Universidad Nacional Autónoma de México
Facultad de Ciencias

Redes de computadoras

Números de secuencia en TCP

Paulo Contreras Flores
paulo.contreras.flores@ciencias.unam.mx

Cada capa ya sea del modelo OSI o del modelo TCP/IP, está conformada por datos que tienen un significado para esa capa es decir, que con esos datos la capa lleva a cabo su función. Cada conjunto de estos datos tiene un nombre dependiendo de la capa en la que se encuentre, en la siguiente figura 1 se muestran dichos nombres.

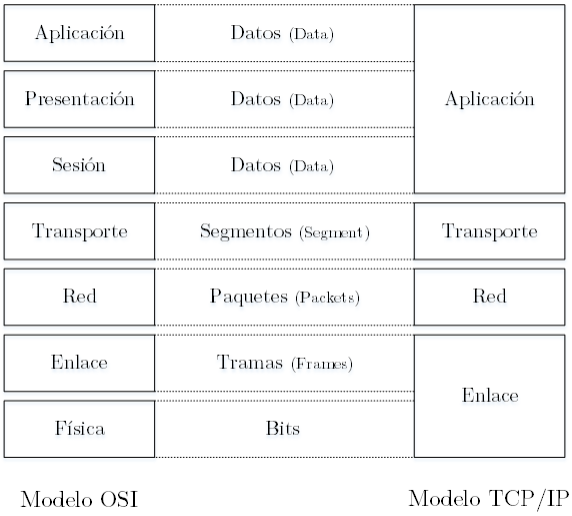

Figura 1: Nombre para cada tipo de datos en cada capa del modelo OSI y TCP/IP

Ray Tomlinson introdujo en 1975 el concepto del three-way handshake, el cual es un protocolo de establecimiento de comunicación para la transmisión de datos. Este proceso se realiza generalmente antes de que cualquier dato sea enviado entre los equipos. Lo que se representa, es un cliente o equipo origen iniciando una conexión al servidor o equipo destino. Debido a que es una conexión TCP (capa de transporte), un puerto o servicio debe identificar al servicio al cual se desea conectar.

Cada equipo de la sesión seleccionará un número de secuencia inicial (ISN por sus siglas en inglés) como el primer número de secuencia. Esto significa que el cliente y el servidor seleccionarán diferentes números de secuencia individuales. ¿Cómo son generados estos números de secuencia?

Depende de la implementación de la pila TCP/IP de cada sistema operativo. Claro que es mejor que estos números sean generados de manera aleatoria, de manera que no puedan ser adivinados y, con esto, la seguridad de la comunicación podría verse afectada.

Estos números de secuencia proporcionan el mecanismo de ordenamiento de segmentos para el flujo de datos TCP. Desde que el flujo es iniciado con un número de secuencia inicial, un segmento perdido es fácil de identificar. También, si un segmento TCP llega al equipo destino en un orden diferente al que fue enviado, el equipo destino puede ordenarlo de la forma correcta a través del número de secuencia.

El cliente envía un segmento con la bandera **SYN** (Synchronize) estableciendo la solicitud para una conexión TCP al servidor con un número de secuencia $seq_{client} = x$.

Después, si el servidor está operando y ofrece el servicio deseado, además de que pueda aceptar conexiones entrantes, responde a esta solicitud de conexión con la bandera **SYN** establecida y con un propio número de secuencia $seq_{server} = y$ al cliente, y reconoce la solicitud de conexión del cliente con la bandera **ACK** (acknowledgement) y el número de confirmación que corresponde al número de secuencia recibido más uno, $ack_{server} = x+1$, todo en un sólo segmento. En este punto, el segundo paso de los tres es completado.

Finalmente, si el cliente recibe el **SYN** y **ACK** del servidor y todavía necesita continuar con la conexión, envía un segmento final con la bandera **ACK** activada, el número de secuencia $seq_{client} = x + 1$ y el número de confirmación $ack_{client} = y + 1$ al servidor reconociendo que ha recibido el **SYN/ACK** del servidor. Una vez que el servidor recibe y acepta el **ACK**, la conexión se establece. Los datos pueden ser intercambiados entre los dos equipos. Este proceso está representado en la figura 2, y en la figura 3 se muestra el three-way handshake usando Wireshark.

