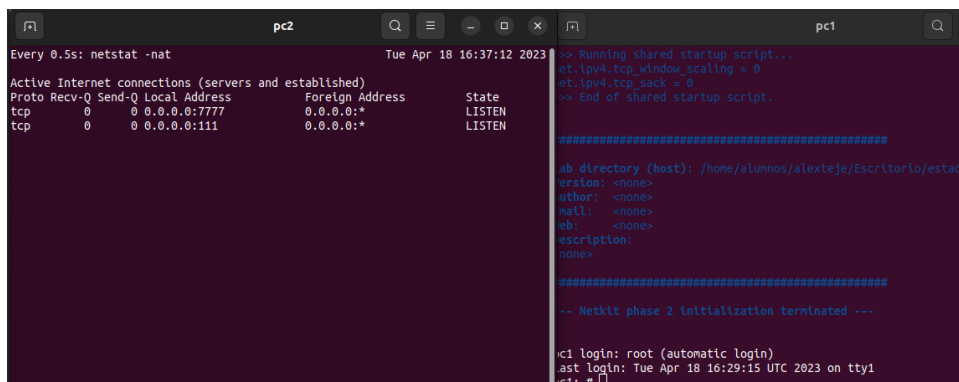


```
pc2
Every 0.5s: netstat -nat Tue Apr 18 16:36:44 2023

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:7777             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111              0.0.0.0:*               LISTEN
```

(watch -n 0.5 netstat -nat. 0,5 son los seg en los que se repite la acción.)

Para ver el estado en el que se encuentran las conexiones tcp de la máquina.



```
pc2
Every 0.5s: netstat -nat Tue Apr 18 16:37:12 2023

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:7777             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111              0.0.0.0:*               LISTEN

pc1
>> Running shared startup script...
et.ipv4.tcp_window_scaling = 0
et.ipv4.tcp_sack = 0
>> End of shared startup script.

=====
ab directory (host): /home/alumnos/alexteje/Escritorio/estado
version: <none>
author: <none>
mail: <none>
web: <none>
description:
none>

=====
-- Netkit phase 2 initialization terminated ---

pc1 login: root (automatic login)
Last login: Tue Apr 18 16:29:15 UTC 2023 on tty1
pc1:~#
```

Reacción del comando en pc1, no hace nada porque el cliente aún no está lanzado.

5. Explica en qué estado se encuentra la conexión en el servidor antes de arrancar el cliente.

EL servidor ya lo hemos lanzado y está en espera para que algún cliente se conecte a su puerto, mientras tanto no hace nada.

8. Observa en el tráfico capturado en r1 como se han intercambiado los 3 segmentos del establecimiento de la conexión

```
pc2
Every 0.5s: netstat -nat
Tue Apr 18 16:38:27 2023

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 12.0.0.2:7777           11.0.0.2:47472         ESTABLISHED

pc1
net.ipv4.tcp_window_scaling = 0
net.ipv4.tcp_sack = 0
>>> End of shared startup script.

#####

Lab directory (host): /home/alumnos/alexteje/Escritorio
Version: <none>
Author: <none>
Email: <none>
Web: <none>
Description:
<none>

#####

--- Netkit phase 2 initialization terminated ---

pc1 login: root (automatic login)
Last login: Tue Apr 18 16:29:15 UTC 2023 on tty1
pc1:~# nc 12.0.0.2 7777
```

```
r1
Author: <none>
Email: <none>
Web: <none>
Description:
<none>

#####

--- Netkit phase 2 initialization terminated ---

r1 login: root (automatic login)
Last login: Tue Apr 18 16:29:25 UTC 2023 on tty1
r1:~# tcpdump -i eth0 tcp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
16:37:36.697961 IP 11.0.0.2.47472 > 12.0.0.2.7777: S 901153382:901153382(0) win 5840 <mss 1460,nop,nop,timestamp 20253 0>
16:37:36.726001 IP 12.0.0.2.7777 > 11.0.0.2.47472: S 1113754533:1113754533(0) ack 901153383 win 36 <mss 1460,nop,nop,timestamp 20682 20253>
16:37:37.861629 IP 11.0.0.2.47472 > 12.0.0.2.7777: S 901153382:901153382(0) win 5840 <mss 1460,nop,nop,timestamp 20553 0>
16:37:37.861838 IP 12.0.0.2.7777 > 11.0.0.2.47472: S 1113754533:1113754533(0) ack 901153383 win 36 <mss 1460,nop,nop,timestamp 20795 20253>
16:37:38.527100 IP 11.0.0.2.47472 > 12.0.0.2.7777: . ack 1 win 5840 <nop,nop,timestamp 20618 20682>
16:37:39.667804 IP 11.0.0.2.47472 > 12.0.0.2.7777: . ack 1 win 5840 <nop,nop,timestamp 20733 20795>
```

9. Explica el valor de la ventana anunciada por el servidor.

En la pestaña de pc2 observamos la dirección de pc2 y el puerto correspondiente por el que se está ejerciendo el servidor. Que pone established, diciendo que ya se ha establecido la conexión cliente-servidor.

```
pc2
Every 0.5s: netstat -nat
Tue Apr 18 16:40:32 2023

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 12.0.0.2:7777           11.0.0.2:47472         ESTABLISHED
```

1.2. Intercambio de datos

3. Ejecuta en el lado servidor de nuevo el comando netstat de la siguiente forma:

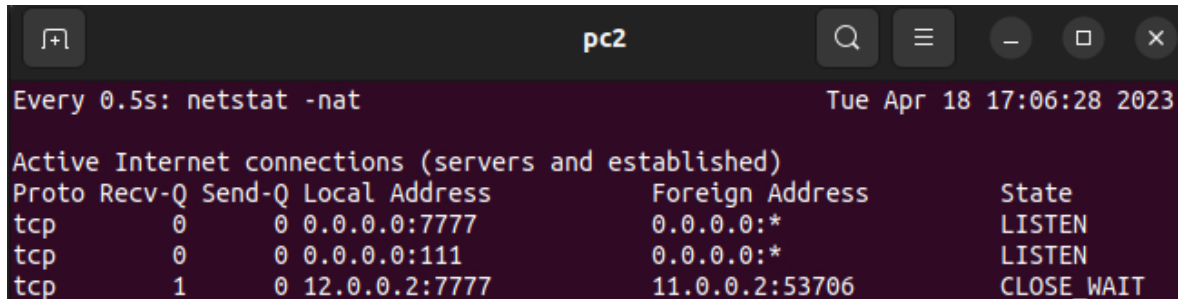
```
pc2
Every 0.5s: netstat -nat
Tue Apr 18 16:56:07 2023

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 12.0.0.2:7777           11.0.0.2:56073         ESTABLISHED
```

4. En el proceso nc del cliente en pc1 introduce una cadena de caracteres larga por la entrada estándar (por ejemplo, pulsa un rato la tecla de alguna letra hasta que aparezcan 2 líneas enteras de esa letra por pantalla y después pulsa la tecla Enter). Esta cadena de caracteres se recibirá en la implementación de TCP en el lado servidor, pero la aplicación no leerá estos datos porque se encuentra detenida.

Al estar detenido el servidor, si pc1 envía mensajes el servidor los recibe, pero no los muestra ya que en este momento teníamos el servidor en pausa, una vez que volvemos a iniciar los mensajes enviados por el cliente aparecen en el servidor.

1.3. Finalización de la conexión



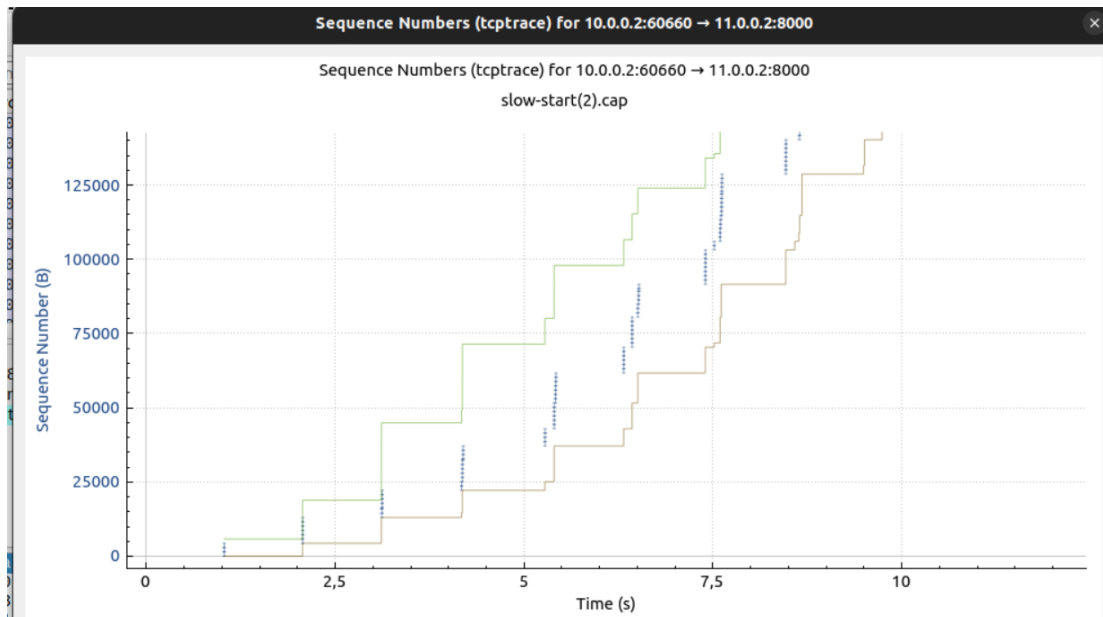
```
Every 0.5s: netstat -nat Tue Apr 18 17:06:28 2023
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:7777             0.0.0.0:*              LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*              LISTEN
tcp        1      0 12.0.0.2:7777           11.0.0.2:53706         CLOSE_WAIT
```

