

TECNOLOGÍAS DEL SECTOR FINANCIERO

Práctica 2: Mensajería de baja latencia

Andoni Alcelay

MÁSTER UNIVERSITARIO EN TECNOLOGÍAS DEL SECTOR FINANCIERO: FINTECH

# Introducción

En este documento se explicarán los resultados obtenidos en la práctica 2 realizada con shockets y wireshark. El documento está dividido en 3 partes donde 2 de ellas son resultados e implementación de diferentes tipos de shockets para enviar y recibir mensajes ejecutando una aplicación servidor y otra cliente. El último punto será analizar todas estas implementaciones con whireshark y analizar sus resultados.

# Práctica 1: Creación de comunicaciones TCP

En esta práctica crearemos un canal de comunicación simple TCP en Java, probando el envío

y recepción y la serialización de mensajes. Cada mensaje representa un precio de mercado,

con su instrumento, cantidad y precio. Esta práctica se divide en dos partes, la mensajería de tamaño fijo y la de tipo variable.

## Conexión de tamaño fijo

Tras seguir los pasos de la práctica, se procede a ejecutar la aplicación en el orden correcto, es decir, empezando por lanzar el servidor y después el cliente. El resultado ha sido el mostrado en las ilustraciones 1 y 2.

Servidor:

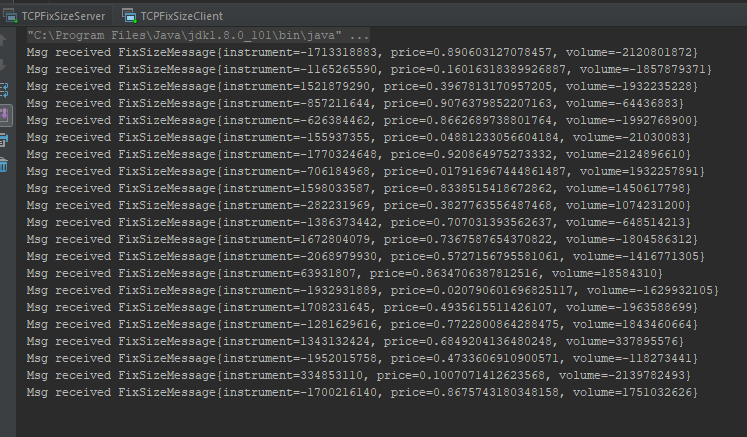


Ilustración 1: Ejecución de sockets tamaño fijo, servidor

Cliente:

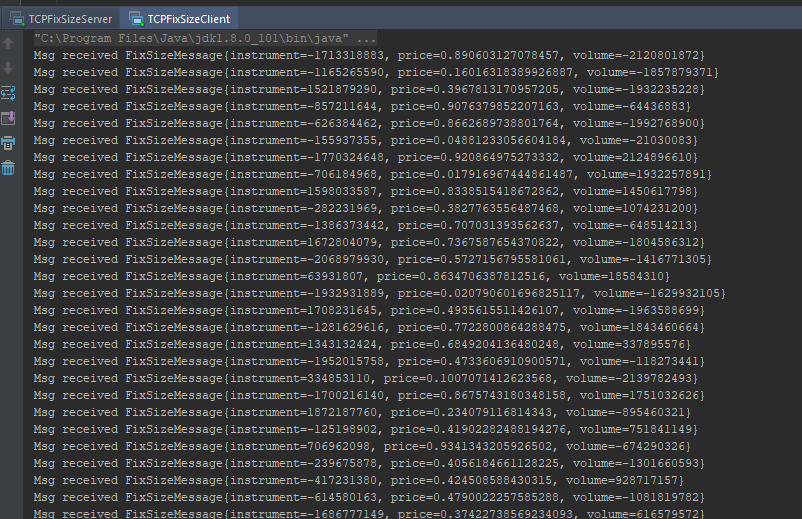


Ilustración 2: Ejecución de sockets tamaño fijo, cliente

## Conexión de tamaño variable

En este caso el tamaño de los mensajes que se envía es variable, para este primer intento se utilizar el valor 8 para el tamaño del mensaje aleatorio que se genera, valor que se le pasa al método generateRandomMsg();

Los resultados son los que se muestran en las ilustraciones 3 y 4 respectivamente:

Servidor

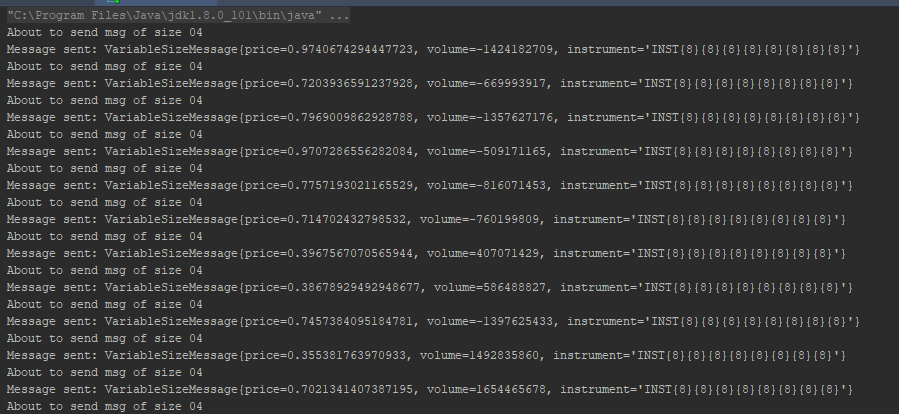


Ilustración 3: Mensajería de tamaño variable (8), servidor

Cliente

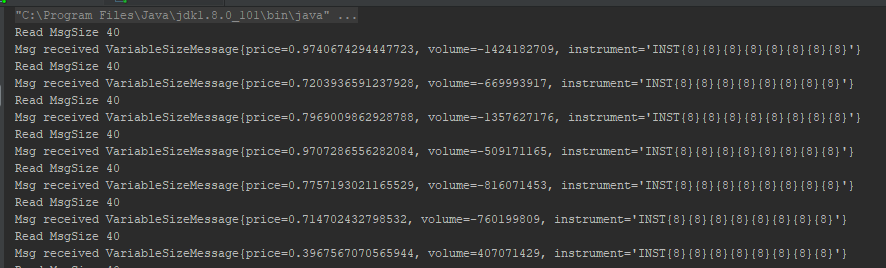


Ilustración 4: Mensajería de tamaño variable (8), cliente

Para la segunda parte de la práctica, hay que intentar cambiar este tamaño de los mensajes enviados, que actualmente es 8. Para este segundo intento se le subirá ese valor a 80000. Los resultados son mostrados en las ilustraciones 5 y 6.

Servidor:

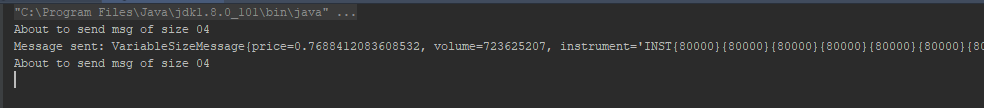


Ilustración 5: Mensajería de tamaño variable (80000), servidor

Cliente:

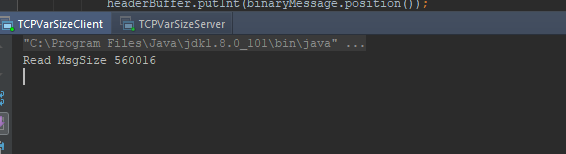


Ilustración 6: Mensajería de tamaño variable (80000), cliente

En el ejercicio anterior usamos un 8, creando un mensaje pequeño. Si lo incrementáis a 80000 y volvéis a ejecutar veréis que el servidor envía el primer mensaje y se bloquea justo antes de enviar el segundo. El cliente recibe la cabecera con el tamaño del mensaje, pero se queda en bucle infinito esperando que haya suficientes bytes en el stream.

Para evitar esto se ha creado una clase capaz de recibir mensajes de gran tamaño, su resultado se muestra en las ilustraciones 7 y 8.

Servidor:

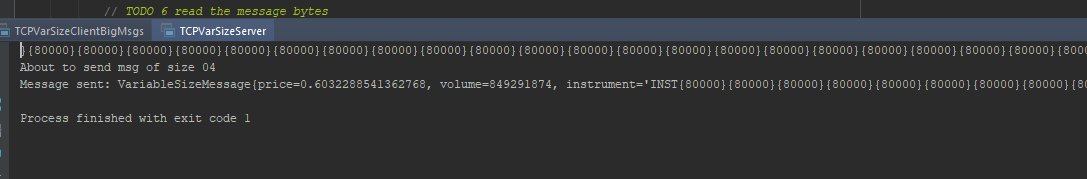


Ilustración 7: Mensajería de gran tamaño, servidor

Cliente:

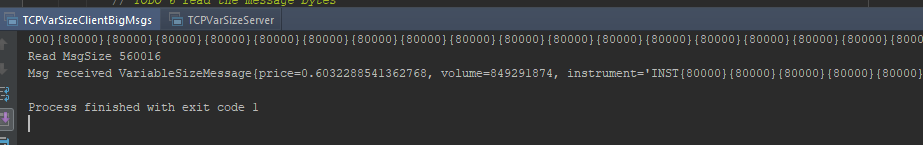


Ilustración 8: Mensajería de gran tamaño, cliente

# Práctica 2: Creación de comunicaciones Multicast

En esta práctica crearemos un canal de comunicación simple Multicast en Java, probando el envío y recepción y la serialización de mensajes. Cada mensaje representa un precio de mercado, con su instrumento, cantidad y precio. A diferencia de TCP, en Multicast las comunicaciones son UDP, es decir no hay sesión, ni control de flujo, ni de orden, etc.

Tras ejecutar el cliente y el servidor el resultado ha sido el mostrado en las ilustraciones 9 y 10.

Servidor:

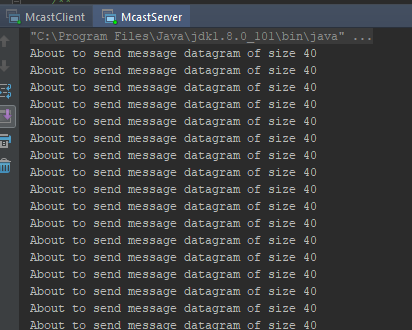


Ilustración 9: Mensajería multicast, servidor

Cliente:

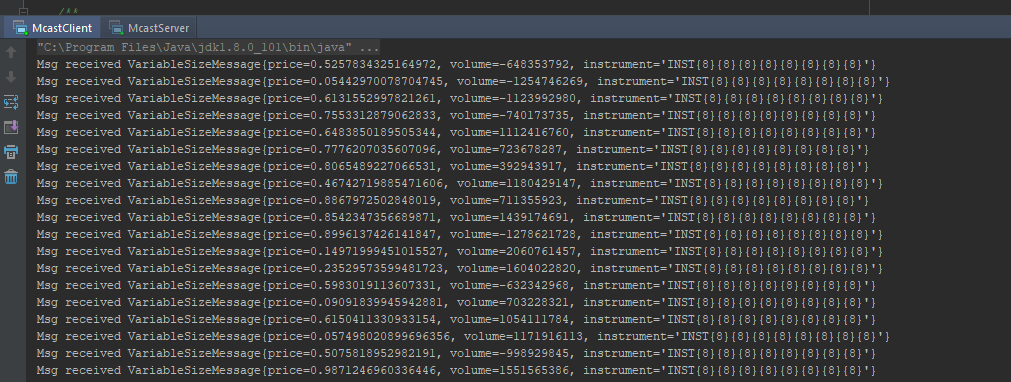


Ilustración 10: Mensajería multicast, cliente

Como en la práctica anterior se cambiará el tamaño del mensaje a enviar a 80000, y podrá verse que no es capaz de enviar porque en este caso se usan Datagramas, y estos tienen un tamaño fijo que no se puede sobrepasar.

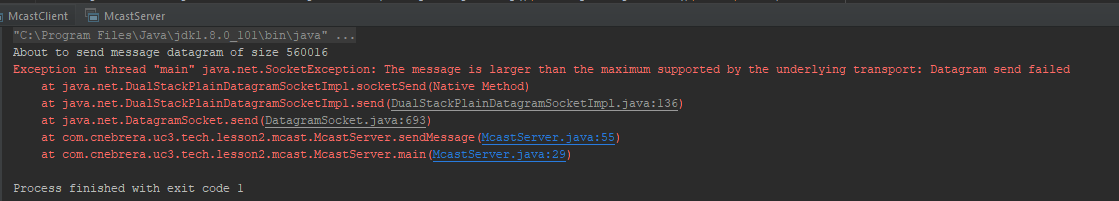


Ilustración 11: Error por sobrepasar el tamaño máximo de Datagram

# Wireshark

## Cliente sin servidor

Como se ve en la ilustración 12, se ha hecho un envío de paquetes con el servidor apagado, esto se ve porque hay envíos por el puerto 6789 que no se reciben nunca, pueden verse con color rojo

