

## Práctica Lección 2. Mensajería de Baja Latencia I

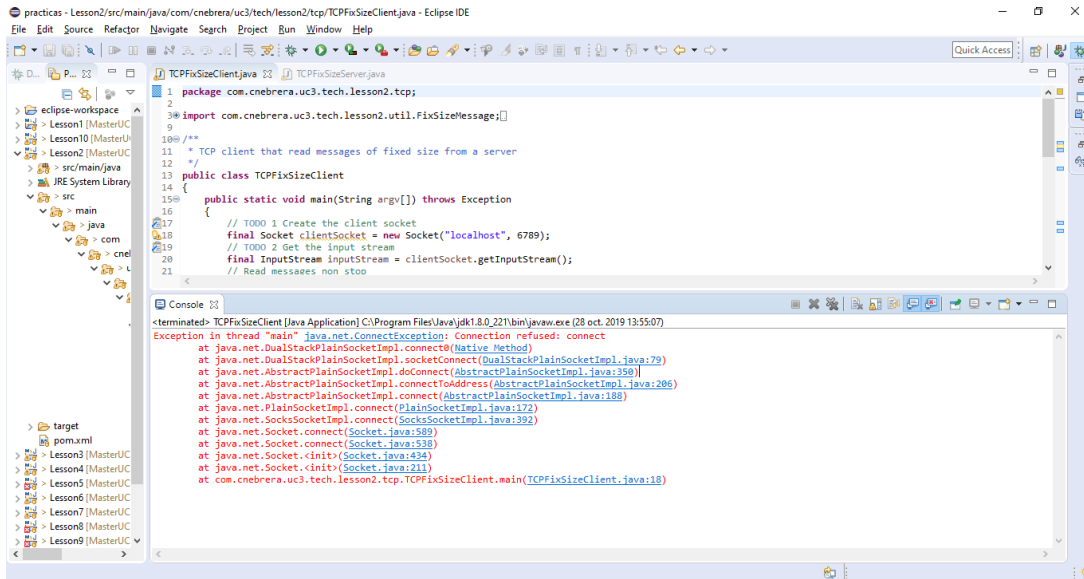
### Práctica 1: Creación de comunicaciones TCP

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.

- Mensajes de tamaño fijo

#### Ejecutando

1) Si ejecutáis únicamente el cliente veréis que da una excepción “ConnectionRefused” ya que no encuentra al servidor en la IP y puerto indicada.



```
package com.cnebrera.uc3.tech.lesson2.tcp;

import com.cnebrera.uc3.tech.lesson2.util.FixSizeMessage;

/**
 * TCP client that read messages of fixed size from a server
 */
public class TCPFixSizeClient {

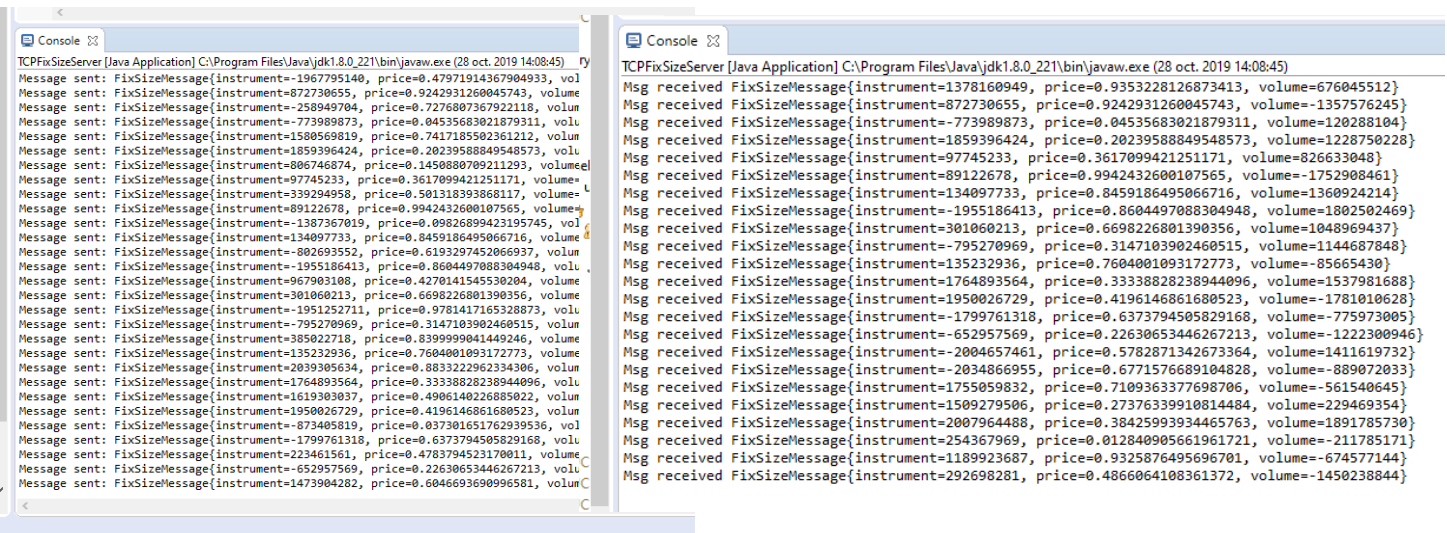
    public static void main(String argv[]) throws Exception {
        // TODO 1 Create the client socket
        final Socket clientSocket = new Socket("localhost", 6789);
        // TODO 2 Get the input stream
        final InputStream inputStream = clientSocket.getInputStream();
        // Read messages non stop
    }
}
```

```
<terminated> TCPFixSizeClient [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (28 oct. 2019 13:55:07)
Exception in thread "main" java.net.ConnectException: Connection refused: connect
    at java.net.DualStackPlainSocketImpl.connect0(Native Method)
    at java.net.DualStackPlainSocketImpl.connect(SocketImpl.java:79)
    at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
    at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)
    at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188)
    at java.net.Socket.connect(Socket.java:122)
    at java.net.SocksSocketImpl.connect(SocketImpl.java:392)
    at java.net.Socket.connect(Socket.java:589)
    at java.net.Socket.connect(Socket.java:538)
    at java.net.Socket.<init>(Socket.java:434)
    at java.net.Socket.<init>(Socket.java:211)
    at com.cnebrera.uc3.tech.lesson2.tcp.TCPFixSizeClient.main(TCPFixSizeClient.java:18)
```