Una vez que se cierra el cliente, el servidor da por finalizada la conexión. Aunque tarda un tiempo en cerrar la conexión. Pasado ese tiempo se cierran todas las conexiones tcp del servidor

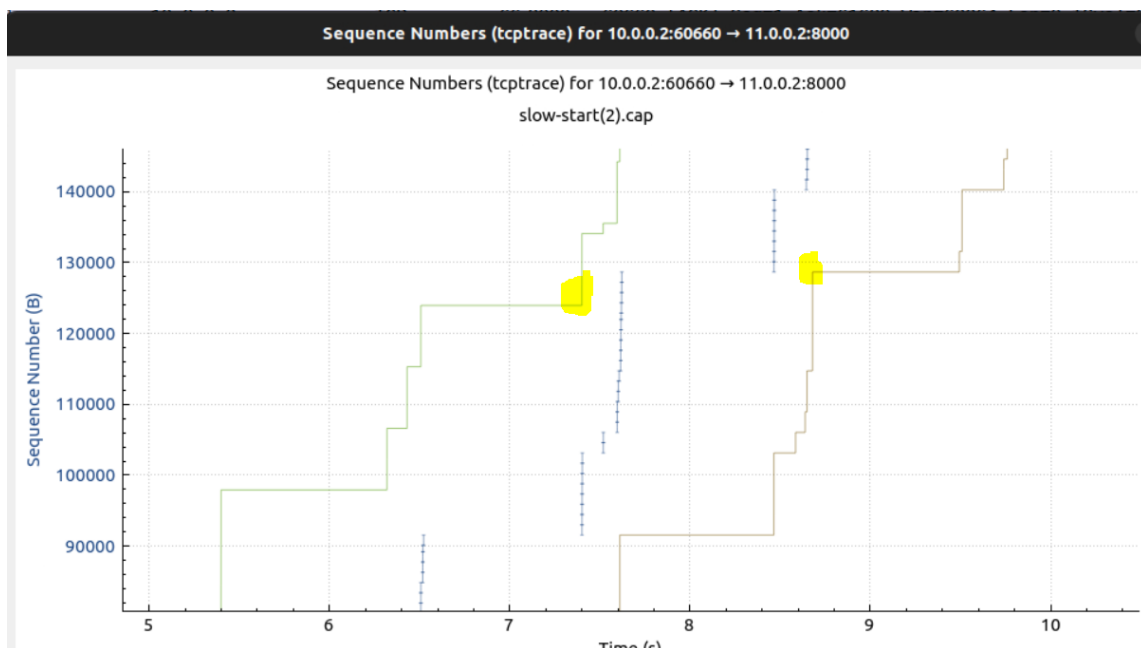
Información importante: El servidor nc está programado para terminar si el cliente le manda un segmento con el flag FIN activado, por este motivo, la aplicación nc en pc2 finaliza la conexión mandando su segmento FIN+ACK. Cuando el servidor recibe el último ACK de pc1 da por terminada la comunicación y finaliza su ejecución.

2. Slow Start en el inicio de la conexión

1. Indica el valor del flightsize en los siguientes instantes: 1'5s, 2'5s, 3'5s, 4'5s, 5s, 6s.



2. ¿En esta captura hay algún momento en que sea menor la ventana de control de flujo que la ventana de control de congestión?



En el segundo 7-9 ocurre que la ventana de ack enviados es menor que la ventana de ack asentidos, por lo que se están enviando más acks mientras que aún no se han asentido los anteriores.

3. ¿Por qué se envían inicialmente 3 segmentos con datos?

Porque al inicio de la conexión se observa el mms. Si este está entre 1095-2190 bytes, se pueden enviar como mucho 3 segmentos. En este caso el mms es 1460.

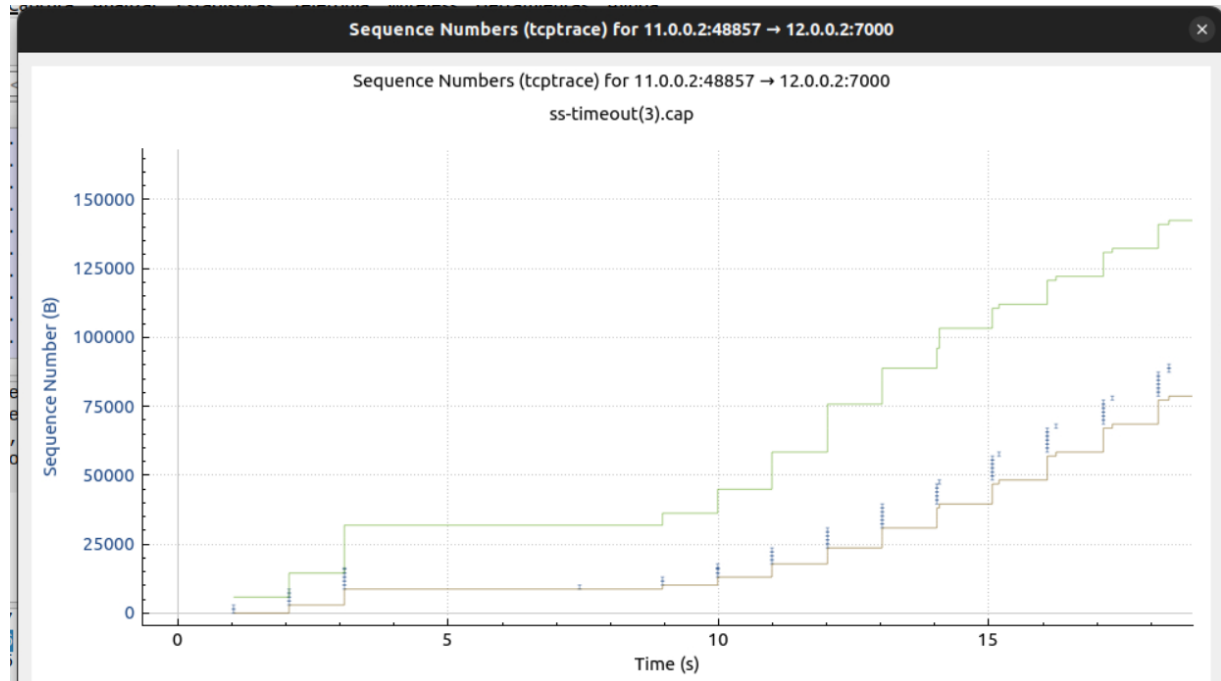
4. ¿Cuántos segmentos con nuevos datos se podrían enviar en el instante 3'5s?

Se podrían enviar segmentos hasta rellenar los bytes de datos del buffer, que los marca el paquete 2, que nos indica que leerla hasta el 5792. Es decir, se podrían enviar 4 paquetes de 1448 bytes.

5. ¿Cuántos segmentos con nuevos datos se podrían enviar en el instante 4'5s?

Se podrían enviar 3 segmentos hasta completar los bytes del buffer menos lo que ya se había enviado que está ocupando espacio (paquete 3 con len 0 y paquete 4 con len 1448), es decir, $5792 - 1448 = 4344$. Que son exactamente 3 paquetes.

3. Control de Congestión tras Timeout



1. Observa la captura y analiza cómo afecta Slow Start al envío de nuevos segmentos al inicio de la conexión. Entre el instante 3s y 3'5s se envían 6 segmentos. ¿Podrían haberse enviado más segmentos en ese instante? Razona la respuesta.

Si se pueden transmitir más paquetes ya que anteriormente teníamos 2 paquetes, con el ss se convierten en 4 y más tarde en 8, pero como solo se han enviado 6 paquetes, se podrían enviar dos más.

2. Localiza la retransmisión que se produce alrededor del instante 7'5s. Estudia el comportamiento de TCP tras el timeout. Analiza tanto la gráfica como cada uno de los segmentos. Responde a las siguientes preguntas:

a) ¿Qué número de secuencia tiene el segmento retransmitido?

El número de seq es 8689. Es la retrasmisión del paquete 15 enviado anteriormente.

b) ¿Qué valor se calcula para el umbral ssthresh?

$ssthresh := \max(\text{flightSize}/2, 2 * MSS)$

siendo flightSize los bytes en vuelo, es decir, los enviados y no asentidos hasta ese momento

c) ¿Qué tamaño tiene la ventana de congestión en el instante 8s?

$31857 - 23625 = 8232$.

Ya que los paquetes enviados anteriormente (15,16,18,21,22,23) no se han asentido aun y siguen ocupando espacio, y la retrasmisión del paquete 15 no ocupa espacio al ser una retrasmisión.

d) ¿Por qué se envían 2 segmentos alrededor del instante 9s?

Porque no se ha recibido un ack de afirmación de llegada sobre los paquetes 16 y 18.

e) ¿Podría enviarse algún segmento más en el instante 9s? ¿Por qué?