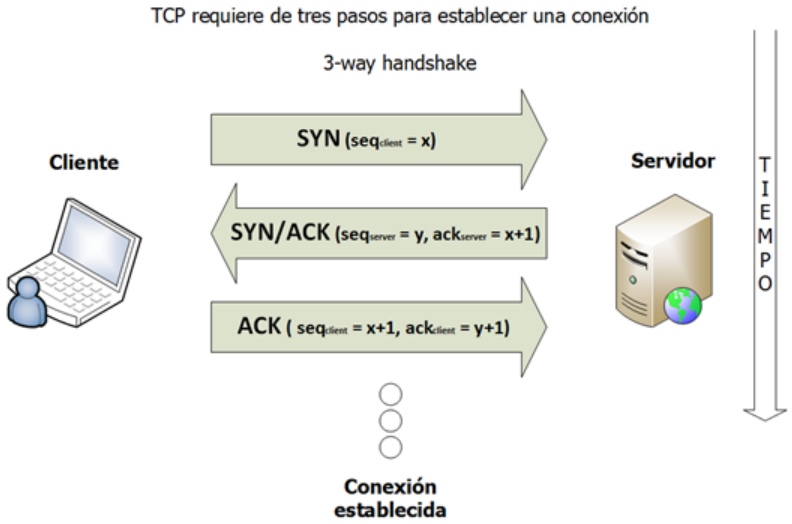

Figura 2: three-way handshake

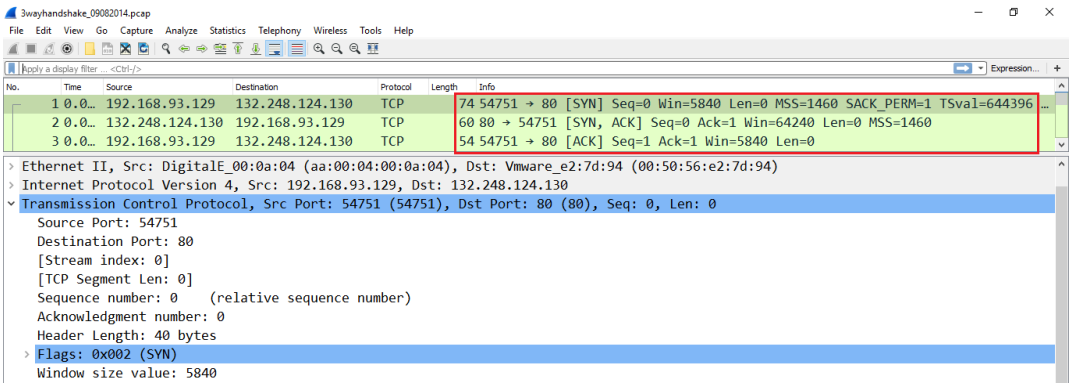

Figura 3: three-way handshake en Wireshark

En este punto de la comunicación ya se ha establecido el inicio de sesión y comienza a haber intercambio de información. El primer segmento después del three-way handshake es enviado por el equipo cliente con las banderas PUSH y ACK activadas. El número de secuencia es 1 (número de secuencia relativo) ya que ningún dato ha sido transmitido desde el último paquete. El número de confirmación también es 1 (número de confirmación relativo) porque ningún dato ha sido recibido desde el servidor. Este segmento enviado por el cliente contiene datos de la capa de aplicación, específicamente es una petición de HTTP. La longitud de los datos del segmento es de 219 bytes sin contar cabeceras de IP ni TCP, figura 4.