2) Hagámoslo ahora en el orden correcto, primero el servidor. Veremos que se queda esperando a que llegue algún cliente. Si ejecutamos ahora el cliente veremos como el servidor escribe en el socket de cliente y el cliente los recibe, 1 mensaje por segundo hasta que no los paremos.

#### Mensajes enviados

#### Mensajes recibidos



```
TCPFixSizeServer [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (28 oct. 2019 14:08:45)
Msg sent: FixSizeMessage{instrument=1967795140, price=0.47971914367904933, vol
Msg sent: FixSizeMessage{instrument=872730655, price=0.9242931260045743, volume=
Msg sent: FixSizeMessage{instrument=258949704, price=0.727680736792118, volum
Msg sent: FixSizeMessage{instrument=773989873, price=0.04535683021879311, volu
Msg sent: FixSizeMessage{instrument=1580569819, price=0.7417185502361212, volum
Msg sent: FixSizeMessage{instrument=1859396424, price=0.20239588849548573, volu
Msg sent: FixSizeMessage{instrument=806746874, price=0.1450880709211293, volume
Msg sent: FixSizeMessage{instrument=97745233, price=0.3617099421251171, volume=
Msg sent: FixSizeMessage{instrument=339294958, price=0.501318393068117, volume=
Msg sent: FixSizeMessage{instrument=89122678, price=0.9942432600107565, volume=
Msg sent: FixSizeMessage{instrument=1387367810, price=0.09826899423195745, vol
Msg sent: FixSizeMessage{instrument=134097733, price=0.8459186495066716, volum
Msg sent: FixSizeMessage{instrument=802693552, price=0.6193297452066937, volum
Msg sent: FixSizeMessage{instrument=1955186413, price=0.8604497088304948, volu
Msg sent: FixSizeMessage{instrument=967903108, price=0.4270141545530204, volum
Msg sent: FixSizeMessage{instrument=301060213, price=0.6698226801390356, volum
Msg sent: FixSizeMessage{instrument=1951252711, price=0.9781417165328873, volu
Msg sent: FixSizeMessage{instrument=795270969, price=0.3147103902460515, volum
Msg sent: FixSizeMessage{instrument=385022718, price=0.8399999041449246, volum
Msg sent: FixSizeMessage{instrument=13523236, price=0.7604001093172773, volum
Msg sent: FixSizeMessage{instrument=2039305634, price=0.883222962334306, volum
Msg sent: FixSizeMessage{instrument=1764893564, price=0.3338828238944096, volu
Msg sent: FixSizeMessage{instrument=1619303037, price=0.4906140226885022, volum
Msg sent: FixSizeMessage{instrument=1950026729, price=0.4196146861680523, volum
Msg sent: FixSizeMessage{instrument=873405819, price=0.037301651762939536, vol
Msg sent: FixSizeMessage{instrument=1799761318, price=0.637394505829168, volu
Msg sent: FixSizeMessage{instrument=223461561, price=0.4783794523170011, volum
Msg sent: FixSizeMessage{instrument=652957569, price=0.22630653446267213, volu
Msg sent: FixSizeMessage{instrument=1473904282, price=0.6046693690996581, volum
```

```
TCPFixSizeServer [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (28 oct. 2019 14:08:45)
Msg received FixSizeMessage{instrument=1378160949, price=0.9353228126873413, volume=676045512}
Msg received FixSizeMessage{instrument=872730655, price=0.9242931260045743, volume=-1357576245}
Msg received FixSizeMessage{instrument=773989873, price=0.04535683021879311, volume=120288104}
Msg received FixSizeMessage{instrument=1859396424, price=0.20239588849548573, volume=1228750228}
Msg received FixSizeMessage{instrument=97745233, price=0.3617099421251171, volume=826633048}
Msg received FixSizeMessage{instrument=89122678, price=0.9942432600107565, volume=-1752908461}
Msg received FixSizeMessage{instrument=134097733, price=0.8459186495066716, volume=1360924214}
Msg received FixSizeMessage{instrument=1955186413, price=0.8604497088304948, volume=1802502469}
Msg received FixSizeMessage{instrument=301060213, price=0.6698226801390356, volume=1048969437}
Msg received FixSizeMessage{instrument=795270969, price=0.3147103902460515, volume=1144687848}
Msg received FixSizeMessage{instrument=13523236, price=0.7604001093172773, volume=-85665430}
Msg received FixSizeMessage{instrument=1764893564, price=0.3338828238944096, volume=1537981688}
Msg received FixSizeMessage{instrument=1950026729, price=0.4196146861680523, volume=-1781010628}
Msg received FixSizeMessage{instrument=1799761318, price=0.637394505829168, volume=229469354}
Msg received FixSizeMessage{instrument=652957569, price=0.22630653446267213, volume=-1222300946}
Msg received FixSizeMessage{instrument=-2004657461, price=0.5782871342673364, volume=1411619732}
Msg received FixSizeMessage{instrument=2034866955, price=0.6771576689104828, volume=-889072033}
Msg received FixSizeMessage{instrument=1755059832, price=0.7109363377698706, volume=-561540645}
Msg received FixSizeMessage{instrument=1509279506, price=0.38425993394465763, volume=1891785730}
Msg received FixSizeMessage{instrument=254367969, price=0.012840905661961721, volume=-211785171}
Msg received FixSizeMessage{instrument=1189923687, price=0.9325876495696701, volume=-674577144}
Msg received FixSizeMessage{instrument=292698281, price=0.4866064108361372, volume=-1450238844}
```