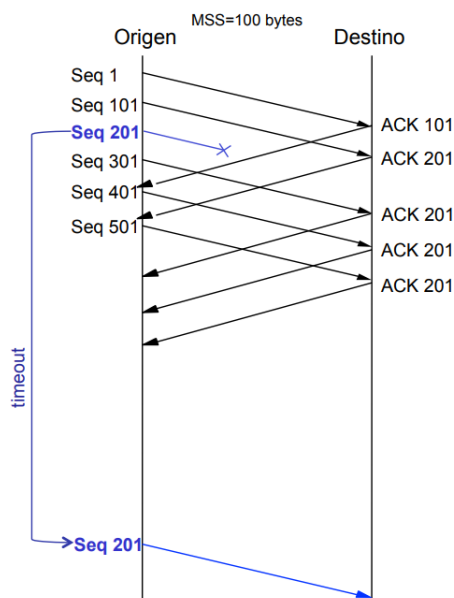
Si, cualquier otro paquete ya que la ventana del buffer aún tiene espacio.

f) ¿En qué modo de control de congestión se haya la conexión en el instante 5s? ¿Por qué?

En **timeout** o slow start reset.

Ejemplo importante:

Los timeouts provocan períodos grandes de inactividad. Tras la pérdida aún no detectada de un paquete (después de enviar el segmento número 201) se siguen enviando tantos paquetes como permite la ventana efectiva, hasta que se hace cero (después de enviar el segmento número 501). Desde ese momento hasta que vence el timeout el emisor se para: no se puede enviar nada nuevo.

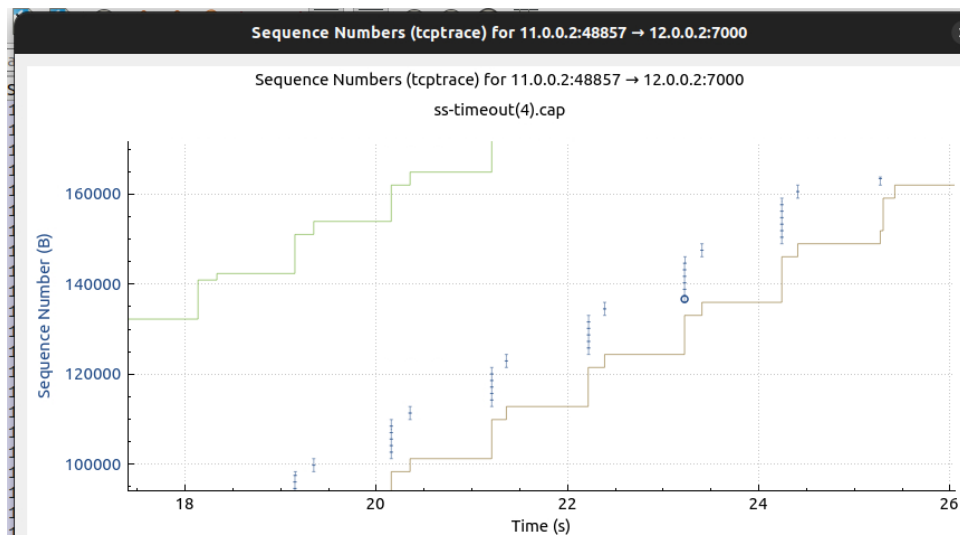


g) ¿En qué modo de control de congestión se haya la conexión en el instante 10'5s? ¿Por qué?

Se encuentra en modo fast recovery. Porque podemos ver como se retrasmite un paquete, se asiente y asi sucesivamente con todos los paquetes que no habían llegado hasta que ya se asienten todos.

21	3.580703	11.0.0.2	12.0.0.2	TCP	1514	48857 → 7000	[ACK] Seq=13033 Ack=1 Win=5840 Len=1448 TSval=
22	3.580735	11.0.0.2	12.0.0.2	TCP	1514	48857 → 7000	[ACK] Seq=14481 Ack=1 Win=5840 Len=1448 TSval=
23	3.580762	11.0.0.2	12.0.0.2	TCP	522	48857 → 7000	[PSH, ACK] Seq=15929 Ack=1 Win=5840 Len=456 TS
24	6.513940	c6:bb:73:b5:6d:08	46:a7:56:e6:7a:61	ARP	42	Who has 11.0.0.2? Tell 11.0.0.1	
25	6.514003	46:a7:56:e6:7a:61	c6:bb:73:b5:6d:08	ARP	42	11.0.0.2 is at 46:a7:56:e6:7a:61	
26	7.929096	11.0.0.2	12.0.0.2	TCP	1514	[TCP Retransmission] 48857 → 7000	[ACK] Seq=8689 Ack=1 Win=
27	9.464425	12.0.0.2	11.0.0.2	TCP	66	7000 → 48857	[ACK] Seq=1 Ack=10137 Win=26064 Len=0 TSval=34
28	9.464505	11.0.0.2	12.0.0.2	TCP	1514	[TCP Retransmission] 48857 → 7000	[PSH, ACK] Seq=10137 Ack=
29	9.464539	11.0.0.2	12.0.0.2	TCP	1514	[TCP Retransmission] 48857 → 7000	[PSH, ACK] Seq=11585 Ack=
30	10.485977	12.0.0.2	11.0.0.2	TCP	66	7000 → 48857	[ACK] Seq=1 Ack=11585 Win=28960 Len=0 TSval=34
31	10.486077	11.0.0.2	12.0.0.2	TCP	1514	[TCP Retransmission] 48857 → 7000	[ACK] Seq=13033 Ack=1 Win=
32	10.486111	11.0.0.2	12.0.0.2	TCP	1514	[TCP Retransmission] 48857 → 7000	[ACK] Seq=14481 Ack=1 Win=
33	10.486000	12.0.0.2	11.0.0.2	TCP	66	7000 → 48857	[ACK] Seq=1 Ack=13033 Win=31856 Len=0 TSval=34
34	10.486153	11.0.0.2	12.0.0.2	TCP	522	[TCP Retransmission] 48857 → 7000	[PSH, ACK] Seq=15929 Ack=
35	10.486185	11.0.0.2	12.0.0.2	TCP	1514	48857 → 7000	[ACK] Seq=16385 Ack=1 Win=5840 Len=1448 TSval=
36	11.488394	12.0.0.2	11.0.0.2	TCP	66	7000 → 48857	[ACK] Seq=1 Ack=14481 Win=34752 Len=0 TSval=34

3. ¿En qué modo de control de congestión termina la conexión?



Acaba en CA (Congestion Avoidance).

4. ¿Qué valor aproximado tiene la ventana de congestión al final de la conexión?

Hay que mirar la ventana de control de congestión (cwnd).

Como el mms es 1460, la ventana de congestion $cwnd = 3 * MSS$ y no debe ser mayor de 3 segmentos, es decir, 4380.

4. Fast Retransmit / Fast Recovery

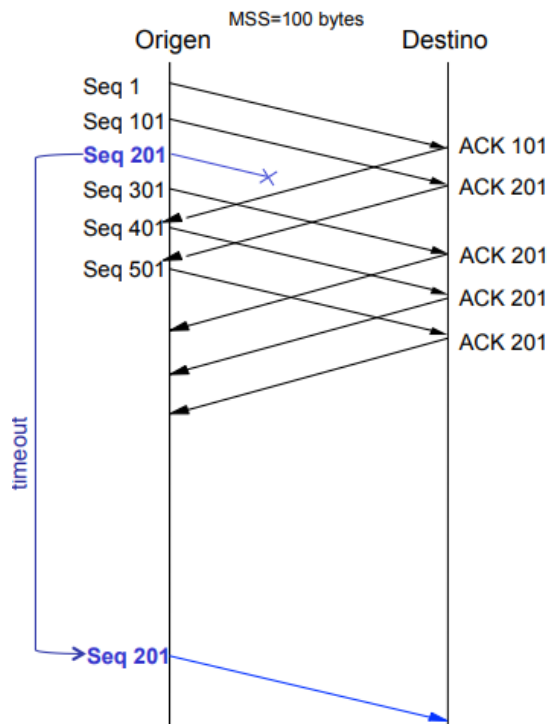
1. ¿Cuántas retransmisiones debidas a timeout se observan en la traza? Identifica en qué instante se producen

Cerca del segundo 3 se envía un paquete con numero de secuencia 17833, no se recibe el ack de asentimiento, pero mientras se siguen enviando más mensajes, pero mientras se sigue enviando un mensaje de que es necesario el asentimiento del primer mensaje. Hasta que ya se envia el ack de ese mensaje o uno con un ack mayor.