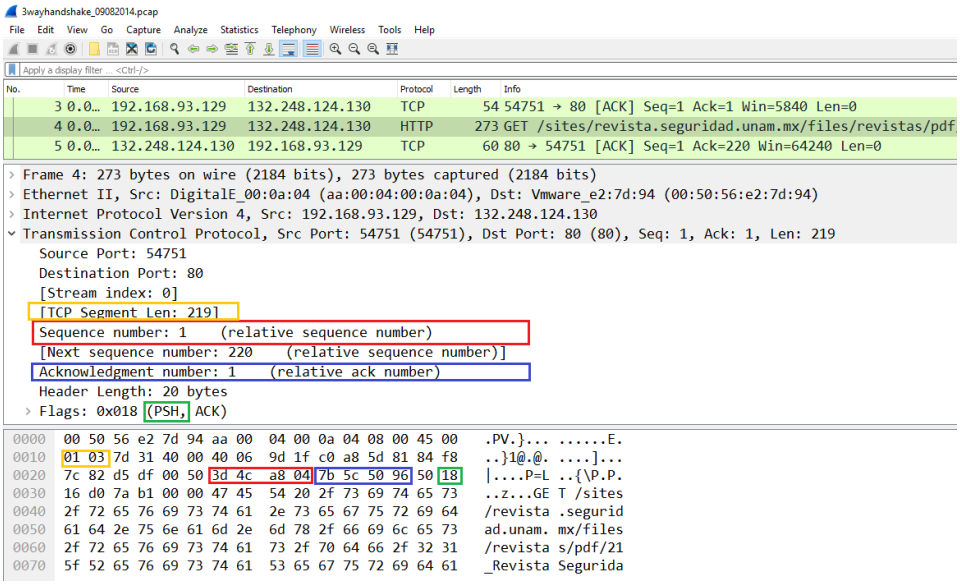

Figura 4: Petición de HTTP

PUSH/ACK: $seq_{client} = 0x3d4ca804$ $ack_{client} = 0x7b5c5096$
Total length: 259 bytes = 0x0103
20 bytes IP + 20 bytes TCP + 219 bytes Datos

Note que Wireshark ha mostrado el número de secuencia siguiente (o *Next sequence number*) como 220, que corresponde al número de secuencia relativo, es decir al 1, más el número de bytes de datos transmitidos en el segmento (219 bytes), este número será el número de confirmación del segmento esperado que enviará el servidor.

Este segmento de la figura 5 es enviado por el servidor exclusivamente para confirmar la recepción de los datos enviados por el cliente en el segmente anterior, mientras que la capa de aplicación procesa la petición de HTTP. Note que el número de confirmación se ha incrementado en 219 bytes, que corresponde a la longitud de datos del segmento anterior enviado por el cliente. El número de secuencia del servidor se mantiene en el mismo valor.

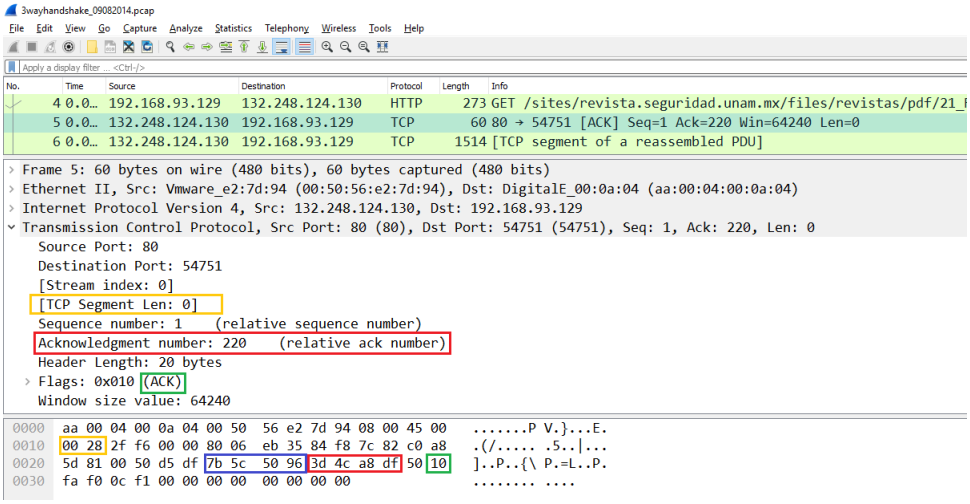

Figura 5: Confirmación de TCP al segmento anterior

ACK: $seq_{server} = 0x7b5c5096$ $ack_{server} = 0x3d4ca804 + 0xdb = 0x3d4ca8df$

Total length: 40 bytes = 0x0028

20 bytes IP + 20 bytes TCP

El segmento de la figura 6 marca el inicio de la respuesta HTTP del servidor. Su número de secuencia se sigue manteniendo, ya que ninguno de sus segmentos anteriores a éste contienen datos. Este segmento contiene datos por 1460 bytes.