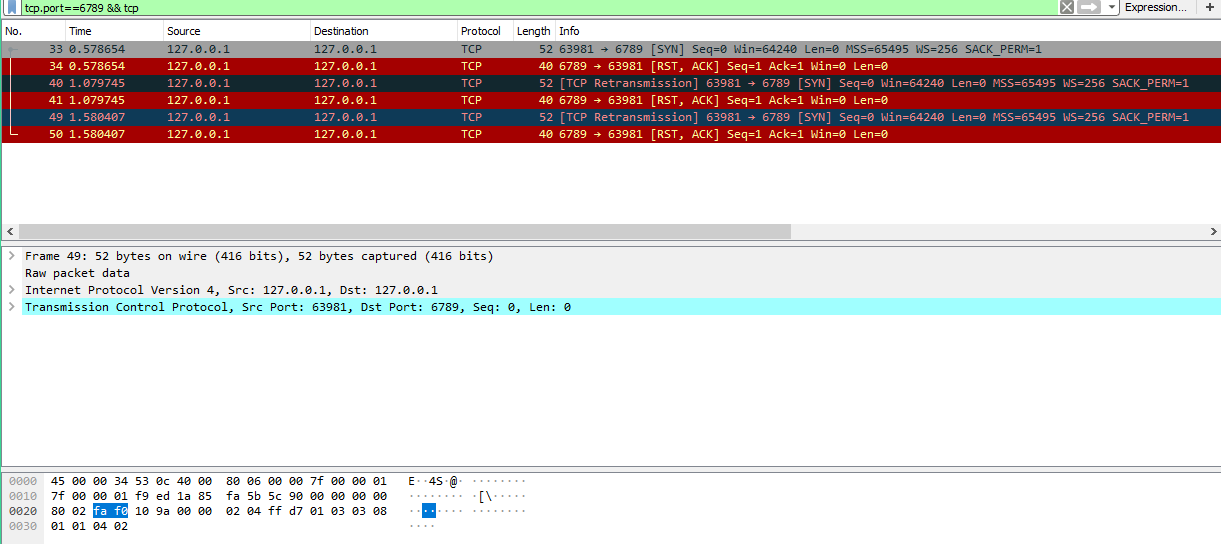


Ilustración : Envío de paquetes sin respuesta

En cambio en la ilustración 13 la cosa cambia, se envían paquetes por el puerto 6789 que sí que tienen respuesta, y esta se puede ver en la siguiente traza al envío.

## Cliente con servidor

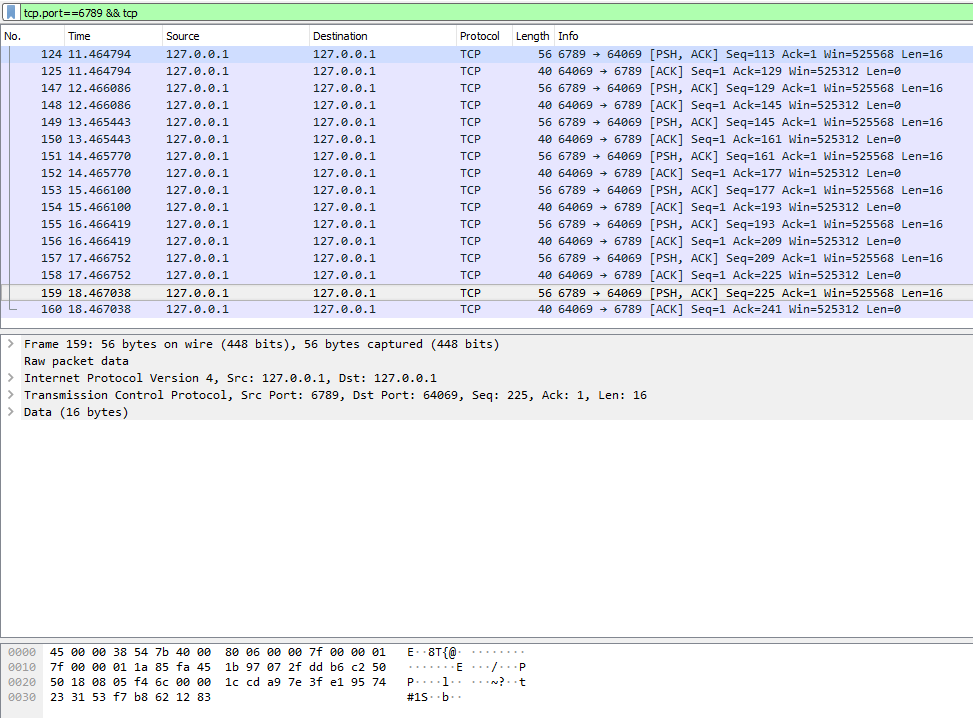


Ilustración : Envío de paquetes con respuesta