No.	Time	Source	Destination	Protocol	Length	Info
37	4.033670	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=29417 Ack=1 Win=5840 Len=1448 TSval=
38	4.047928	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=17833 Win=40544 Len=0 TSval=40
40	4.047952	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#1] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=4
41	4.047975	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#2] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=4
39	4.048046	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [PSH, ACK] Seq=30865 Ack=1 Win=5840 Len=1448 TS
42	4.048981	11.0.0.2	12.0.0.2	TCP	522	42413 → 8000 [PSH, ACK] Seq=32313 Ack=1 Win=5840 Len=456 TS
43	4.049371	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=32769 Ack=1 Win=5840 Len=1448 TSval=
44	4.049399	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=34217 Ack=1 Win=5840 Len=1448 TSval=
45	5.040211	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#3] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=4
47	5.040235	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#4] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=4
48	5.040258	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#5] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=4
50	5.040280	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#6] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=4
51	5.040304	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#7] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=4
53	5.040325	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#8] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=4
46	5.040427	11.0.0.2	12.0.0.2	TCP	1514	[TCP Fast Retransmission] 42413 → 8000 [ACK] Seq=17833 Ack=
49	5.040485	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=35665 Ack=1 Win=5840 Len=1448 TSval=
52	5.040521	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=37113 Ack=1 Win=5840 Len=1448 TSval=
54	5.071021	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#9] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=4
56	5.071037	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#10] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=
57	5.071062	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#11] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=
59	5.071065	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#12] 8000 → 42413 [ACK] Seq=1 Ack=17833 Win=
55	5.071140	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=38561 Ack=1 Win=5840 Len=1448 TSval=
58	5.071171	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=40009 Ack=1 Win=5840 Len=1448 TSval=
60	6.003050	c6:bb:73:b5:6d:08	46:a7:56:e6:7a:61	ARP	42	Who has 11.0.0.2? Tell 11.0.0.1
61	6.003096	46:a7:56:e6:7a:61	c6:bb:73:b5:6d:08	ARP	42	11.0.0.2 is at 46:a7:56:e6:7a:61
62	6.048612	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=35665 Win=43440 Len=0 TSval=40
64	6.048696	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=37113 Win=46336 Len=0 TSval=40
66	6.048745	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=38561 Win=44888 Len=0 TSval=40
63	6.048835	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=41457 Ack=1 Win=5840 Len=1448 TSval=

En el segundo 8-9 ocurre lo mismo con otro paquete.

No.	Time	Source	Destination	Protocol	Length	Info
99	8.109534	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [PSH, ACK] Seq=67521 Ack=1 Win=5840 Len=1448 TS
101	8.109573	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=68969 Ack=1 Win=5840 Len=1448 TSval=
102	8.109604	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=70417 Ack=1 Win=5840 Len=1448 TSval=
103	9.081527	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=63177 Win=62264 Len=0 TSval=40
106	9.081552	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=66073 Win=63712 Len=0 TSval=40
104	9.081625	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=71865 Ack=1 Win=5840 Len=1448 TSval=
105	9.081658	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=73313 Ack=1 Win=5840 Len=1448 TSval=
107	9.081698	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=74761 Ack=1 Win=5840 Len=1448 TSval=
108	9.081729	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=76209 Ack=1 Win=5840 Len=1448 TSval=
109	9.111915	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=68969 Win=60816 Len=0 TSval=40
110	9.111982	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=77657 Ack=1 Win=5840 Len=1448 TSval=
111	9.112014	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [PSH, ACK] Seq=79105 Ack=1 Win=5840 Len=1448 TS
112	10.008921	12.0.0.2	11.0.0.2	TCP	66	[TCP Window Update] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
113	10.008944	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 109#1] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
115	10.008968	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 109#2] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
116	10.008989	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 109#3] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
114	10.009066	11.0.0.2	12.0.0.2	TCP	1514	[TCP Out-Of-Order] 42413 → 8000 [ACK] Seq=68969 Ack=1 Win=5
117	10.009138	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=80553 Ack=1 Win=5840 Len=1448 TSval=
118	10.132460	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 109#4] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
119	10.132476	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 109#5] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
120	10.132539	11.0.0.2	12.0.0.2	TCP	1514	1514 42413 → 8000 [ACK] Seq=82001 Ack=1 Win=5840 Len=1448 TSval=
121	11.103940	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=80553 Win=55024 Len=0 TSval=40



2. ¿Cuántas retransmisiones debidas a Fast Retransmit se observan en la traza? Identifica en qué instante se producen.

Primer fast retransmit.

39	4.033280	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=22177	Ack=1	Win=5840	Len=1448	TSval=
31	4.033311	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=23625	Ack=1	Win=5840	Len=1448	TSval=
33	4.033345	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=25073	Ack=1	Win=5840	Len=1448	TSval=
34	4.033373	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=26521	Ack=1	Win=5840	Len=1448	TSval=
36	4.033406	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=27969	Ack=1	Win=5840	Len=1448	TSval=
37	4.033670	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=29417	Ack=1	Win=5840	Len=1448	TSval=
38	4.047928	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=40544	Len=0	TSval=40
49	4.047952	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#1]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
41	4.047975	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#2]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
39	4.048046	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[PSH, ACK]	Seq=30865	Ack=1	Win=5840	Len=1448	TSval=
42	4.048981	11.0.0.2	12.0.0.2	TCP	522	42413 → 8000	[PSH, ACK]	Seq=32313	Ack=1	Win=5840	Len=456	TSval=
43	4.049371	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=32769	Ack=1	Win=5840	Len=1448	TSval=
44	4.049399	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=34217	Ack=1	Win=5840	Len=1448	TSval=
45	5.040211	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#3]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
47	5.040235	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#4]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
48	5.040258	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#5]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
50	5.040280	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#6]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
51	5.040304	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#7]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
53	5.040325	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#8]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
46	5.040427	11.0.0.2	12.0.0.2	TCP	1514	[TCP Fast Retransmission]	42413 → 8000	[ACK]	Seq=17833	Ack=		
49	5.040485	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=35665	Ack=1	Win=5840	Len=1448	TSval=
52	5.040521	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=37113	Ack=1	Win=5840	Len=1448	TSval=
54	5.071021	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#9]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
56	5.071037	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#10]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
57	5.071052	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#11]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
59	5.071065	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 38#12]	8000 → 42413	[ACK]	Seq=1	Ack=17833	Win=4	
55	5.071140	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=38561	Ack=1	Win=5840	Len=1448	TSval=
58	5.071171	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=40009	Ack=1	Win=5840	Len=1448	TSval=

Segundo fast retransmit.

No.	Time	Source	Destination	Protocol	Length	Info
105	9.081658	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000 [ACK] Seq=73313 Ack=1 Win=5840 Len=1448 TSval=
107	9.081698	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000 [ACK] Seq=74761 Ack=1 Win=5840 Len=1448 TSval=
108	9.081729	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000 [ACK] Seq=76209 Ack=1 Win=5840 Len=1448 TSval=
109	9.111915	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=68969 Win=60816 Len=0 TSval=40,
110	9.111982	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000 [ACK] Seq=77657 Ack=1 Win=5840 Len=1448 TSval=
111	9.112014	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000 [PSH, ACK] Seq=79105 Ack=1 Win=5840 Len=1448 T,
112	10.088921	12.0.0.2	11.0.0.2	TCP	66	[TCP Window Update] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
113	10.088944	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 109#1] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
115	10.088968	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 109#2] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
116	10.088989	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 109#3] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
114	10.089066	11.0.0.2	12.0.0.2	TCP	1514	[TCP Out-Of-Order] 42413 → 8000 [ACK] Seq=68969 Ack=1 Win=5
117	10.089138	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000 [ACK] Seq=80553 Ack=1 Win=5840 Len=1448 TSval=
118	10.132460	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 109#4] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
119	10.132476	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 109#5] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
120	10.132539	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000 [ACK] Seq=82001 Ack=1 Win=5840 Len=1448 TSval=
121	11.103940	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=80553 Win=55024 Len=0 TSval=40,
125	11.103964	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413 [ACK] Seq=1 Ack=82001 Win=53576 Len=0 TSval=40,
122	11.104042	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000 [ACK] Seq=83449 Ack=1 Win=5840 Len=1448 TSval=
123	11.104075	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000 [ACK] Seq=84897 Ack=1 Win=5840 Len=1448 TSval=

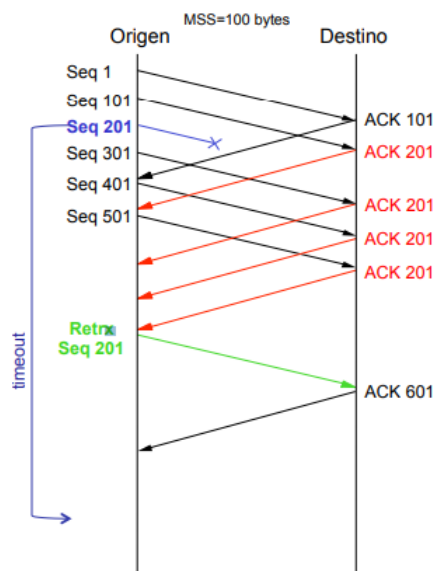
Tercer fast retransmit.