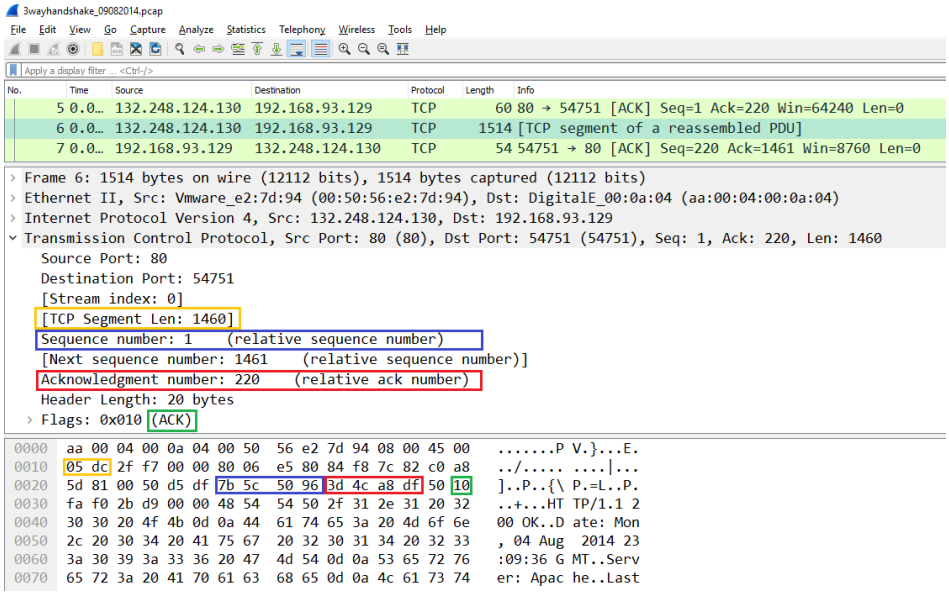

Figura 6: Segmento TCP con la respuesta de HTTP

ACK: $seq_{server} = 0x7b5c5096$ $ack_{server} = 0x3d4ca8df$
Total length: 1500 bytes = 0x05dc
20 bytes IP + 20 bytes TCP + 1460 bytes Datos

El segmento de la figura 7 contiene la confirmación del cliente a la respuesta HTTP del servidor. El número de secuencia del cliente se ha incrementado en 219 porque es el tamaño de los datos del último segmento que envió. Porque recibió 1460 bytes de datos del servidor, el cliente incrementa su número de confirmación en 1460.

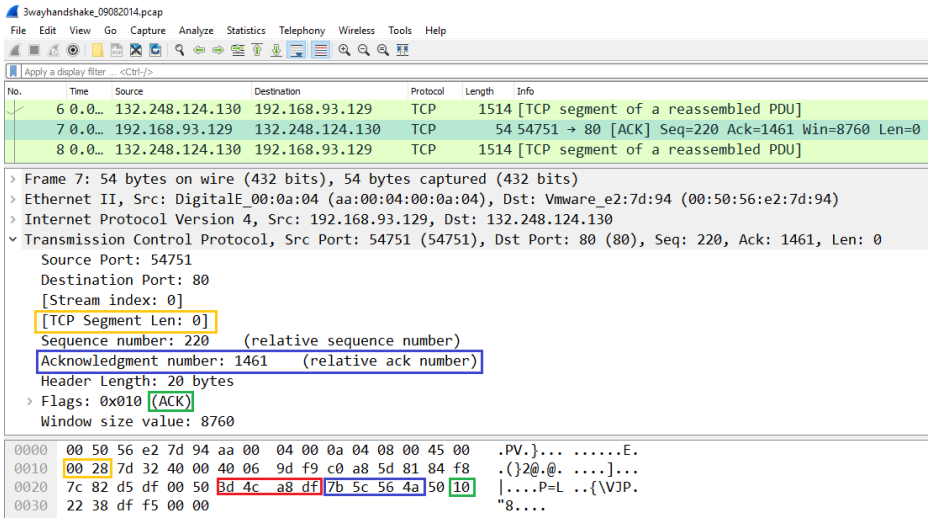

Figura 7: Confirmación de TCP al segmento anterior

ACK: $seq_{client} = 0x3d4ca8df$ $ack_{client} = 0x7b5c5096 + 0x5b4 = 0x7b5c564a$
Total length: 40 bytes = 0x0028
20 bytes IP + 20 bytes TCP

