Nivel de Transporte

Resolución de ejercicios vistos en clase

Lucio Santi

10.05.2017

1. Primer ejercicio

1.1. Enunciado

Dada la siguiente secuencia de segmentos TCP, responder las preguntas que siguen:

#	ORIG	DEST	FLAGS	#SEQ	#ACK	LENGTH
1	1.2.3.4:5678	20.232.1.1:80	S	0	—	0
2	1.1.1.1:2222	3.3.3.3:4444	S	42	—	0
3	20.232.1.1:80	1.2.3.4:5678	SA	10000	1	0
4	1.2.3.4:5678	20.232.1.1:80	A	1	10001	0
5	3.3.3.3:4444	1.1.1.1:2222	SA	54321	43	0
6	1.2.3.4:5678	20.232.1.1:80	A	1	10001	50
7	1.1.1.1:2222	3.3.3.3:4444	A	43	54322	0
8	20.232.1.1:80	1.2.3.4:5678	A	10001	51	0
9	20.232.1.1:80	1.2.3.4:5678	A	10001	51	200
10	1.1.1.1:2222	3.3.3.3:4444	SAU	43	64334	0
11	20.232.1.1:80	1.2.3.4:5678	F	10201	—	0
12	1.2.3.4:5678	20.232.1.1:80	FA	51	10202	0
13	3.3.3.3:4444	1.1.1.1:2222	RA	54321	43	0
14	20.232.1.1:80	1.2.3.4:5678	A	10202	52	0

a. Asociar cada segmento con cada una de las conexiones indicando el criterio usado para determinar a qué conexión pertenece un segmento. ¿Cuántas son?

b. Detectar el cierre anómalo de una de las conexiones e indicar una posible causa.

c. Para la otra conexión, detallar los cambios de estado en cada extremo a lo largo de toda la comunicación.

1.2. Resolución

Para responder cuántas conexiones TCP hay, tal vez sea conveniente mirar primero la segunda pregunta del ejercicio. En general, al momento de determinar la conexión asociada a un paquete dado, hay que mirar cuatro componentes: dirección IP origen, puerto origen, dirección IP destino y puerto destino. Con esta tupla de valores podemos caracterizar el flujo al que pertenece el paquete. En este caso, como los orígenes son distintos, basta con mirar sólo la IP origen. Dicho esto, podemos concluir que en la captura aparecen dos conexiones: (`1.2.3.4:5678`, `20.232.1.1:80`) y

(1.1.1.1:2222, 3.3.3.3:4444).

Respecto al cierre anómalo, puede verse que éste ocurre en el paquete 13, cuando el proceso en 3.3.3.3:4444 envía un RST. Esto se debe al paquete con flags SYN/ACK/URG recibido en la línea 10: recibir un SYN en una conexión sincronizada tiene como efecto el envío de un RST, conforme a la especificación de TCP (para más detalles, referirse a la página 71 del RFC 793). La consecuencia de este envío es que el receptor procederá a liberar los recursos de la conexión de inmediato, sin ejercer el algoritmo de cierre tradicional. A su vez, el host emisor del RST hace esto mismo en reacción al paquete mal formado que recibió.

Veamos finalmente cómo responder la última pregunta. Para esto, separemos primero los paquetes de la conexión:

#	ORIG	DEST	FLAGS	#SEQ	#ACK	LENGTH
1	1.2.3.4:5678	20.232.1.1:80	S	0	—	0
3	20.232.1.1:80	1.2.3.4:5678	SA	10000	1	0
4	1.2.3.4:5678	20.232.1.1:80	A	1	10001	0
6	1.2.3.4:5678	20.232.1.1:80	A	1	10001	50
8	20.232.1.1:80	1.2.3.4:5678	A	10001	51	0
9	20.232.1.1:80	1.2.3.4:5678	A	10001	51	200
11	20.232.1.1:80	1.2.3.4:5678	F	10201	—	0
12	1.2.3.4:5678	20.232.1.1:80	FA	51	10202	0
14	20.232.1.1:80	1.2.3.4:5678	A	10202	52	0

Ahora, analicemos paquete por paquete qué está sucediendo en cada extremo:

- Del paquete 1 se desprende que 1.2.3.4:5678 está haciendo una apertura activa. Por ende, este paquete hace que su TCP se mueva de CLOSED a SYN-SENT.
- El paquete 3 es una respuesta positiva del paquete 1 por parte del proceso 20.232.1.1:80. Para que esto ocurra, su TCP debe haber estado en LISTEN antes de poder contestar. Al recibir el SYN, la reacción consiste en trasladarse a SYN-RCVD y emitir un paquete SYN+ACK.
- El primer host recibe luego este segmento y esto hace que se mueva a ESTABLISHED, enviando como respuesta el ACK final del three-way handshake (paquete 4).
- Una vez procesado esto, el TCP de 20.232.1.1:80 se moverá también a ESTABLISHED.
- Luego tenemos que los paquetes 6, 8 y 9 corresponden a envío y reconocimiento de datos, por lo que no motivan transiciones entre estados.
- En la porción final de la captura, vemos que se inicia el proceso de cierre de conexión. El segmento 11 nos muestra que 20.232.1.1:80 está haciendo un cierre activo, revirtiendo así su rol "pasivo" al momento de establecer la conexión. El efecto de este envío es transicionar al estado FIN-WAIT-1.
- En la siguiente línea vemos que su interlocutor recibe este mensaje y reacciona respondiendo un paquete FIN+ACK que no sólo reconoce el primer FIN (puesto que su #ACK es 10202) sino que además inicia el cierre de su stream de escritura. En consecuencia, su TCP colapsa dos estados, pasando de ESTABLISHED a LAST-ACK.

- El segmento 14 reconoce correctamente el `FIN` de `1.2.3.4:5678`. Al recibirlo, el TCP de dicho proceso transicionará a CLOSED, mientras que el de `20.232.1.1:80` se moverá a TIME-WAIT como respuesta al segmento 12. Asumiendo que el último `ACK` no se extravía en la red, allí permanecerá por 120 segundos (i.e., 2 MSLs) antes de pasar también a CLOSED.

2. Segundo ejercicio

2.1. Enunciado

La siguiente captura de paquetes TCP corresponde a un programa corriendo en un host A comunicándose con el servicio corriendo en el puerto 5900 del host B:

```
No.  Source  Dest  Info
1    A       B     26574 >5900 [SYN] Seq=0 Ack=0 Len=0
2    B       A     5900 >25674 [SYN,ACK] Seq=0 Ack=1 Len=0
3    A       B     26574 >5900 [ACK] Seq=1 Ack=1 Len=0
4    A       B     26574 >5900 [PSH,ACK] Seq=1 Ack=1 Len=1024
5    B       A     5900 >25674 [ACK] Seq=1 Ack=1025 Len=200
6    A       B     26574 >5900 [ACK] Seq=1025 Ack=201 Len=0
```

a. Extender la captura proponiendo una serie de paquetes que hagan que el socket del host A atraviese los siguientes estados:

$$\text{ESTABLISHED} \rightarrow \text{FIN-WAIT-1} \rightarrow \text{FIN-WAIT-2} \rightarrow \text{TIME-WAIT} \rightarrow \text{CLOSED}$$

b. Proponer otros dos escenarios de cierre que podrían darse en esta conexión mencionando los estados transitados por cada socket.

2.2. Resolución

Antes de mostrar los segmentos que satisfagan lo pedido, observemos lo siguiente:

- Para que el host `A` pase de ESTABLISHED a FIN-WAIT-1, debe ocurrir que tome una postura activa y decida cerrar la conexión primero.
- Si el siguiente estado debe ser FIN-WAIT-2, necesitamos que el host `B` responda simplemente con `ACK` al `FIN` original.
- Luego, al cabo de una cantidad arbitraria de paquetes de datos por parte de `B`, debe darse la transición a TIME-WAIT de `A`. Esto lo lograremos haciendo que `B` envíe su `FIN`.
- Naturalmente, `A` debe reconocer correctamente este `FIN`, lo cual motivará su paso a TIME-WAIT.
- La transición final a CLOSED se va a dar implícitamente; no debemos agregar ningún otro paquete para alcanzar este objetivo.

Con esto en mente, una posible solución podría ser la que se detalla a continuación. Notar que la respuesta no es única dado que `B` puede enviar tantos datos como se quiera. Prestar también especial atención a los números de secuencia y de reconocimiento utilizados, recordando que el flag de `FIN` es secuenciable.

```
No.  Source  Dest  Info
7    A       B     26574 >5900 [FIN,ACK] Seq=1025 Ack=201 Len=0
8    B       A     5900 >25674 [ACK] Seq=201 Ack=1026 Len=0
9    B       A     5900 >26574 [FIN,ACK] Seq=201 Ack=1026 Len=0
10   A       B     26574 >5900 [ACK] Seq=1026 Ack=202 Len=0
```

En este escenario, como dijimos, el host `A` es el que inicia activamente el cierre de conexión. Si invertimos este comportamiento, haciendo que sea `B` el que lo inicie, obtendremos otro escenario de cierre válido. También podría ocurrir que ambos hosts decidan cerrar la conexión simultáneamente, lo cual daría origen a otro escenario de cierre distinto. Este último caso tiene la particularidad de que ambos hosts transitarán por los mismos estados: de ESTABLISHED saltarán a FIN-WAIT-1 al enviar el `FIN`, para luego trasladarse a CLOSING al recibir el `FIN` de la contraparte. La respuesta de sendos `FIN`es hará que, finalmente, los interlocutores pasen a TIME-WAIT y, al cabo de 2 MSLs, a CLOSED.