159	14.161665	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=113857	Ack=1	Win=5840	Len=1448	TSval=4	
160	14.161697	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=115305	Ack=1	Win=5840	Len=1448	TSval=4	
161	14.161728	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=116753	Ack=1	Win=5840	Len=1448	TSval=4	
162	15.132962	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413	[ACK]	Seq=1	Ack=110961	Win=63712	Len=0	TSval=4	
163	15.133034	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=118201	Ack=1	Win=5840	Len=1448	TSval=4	
164	15.133066	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=119649	Ack=1	Win=5840	Len=1448	TSval=4	
165	15.147809	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413	[ACK]	Seq=1	Ack=113857	Win=63712	Len=0	TSval=4	
166	15.147880	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=121097	Ack=1	Win=5840	Len=1448	TSval=4	
167	15.147912	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=122545	Ack=1	Win=5840	Len=1448	TSval=4	
168	15.178195	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 165#1]	8000 → 42413	[ACK]	Seq=1	Ack=113857	Win=63712	Len=0	TSval=4
170	15.178220	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 165#2]	8000 → 42413	[ACK]	Seq=1	Ack=113857	Win=63712	Len=0	TSval=4
169	15.178288	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[PSH, ACK]	Seq=123993	Ack=1	Win=5840	Len=1448	TSval=4	
171	15.178328	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=125441	Ack=1	Win=5840	Len=1448	TSval=4	
172	16.145086	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 165#3]	8000 → 42413	[ACK]	Seq=1	Ack=113857	Win=63712	Len=0	TSval=4
174	16.145109	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 165#4]	8000 → 42413	[ACK]	Seq=1	Ack=113857	Win=63712	Len=0	TSval=4
173	16.145181	11.0.0.2	12.0.0.2	TCP	1514	[TCP Fast Retransmission]	42413 → 8000	[ACK]	Seq=113857	Ack=1	Win=5840	Len=1448	TSval=4
175	16.175883	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 165#5]	8000 → 42413	[ACK]	Seq=1	Ack=113857	Win=63712	Len=0	TSval=4
177	16.175899	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 165#6]	8000 → 42413	[ACK]	Seq=1	Ack=113857	Win=63712	Len=0	TSval=4
176	16.175972	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=126889	Ack=1	Win=5840	Len=1448	TSval=4	
178	16.180202	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 165#7]	8000 → 42413	[ACK]	Seq=1	Ack=113857	Win=63712	Len=0	TSval=4
180	16.180216	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 165#8]	8000 → 42413	[ACK]	Seq=1	Ack=113857	Win=63712	Len=0	TSval=4
179	16.180280	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=128337	Ack=1	Win=5840	Len=1448	TSval=4	
181	17.149534	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413	[ACK]	Seq=1	Ack=126889	Win=53576	Len=0	TSval=4	

Cuarto fast retransmit.

202	19.206621	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=145713	Ack=1	Win=5840	Len=1448	TSval=4	
204	19.206658	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=147161	Ack=1	Win=5840	Len=1448	TSval=4	
205	20.192618	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413	[ACK]	Seq=1	Ack=142817	Win=63712	Len=0	TSval=4	
207	20.192642	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413	[ACK]	Seq=1	Ack=144265	Win=62264	Len=0	TSval=4	
206	20.192714	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=148609	Ack=1	Win=5840	Len=1448	TSval=4	
208	20.192755	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=150057	Ack=1	Win=5840	Len=1448	TSval=4	
209	20.222960	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413	[ACK]	Seq=1	Ack=145713	Win=63712	Len=0	TSval=4	
212	20.222984	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 209#1]	8000 → 42413	[ACK]	Seq=1	Ack=145713	Win=63712	Len=0	TSval=4
210	20.223053	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=151505	Ack=1	Win=5840	Len=1448	TSval=4	
211	20.223086	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=152953	Ack=1	Win=5840	Len=1448	TSval=4	
213	20.223125	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=154401	Ack=1	Win=5840	Len=1448	TSval=4	
214	21.194873	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 209#2]	8000 → 42413	[ACK]	Seq=1	Ack=145713	Win=63712	Len=0	TSval=4
216	21.194896	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 209#3]	8000 → 42413	[ACK]	Seq=1	Ack=145713	Win=63712	Len=0	TSval=4
215	21.194968	11.0.0.2	12.0.0.2	TCP	1514	[TCP Fast Retransmission]	42413 → 8000	[ACK]	Seq=145713	Ack=1	Win=5840	Len=1448	TSval=4
217	21.226291	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 209#4]	8000 → 42413	[ACK]	Seq=1	Ack=145713	Win=63712	Len=0	TSval=4
219	21.226313	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 209#5]	8000 → 42413	[ACK]	Seq=1	Ack=145713	Win=63712	Len=0	TSval=4
220	21.226333	12.0.0.2	11.0.0.2	TCP	66	[TCP Dup ACK 209#6]	8000 → 42413	[ACK]	Seq=1	Ack=145713	Win=63712	Len=0	TSval=4
218	21.226432	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[PSH, ACK]	Seq=155849	Ack=1	Win=5840	Len=1448	TSval=4	
221	21.226471	11.0.0.2	12.0.0.2	TCP	1514	42413 → 8000	[ACK]	Seq=157297	Ack=1	Win=5840	Len=1448	TSval=4	
222	22.207375	12.0.0.2	11.0.0.2	TCP	66	8000 → 42413	[ACK]	Seq=1	Ack=155849	Win=56472	Len=0	TSval=4	

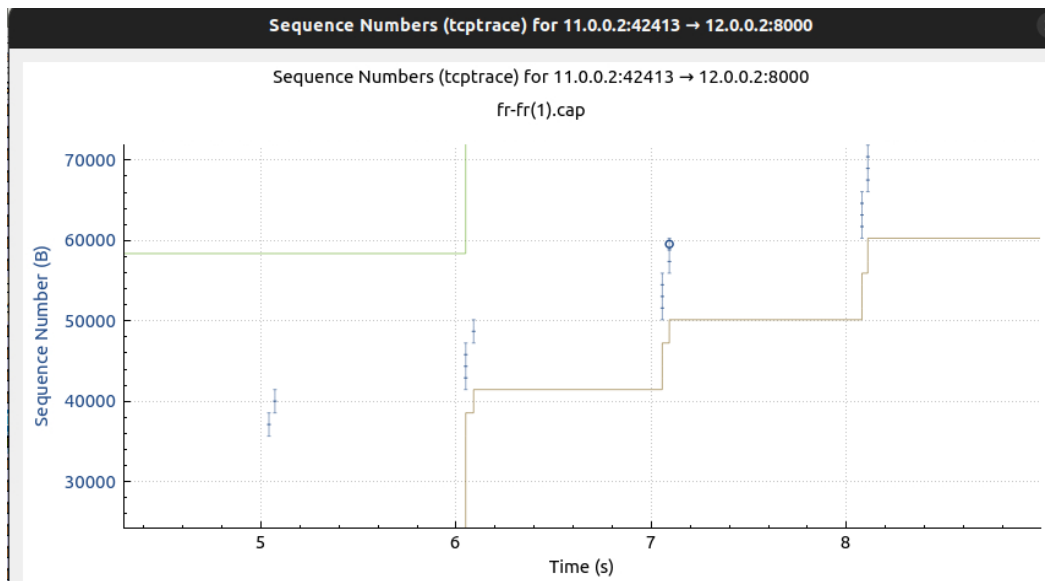
Imagen de la estructura basica del fast retransmit:



Con *Fast Retransmit* no se espera al *timeout* para retransmitir

El fast retransmit funciona de tal forma que, si se envía un paquete y no le ha llegado al servidor, este solicita un dup ack del que no ha llegado hasta que se realice el fast retransmit.

3. Observa la primera ocasión en la que se produce Fast Retransmit. ¿En qué modo de control de congestión se entra tras efectuarse dicha retransmisión?



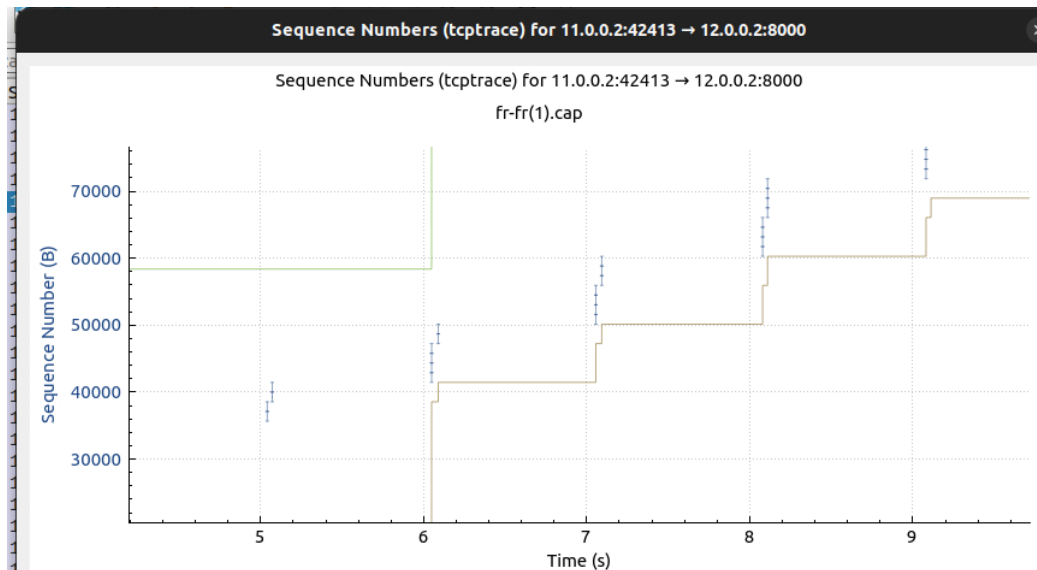
En la gráfica podemos ver como se sigue en modo CA

4. Poco después de producirse esa primera retransmisión debida a Fast Retransmit se envían nuevos segmentos de datos antes de recibir ningún ACK. ¿Cuántos nuevos segmentos de datos se envían? ¿Por qué se pueden enviar en ese instante? ¿Se podría enviar alguno más en ese periodo hasta que llegue el ACK del paquete retransmitido?