Para la mayoría de los siguientes segmentos del cliente el comportamiento será el mismo, el número de secuencia será constante en 0x3d4ca8df, porque no se trasmitieron datos más allá de los 219 bytes de la petición HTTP. En contraste, los números de secuencia del servidor continuarán incrementándose tanto como segmentos se envíen de la respuesta de HTTP.

Después del último segmento de la comunicación con el servidor, el cliente procesa la respuesta HTTP como un todo y decide que la comunicación ya no es necesaria. El cliente envía un paquete con la bandera de FYN activada. Su número de confirmación sigue siendo el mismo que el del segmento anterior, figura 8.

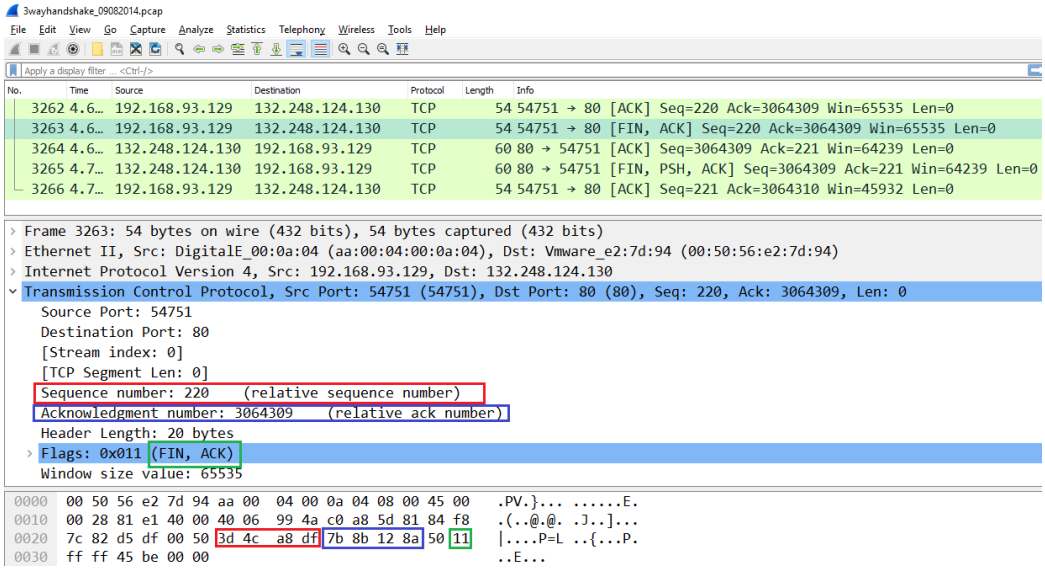

Figura 8: Fin de la conexión por parte del cliente

El servidor confirma al cliente el cierre de la conexión al incrementar el número de confirmación en 1 y coloca la bandera de FIN.

