Actividad Stop & Wait

1. `socketTCP.py`

1.1. 3-Way Handshake

En la presente sección se comentan las diferentes decisiones de diseño tomadas en consideración a la hora de diseñar e implementar el 3-Way Handshake para la clase `SocketTCP`.

- La primera consideración fue que para el lado del cliente los mensajes recibidos poseen un buffer de 16 bytes fijo a la hora de utilizar `sendto` del modulo sockets de python, esto se definió así pues los headers no alcanzan a sobrepasar este limite de espacio dada la manera en que se codifican, al tratarse de un tipo de dato `string` cada carácter ocupa de espacio 1 byte, luego el string de largo máximo, que representa el header que se puede enviar es:

$$1|||1|||1|||100 \tag{1}$$

  Donde el espacio ocupado es de 15 bytes, además cabe mencionar que las flags SYN, ACK, FIN no interesa como tal su valor pues están toman valor binario 0 o 1, ambos de tamaño 1 byte al representarse como string. Por otra parte, el numero de secuencia SEQ, también se encuentra acotado, pues este se escoge de manera aleatoria en un intervalo de 0 a 100, incluidos, por tanto el mayor SEQ posible, en cuanto al espacio que ocupa, es aquel que posea tres dígitos, por tanto 3 bytes al transformar el header a un string para su posterior codificación y envío.

- Para el handshake se estableció un paso extra dado que se esta utilizando el método de envío Stop & Wait, este paso es enviar desde el Servidor un ultimo ACK luego de haber recibido el ACK final del 3-Way Handshake, esta decisión de diseño fue tomada debido a que así el numero de secuencia finaliza incrementado para el lado del Cliente, de manera que al momento de empezar el envío de información como tal no exista conflictos en las respectivas verificaciones de SEQ. En la Figura 1 puede observarse un diagrama con el paso extra implementado, con lo cual se soluciona el caso borde en que el ACK enviado por el Cliente no es recibido por el Servidor.

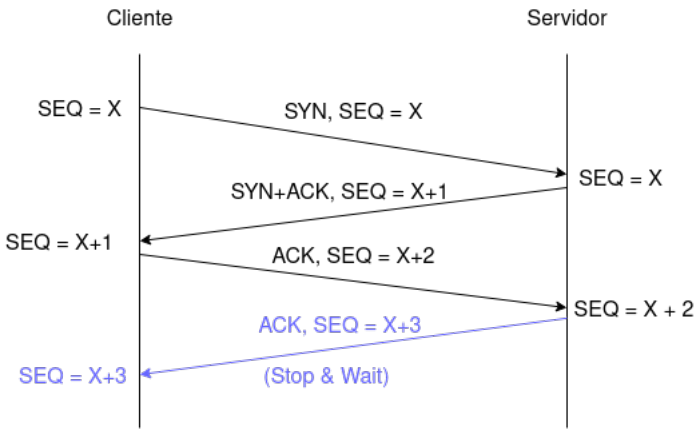

Figura 1: 3-Way Handshake con paso Stop & Wait.

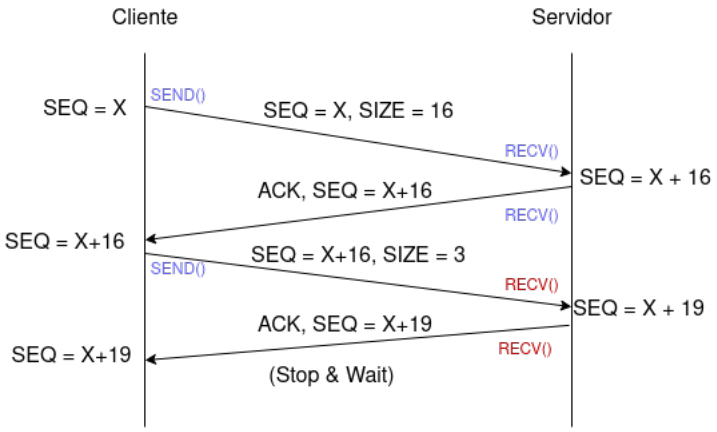

Figura 2: Envío de mensajes usando `send()` y `recv()`.

2. Envío de mensajes

En la Figura 2 se muestra el diagrama de envío de mensajes cuando se utiliza `send()` y `recv()` de la clase `SocketTCP`, donde los colores en las llamadas indica el proceso completo de una misma llamada.

3. Código

Es importante mencionar que la clase `SocketTCP` se encuentra completamente documentada y acorde a la convención PEP8 por tanto no es necesario realizar algún tipo de explicación respecto a los métodos que posee, sin embargo es importante mencionar el nombre que esta posee, que de acuerdo a la convención en los nombres, el modulo, es decir, el archivo que contiene la clase se llama `socketTCP`, por tanto fueron modificados los códigos entregados en la actividad que indicaban que el modulo se llamaba `SocketTCP`.