Se envían dos segmentos nuevos, se envían porque aún hay capacidad en el buffer y porque tienen mayor número de secuencia que el segmento del retrasmit. Si se pueden enviar más segmentos hasta que se llene la ventana o se llegue al tope en la ventana de congestión.

- Mejoras que aporta *fast retransmit*:
 - Se retransmite antes
 - Se evita que se produzca el *timeout*, y por tanto no se baja a $cwnd=1$.
 - Se aprovecha más la capacidad de la red: No se vacía la red, sino que se siguen enviando mensajes y recibiendo ACKs

5. Observa en qué momento llega el primer ACK nuevo. ¿En qué modo de control de congestión se entra tras recibirse dicho ACK nuevo? Indica cuántos paquetes se envían en ese momento, y explica la razón de este número. Observa la evolución de la ventana de congestión desde este momento hasta la siguiente retransmisión.



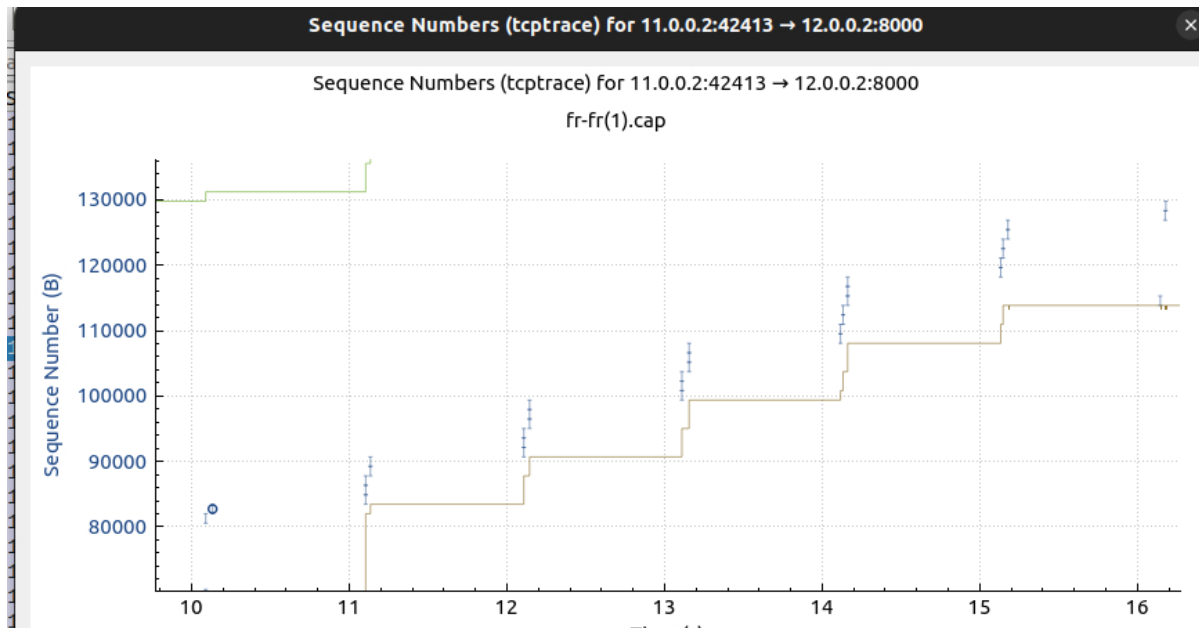
Se entra en modo CA, se van aumentando los ack uno a uno con cada retransmisión.

Se envían 4 paquetes antes del nuevo asentimiento que es el paquete 62.

6. Observa la segunda ocasión en la que se produce Fast Retransmit. Es en el paquete 114, aunque Wireshark no lo identifica correctamente. Estudia la evolución del control de congestión tras esta retransmisión.

101	8.109573	11.0.0.2	12.0.0.2	TCP	1514 42413 → 8000 [ACK] Seq=68969 Ack=1 Win=5840 Len=1448 TSval=
102	8.109604	11.0.0.2	12.0.0.2	TCP	1514 42413 → 8000 [ACK] Seq=70417 Ack=1 Win=5840 Len=1448 TSval=
103	9.081527	12.0.0.2	11.0.0.2	TCP	66 8000 → 42413 [ACK] Seq=1 Ack=63177 Win=62264 Len=0 TSval=40
106	9.081552	12.0.0.2	11.0.0.2	TCP	66 8000 → 42413 [ACK] Seq=1 Ack=66073 Win=63712 Len=0 TSval=40
104	9.081625	11.0.0.2	12.0.0.2	TCP	1514 42413 → 8000 [ACK] Seq=71865 Ack=1 Win=5840 Len=1448 TSval=
105	9.081658	11.0.0.2	12.0.0.2	TCP	1514 42413 → 8000 [ACK] Seq=73313 Ack=1 Win=5840 Len=1448 TSval=
107	9.081698	11.0.0.2	12.0.0.2	TCP	1514 42413 → 8000 [ACK] Seq=74761 Ack=1 Win=5840 Len=1448 TSval=
108	9.081729	11.0.0.2	12.0.0.2	TCP	1514 42413 → 8000 [ACK] Seq=76209 Ack=1 Win=5840 Len=1448 TSval=
109	9.111915	12.0.0.2	11.0.0.2	TCP	66 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=60816 Len=0 TSval=40
110	9.111982	11.0.0.2	12.0.0.2	TCP	1514 42413 → 8000 [ACK] Seq=77657 Ack=1 Win=5840 Len=1448 TSval=
111	9.112014	11.0.0.2	12.0.0.2	TCP	1514 42413 → 8000 [PSH, ACK] Seq=79105 Ack=1 Win=5840 Len=1448 T
112	10.088921	12.0.0.2	11.0.0.2	TCP	66 [TCP Window Update] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
113	10.088944	12.0.0.2	11.0.0.2	TCP	66 [TCP Dup ACK 109#1] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
115	10.088968	12.0.0.2	11.0.0.2	TCP	66 [TCP Dup ACK 109#2] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
116	10.088989	12.0.0.2	11.0.0.2	TCP	66 [TCP Dup ACK 109#3] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
114	10.089066	11.0.0.2	12.0.0.2	TCP	1514 [TCP Out-Of-Order] 42413 → 8000 [ACK] Seq=68969 Ack=1 Win=5
117	10.089138	11.0.0.2	12.0.0.2	TCP	1514 42413 → 8000 [ACK] Seq=80553 Ack=1 Win=5840 Len=1448 TSval=
118	10.132460	12.0.0.2	11.0.0.2	TCP	66 [TCP Dup ACK 109#4] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
119	10.132476	12.0.0.2	11.0.0.2	TCP	66 [TCP Dup ACK 109#5] 8000 → 42413 [ACK] Seq=1 Ack=68969 Win=
120	10.132539	11.0.0.2	12.0.0.2	TCP	1514 42413 → 8000 [ACK] Seq=82001 Ack=1 Win=5840 Len=1448 TSval=
121	11.103940	12.0.0.2	11.0.0.2	TCP	66 8000 → 42413 [ACK] Seq=1 Ack=80553 Win=55024 Len=0 TSval=40

Después de que se produzca este segundo retrasmit se continua de modo CA.

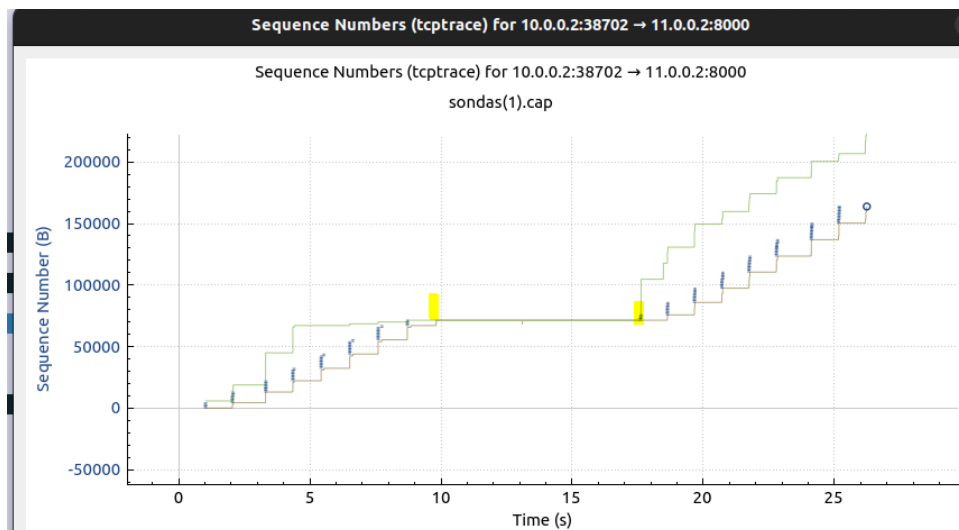


7. Observa las otras 2 retransmisiones, que también son Fast Retransmit. Para cada una de ellas, estudia la evolución del control de congestión después de dicha retransmisión.

En ambas se siguen con el mismo número de paquetes que existía cuando acabo el retrasmit.

5. Control de Congestión y sondas de ventana

1. Localiza en la traza los anuncios de ventana de tamaño 0 enviados por el servidor al cliente. ¿Qué segmentos son los que transportan estos anuncios?