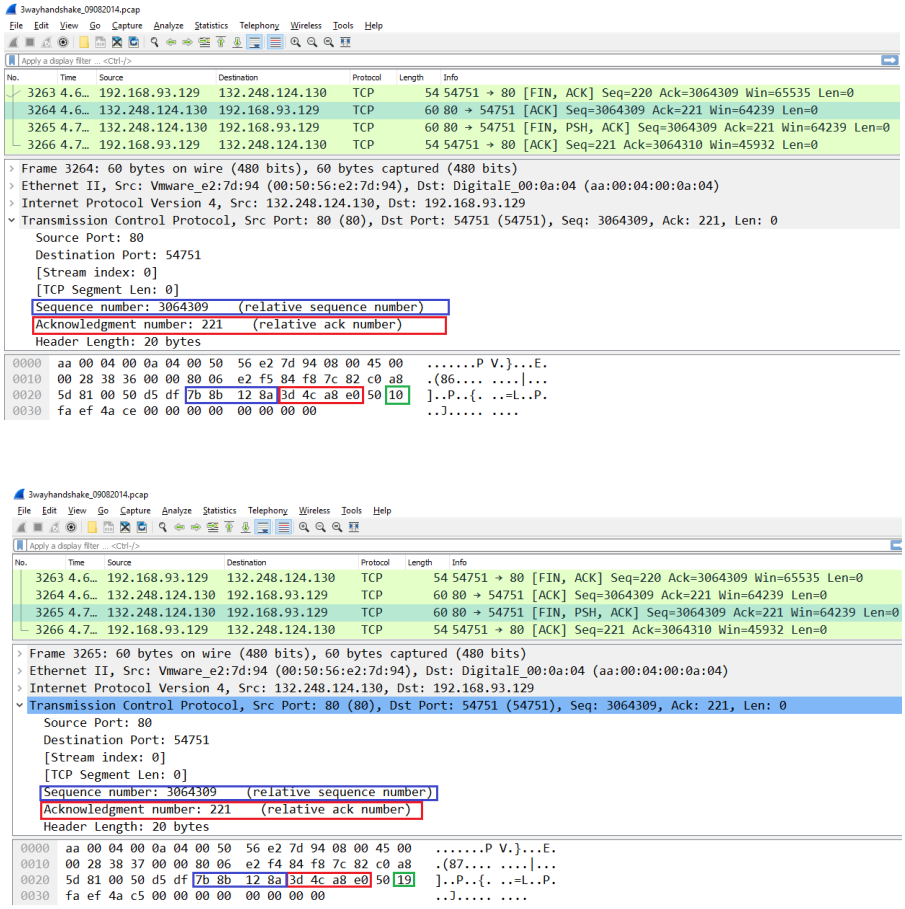

Figura 9: Fin de la conexión por parte del servidor

Finalmente, el cliente envía su número de secuencia final, y confirma al servidor el cierre de la conexión al incrementar el número de confirmación en 1, figura 10.

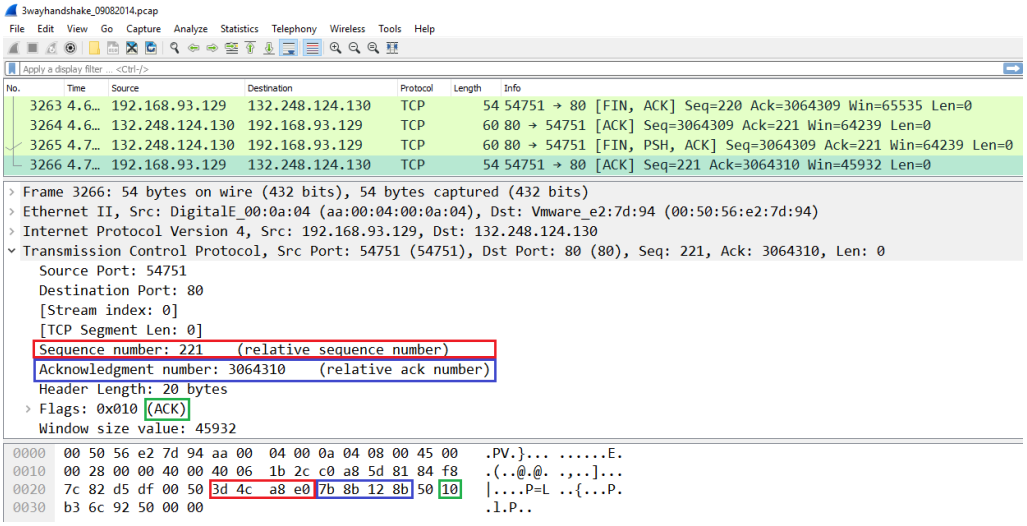

Figura 10: Confirmación del fin de conexión por parte del cliente

Referencias

- Kurose y Ross. Computer Networking, A top-down approach. 6ª ed. Pearson. 2012
- Andrew S. Tanenbaum, David J. Wetherall, Computer Networks, Prentice Hall, 5a ed., 2010.