Flujos y sockets no bloqueantes

Sánchez Rico Raúl

Tarea 2.3

Hacer una pequeña disertación sobre las aplicaciones que manejan sockets no bloqueantes y flujos.
Indicar si es posible o no su implementación, en caso de que no sea posible indicar la razón y las fuentes que lo indican (fuentes verificables), en caso de que, si sea posible, agregar además de la explicación, los programas necesarios para ejecutar una aplicación que funcione como prueba de factibilidad.

Primero tenemos que mencionar la diferencia entre sockets no bloqueantes y bloqueantes, esto puede ser un poco fácil ya que como sus nombres lo indican, los no bloqueantes permiten la realización de las operaciones que llevan de entrada y de salida de los datos por el canal usado sin la necesidad de que un proceso sea bloqueado. Ahora, con lo que hemos llegado a ver en las clases vamos a definir ciertas cosas que desde mi opinión van a ayudara a resolver la cuestión principal. Los sockets de flujos al conectarse realizan una búsqueda de un camino libre entre origen y destino y se mantiene este camino en toda la conexión, en particular recv es un método de bloqueo en el que mientras no existan datos se quedara esperando a que llegue algo, esto si lo vemos de cierto punto va a hacer un problema, no creo que en todos los casos, pero sigue siendo un problema.

Ahora, cuando hacemos la llamada a un conect, el programa va a hacer incapaz de recuperar el control sino hasta que se realice la conexión, en caso de que no se llegue a esto va a producir un error, es aquí donde entran los sockets no bloqueantes.

Teniendo en cuenta que cuando usamos sockets en TCP, esto por default van a estar en bloqueantes, como cuando queremos leer desde una secuencia, esto no va a regresar nada sino hasta que encuentre al menos un byte del otro lado. Este proceso de espera se le dice bloqueante, ósea que hasta que se realice una operación la conexión va a estar bloqueada. Con ayuda de las fuentes y de las clases, al hacer una llamada a recv en modo no bloqueante, este devolverá cualquier tipo de dato que el sistema tenga en el buffer para ese socket sin embargo creo que este no va a poder esperar otros datos.

De igual modo pada con el método send, el cual va a colocar datos en el búfer, si el búfer se llegara a llenar, el sistema simplemente va a devolver un error al volver escribir en el. Por otra parte, un canal de socket no bloqueante no va a hacer capaz de leer mas datos de los que ya están definidos en el búfer de entrada del socket.

Es por ello por lo que no creo que seria capaz de hacerse una implementación de sockets no bloqueantes con flujos ya que no se puede leer la información completamente por medio de estos flujos.

Referencias

- SocketChannel (Java Platform SE 6). (2015, 19 noviembre). Recuperado 9 de junio de 2020, de https://docs.oracle.com/javase/6/docs/api/java/nio/channels/SocketChannel.html#read%28java.nio.ByteBuffer%29

- SocketChannel (Java Platform SE 7). (2018, 6 octubre). Recuperado 9 de junio de 2020, de https://docs.oracle.com/javase/7/docs/api/java/nio/channels/SocketChannel.html