Son los paquetes situados entre esos segundos. Que son los paquetes 80 y 82.

2. Observa las sondas de ventana que envía el cliente en ese periodo. ¿Qué segmentos son los que transportan las sondas de ventana?

El segmento 81,83 y 85. Después de este último el servidor envía un mensaje diciendo que se ha llenado la ventana.

3. Estudia el comportamiento de TCP entre los segmentos 43 y 87. Responde a las siguientes preguntas:

a) Indica cuál es el número de byte más alto que puede enviarse tras recibirse los siguientes segmentos: 41, 43, 51, 79.x

Tras recibirse el 41: Tenemos ocupado hasta el byte 30865 excluido por lo paquetes anteriores (30,32,34,36,38,40). Y como la ventana del byte 41 es de 67065 nos queda de hueco 36200 bytes.

Tras recibirse el 43: Ventana = 67065; Ocupado = 30865; Restante = 36200

Tras recibirse el 51: Ventana = 67065; Ocupado = 42449; Restante = 24616

Tras recibirse el 79: Ventana = 71409; Ocupado = ;71409 Restante = 0 (Si vemos la captura el segmento 78 pone tcp window full)

b) ¡Cuántos bytes nuevos pueden enviarse tras recibirse el segmento 80?

No se puede recibir nada porque la ventana sigue llena, nos lo avisa con un mensaje de tcp zerowindow.

c) ¿Cuántos bytes de datos transportan los segmentos con sondas de ventana?

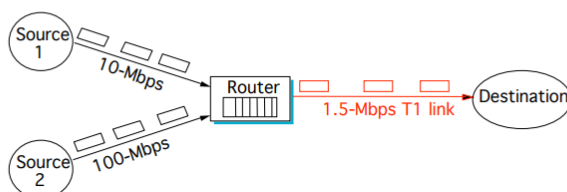
Pues transportaran el tope que podia guardar la ventana, que son 71409 bytes. Este último no. Es decir. Hasta el 71408.

d) Tras haberse cerrado la ventana, ¿En qué segmento vuelve a abrirse? ¿Cuál es el tamaño de la nueva ventana anunciada?

Se abre en el segmento 84. El tamaño de la nueva ventana anunciada será de 74305.

e) Tras recibirse el segmento 84, ¿qué limita al emisor, la ventana de control de flujo o la de congestión?

- **Control de flujo:** Que el receptor no sea desbordado por un emisor que transmite más rápido de lo que él es capaz de procesar.
 - Extremo a extremo.
 - El receptor limita la velocidad de transferencia del emisor debido fundamentalmente al tamaño del *buffer* de recepción y a la velocidad con la que lee la aplicación receptora
- **Control de congestión:** Que un router intermedio no sea desbordado por un emisor que transmite más rápido de lo que él es capaz de procesar.
 - Involucra a extremos y a la red (encaminadores, enlaces)
 - La red limita la velocidad de transferencia del emisor



En este caso no tenemos router intermedio, así que lo que se limita es la ventana del control de flujo.

4. Indica en qué modo de control de congestión se encuentra la conexión TCP en los siguientes puntos:

a) Antes del segmento 43.

Se encuentra en congestion avoidance (CA)

b) Entre el segmento 43 y 87

CA desde el 43 hasta el 78

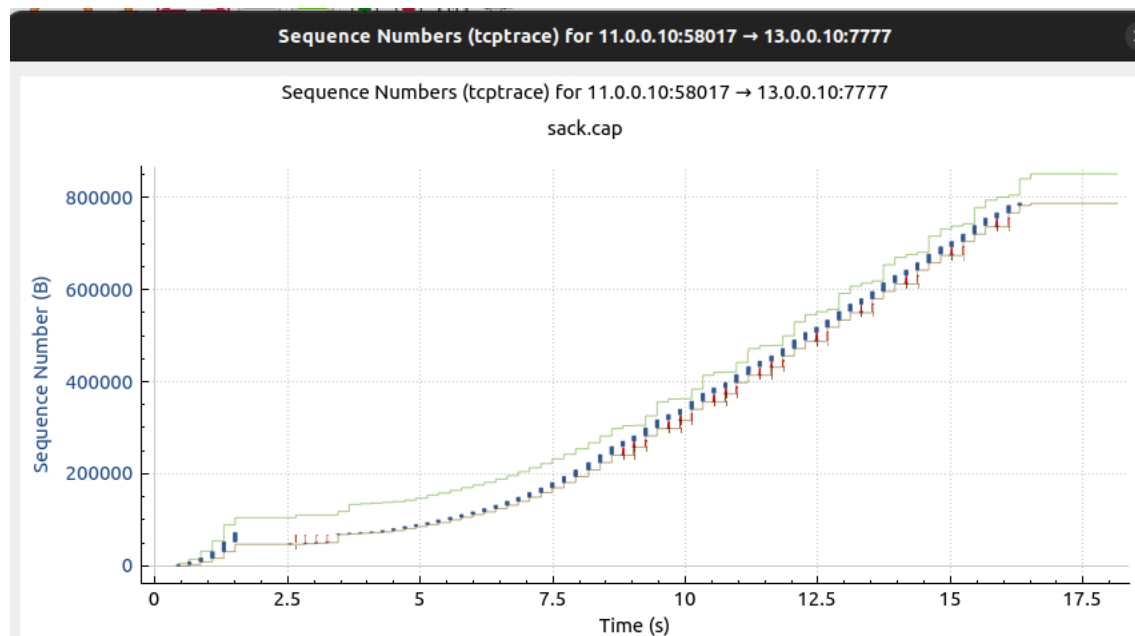
c) Después del segmento 87

Hay un SS y posteriormente se sigue con CA

5. Indica en qué momentos de la conexión es la ventana de control de flujo la que limita al emisor, y en cuáles es la ventana de control de congestión

La limita desde el fragmento 78 hasta el 85.

6. ACKs selectivos (SACK)



1. Dado que SACK es una opción que pueden usar ciertas implementaciones de TCP, indica cómo indica un extremo de la conexión que acepta asentimientos selectivos del otro extremo. ¿En qué segmentos de la conexión ocurre ese acuerdo? ¿Qué número de opción de TCP se utiliza?

Para indicar que acepta asentimientos selectivos del otro extremo, un extremo de la conexión establece el bit SACK en el campo de opciones del encabezado TCP. Este bit se establece en el segmento SYN durante el proceso de establecimiento de conexión. Si uno de los dos no admite SACK, no se podrá usar en la conexión. Los segmentos con sack permited son el 1 y el 2.

Usando `tcp.options.sack_perm`

Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	11.0.0.10	13.0.0.10	TCP	74	58017 → 7777 [SYN] Seq=0 Win=5840 Len=0
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) Ethernet II, Src: 42:ad:67:6e:ee:a1 (42:ad:67:6e:ee:a1), Dst: 9e:b6:26:b6:81:36 (9e:b6:26:b6:81:36) Internet Protocol Version 4, Src: 11.0.0.10, Dst: 13.0.0.10 Transmission Control Protocol, Src Port: 58017, Dst Port: 7777, Seq: 0, Len: 0 Source Port: 58017 Destination Port: 7777 [Stream index: 0] [Conversation completeness: Complete, WITH_DATA (31)] [TCP Segment Len: 0] Sequence Number: 0 (relative sequence number) Sequence Number (raw): 347632640 [Next Sequence Number: 1 (relative sequence number)] Acknowledgment Number: 0 Acknowledgment number (raw): 0 1010 = Header Length: 40 bytes (10) Flags: 0x002 (SYN) Window: 5840 [Calculated window size: 5840] Checksum: 0x8697 [unverified] [Checksum Status: Unverified] Urgent Pointer: 0 Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale [Timestamps]						
2	0.432924	13.0.0.10	11.0.0.10	TCP	74	7777 → 58017 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) Ethernet II, Src: 9e:b6:26:b6:81:36 (9e:b6:26:b6:81:36), Dst: 42:ad:67:6e:ee:a1 (42:ad:67:6e:ee:a1) Internet Protocol Version 4, Src: 13.0.0.10, Dst: 11.0.0.10 Transmission Control Protocol, Src Port: 7777, Dst Port: 58017, Seq: 0, Ack: 1, Len: 0 Source Port: 7777 Destination Port: 58017 [Stream index: 0] [Conversation completeness: Complete, WITH_DATA (31)] [TCP Segment Len: 0] Sequence Number: 0 (relative sequence number) Sequence Number (raw): 352882514 [Next Sequence Number: 1 (relative sequence number)] Acknowledgment Number: 1 (relative ack number) Acknowledgment number (raw): 347632641 1010 = Header Length: 40 bytes (10) Flags: 0x012 (SYN, ACK) Window: 5792 [Calculated window size: 5792] Checksum: 0xdd27 [unverified] [Checksum Status: Unverified] Urgent Pointer: 0 Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale [Timestamps]						

2. Indica cuál es el primer segmento de la captura donde se muestran asentimientos selectivos.
 ¿En qué lugar del segmento va esta información? ¿Qué número de opción se utiliza? ¿Qué números de secuencia se están asintiendo y qué números de secuencia se asienten selectamente?
 ¿Qué números de segmento son los que se están asintiendo, y cuáles se asienten selectivamente?

El primer segmento donde se muestran asentimientos selectivos es el paquete 78.

TCP Option - SACK 52049-65081	
Kind:	SACK (5)
Length:	10
left edge =	52049 (relative)
right edge =	65081 (relative)
[TCP SACK Count:	1]

Numero de opción: 52049-65081.

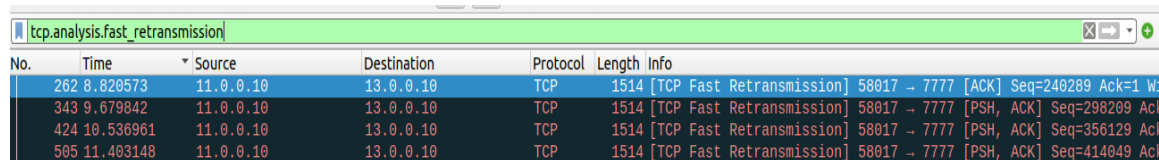
Asiente desde el 52049-65080 (Uno menos siempre)

Asiente desde el segmento 62 hasta el 73 (excluido), así que podríamos decir que del 62-72.

3. Explica si el segmento 78 es una retransmisión por timeout o es una retransmisión rápida

El paquete 78 lleva activado el campo de tcp retrasmision, que no es lo mismo que fast retrasmisiones.

Si filtramos en el buscador (**Tcp.analysis.fast_retrasmision**) nos aparecen las retrasmisiones rápidas, y el paquete 78 no aparece.



No.	Time	Source	Destination	Protocol	Length	Info
262	8.829573	11.0.0.10	13.0.0.10	TCP	1514	[TCP Fast Retransmission] 58017 → 7777 [ACK] Seq=240289 Ack=1 W
343	9.679842	11.0.0.10	13.0.0.10	TCP	1514	[TCP Fast Retransmission] 58017 → 7777 [PSH, ACK] Seq=298209 Ac
424	10.536961	11.0.0.10	13.0.0.10	TCP	1514	[TCP Fast Retransmission] 58017 → 7777 [PSH, ACK] Seq=356129 Ac
505	11.403148	11.0.0.10	13.0.0.10	TCP	1514	[TCP Fast Retransmission] 58017 → 7777 [PSH, ACK] Seq=414049 Ac

4. Explica en qué modo de control de congestión está la máquina 11.0.0.10 justo después de haber enviado el segmento 78. Indica el valor de threshold en ese instante

Se encuentra en control de flujo ya que la ventana se ha llenado y el servidor no quiere ser desbordado por el emisor que envía mensajes más rápido de lo que el servidor es capaz de procesar.

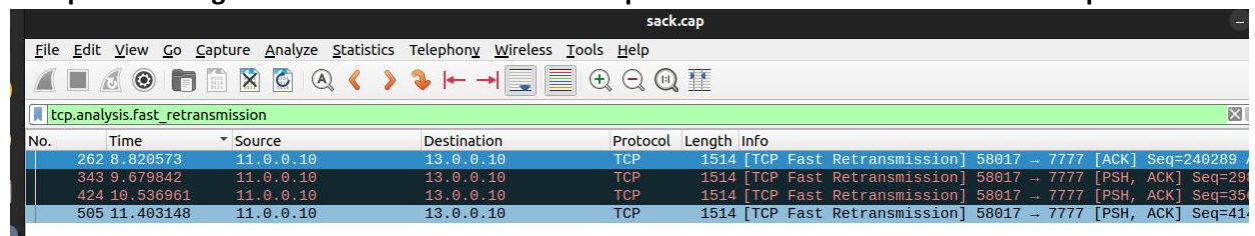
5. Justo antes de enviar el segmento 83, ¿qué segmentos puede suponer la máquina 11.0.0.10 que le faltan a 13.0.0.10?

El paquete 82 asiente del 58 para abajo (58 excluido). Por lo que todos los enviados por el cliente por encima del 58 incluido no están asentidos y están ocupando el buffer, además el paquete con seq 46793 no ha llegado y vemos con en los fragmentos 80 y 81 el servidor está pidiendo una nueva retransmisión de este paquete.

6. Si 11.0.0.10 sabe que a 13.0.0.10 le faltan varios segmentos, ¿por qué justo después de enviar el segmento 83 no retransmite todos los que sabe que le faltan?

Le faltaba el paquete 46793, que lo retransmite en el segmento 78, así ya a partir del 83 empieza a retransmitir los que les faltan.

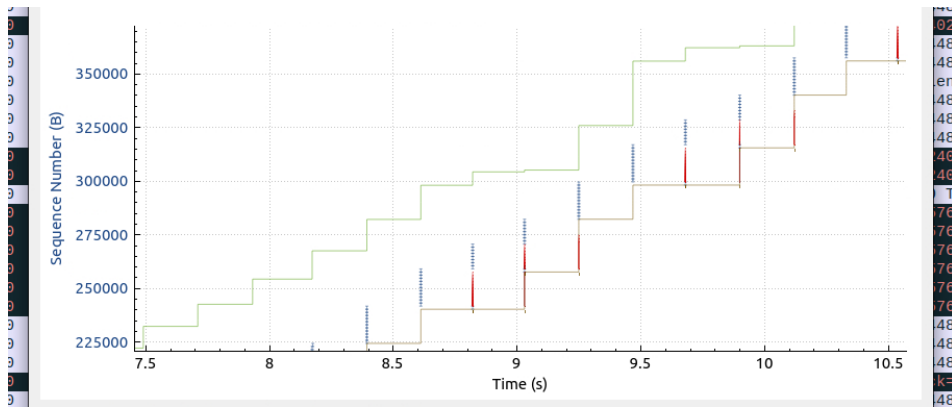
7. Explica si el segmento 262 es una retransmisión por timeout o es una retransmisión rápida.



No.	Time	Source	Destination	Protocol	Length	Info
262	8.829573	11.0.0.10	13.0.0.10	TCP	1514	[TCP Fast Retransmission] 58017 → 7777 [ACK] Seq=240289
343	9.679842	11.0.0.10	13.0.0.10	TCP	1514	[TCP Fast Retransmission] 58017 → 7777 [PSH, ACK] Seq=29
424	10.536961	11.0.0.10	13.0.0.10	TCP	1514	[TCP Fast Retransmission] 58017 → 7777 [PSH, ACK] Seq=35
505	11.403148	11.0.0.10	13.0.0.10	TCP	1514	[TCP Fast Retransmission] 58017 → 7777 [PSH, ACK] Seq=41

Se trata de una retransmisión rápida, ya que dispone del mismo número de seq que el paquete 242. Además, si filtramos por fast.retrasmision vemos como aparece el paquete 262.

8. Explica en qué modo de control de congestión está la máquina 11.0.0.10 justo después de haber enviado el segmento 262. Indica el valor de threshold en ese instante.



Podemos ver que se sigue en modo de Congestion Avoidance, pero el modo de congestion es una congestion de flujo.

9. Al recibir el segmento 257, ¿qué segmento/s puede suponer la máquina 11.0.0.10 que le falta/n a 13.0.0.10?

En ese momento no sabe si le falta alguno ya que no ha enviado ningún mensaje. El único mensaje que manda el fragmento 257 es que la ventana está llena. Pero luego podemos ver en el siguiente paquete (259) que le falta el fragmento con seq = 240289.

10. A recibir el segmento 259, ¿qué segmento/s puede suponer la máquina 11.0.0.10 que le falta/n a 13.0.0.10?

El segmento con numero de seq 240289.

11. Fíjate en las diferencias que hay entre el segmento 257 y 259. ¿Qué crees que ha provocado que la máquina 13.0.0.10 haya enviado ese asentimiento?

El segmento 257 está anunciando que se ha llenado la ventana mientras que el segmento 259 anuncia no ha llegado el paquete con seq 240289. Yo creo que dicho paquete ha llegado justo al límite de tiempo que hay para que se retransmita el mensaje de no recibido, y como la maquina ya tenía el paquete creado no ha dado tiempo a cancelarlo, ya que si vemos la diferencia de tiempo entre los dos paquetes es de 0.000002 seg. Pero no creo que sea así porque más adelante siguen produciéndose solicitudes del mismo paquete.

12. A recibir el segmento 277, ¿qué segmento/s puede suponer la máquina 11.0.0.10 que le falta/n a 13.0.0.10?

El mismo que anteriormente, el paquete con seq 240289. Ya que los paquetes posteriores no lo han asentido y se consideran que siguen en vuelo.

13. Al recibir el segmento 669, ¿qué segmento/s puede suponer la máquina 11.0.0.10 que le falta/n a 13.0.0.10?

Está ocurriendo lo mismo que los paquetes anteriores 257 y 259.

14. Explica si el segmento 747 es una retransmisión por timeout o es una retransmisión rápida.

Es una retransmisión por time out, ya que no hay al menos 3 ack duplicados por encima con el mismo número de secuencia, condición necesaria para que sea transmisión rápida. Además, si filtras las retransmisiones rápidas, este fragmento no aparece.

15. ¿Por qué hay 2 retransmisiones juntas (segmentos 747 y 749)?

El primer paquete (747) es una retransmisión es del paquete con seq 612121 y es una retransmisión rápida, mientras que el paquete 749 es una retransmisión del paquete 613569 y es una retransmisión por timeout. Por lo que al ser diferentes paquetes y diferentes retransmisiones esto puede ocurrir.