- Mensajes de tamaño variable

Mensajes recibidos:

```
Console
TCPVarSizeClient [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (28 oct. 19 14:26:46)
Msg received VariableSizeMessage{price=0.7413483819747549, volume=980015273, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
Read MsgSize 40
Msg received VariableSizeMessage{price=0.8874818129265238, volume=-1283770649, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
Read MsgSize 40
Msg received VariableSizeMessage{price=0.796021818168078, volume=-733161468, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
Read MsgSize 40
Msg received VariableSizeMessage{price=0.8698369581833497, volume=1403485817, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
Read MsgSize 40
Msg received VariableSizeMessage{price=0.012236970793985114, volume=1732195375, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
Read MsgSize 40
Msg received VariableSizeMessage{price=0.49661330684654004, volume=-1716609524, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
Read MsgSize 40
Msg received VariableSizeMessage{price=0.03126170571443876, volume=829647584, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
Read MsgSize 40
```

Mensajes enviados:

```
<terminated> TCPVarSizeServer [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (28 oct. 2019 14:26:41)
Message sent: VariableSizeMessage{price=0.90838616215286, volume=-1686878794, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
About to send msg of size 04
Message sent: VariableSizeMessage{price=0.25234092776951367, volume=729363881, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
About to send msg of size 04
Message sent: VariableSizeMessage{price=0.9469174800691396, volume=317096329, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
About to send msg of size 04
Message sent: VariableSizeMessage{price=0.9345562559918138, volume=-320402353, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
About to send msg of size 04
Message sent: VariableSizeMessage{price=0.31081928344244136, volume=-1819763825, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
About to send msg of size 04
Message sent: VariableSizeMessage{price=0.8247546453892669, volume=1167475365, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
About to send msg of size 04
Message sent: VariableSizeMessage{price=0.3584061188590467, volume=-1971094662, instrument='INST{8}{8}{8}{8}{8}{8}{8}{8}'}
About to send msg of size 04
```

- Usando el mismo código de la práctica anterior, incrementad el número que se da al método:

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. Luego de la ejecución no recibe el servidor no envía ningún mensaje y se queda en la siguiente pantalla:

[illegible]

Para solucionar esto se debe leer desde el cliente en pequeños bloques en lugar de esperar a todo el mensaje. Y al realizar las mejoras planteadas, el servidor envía y recibe de forma continua,

## Mensaje recibido

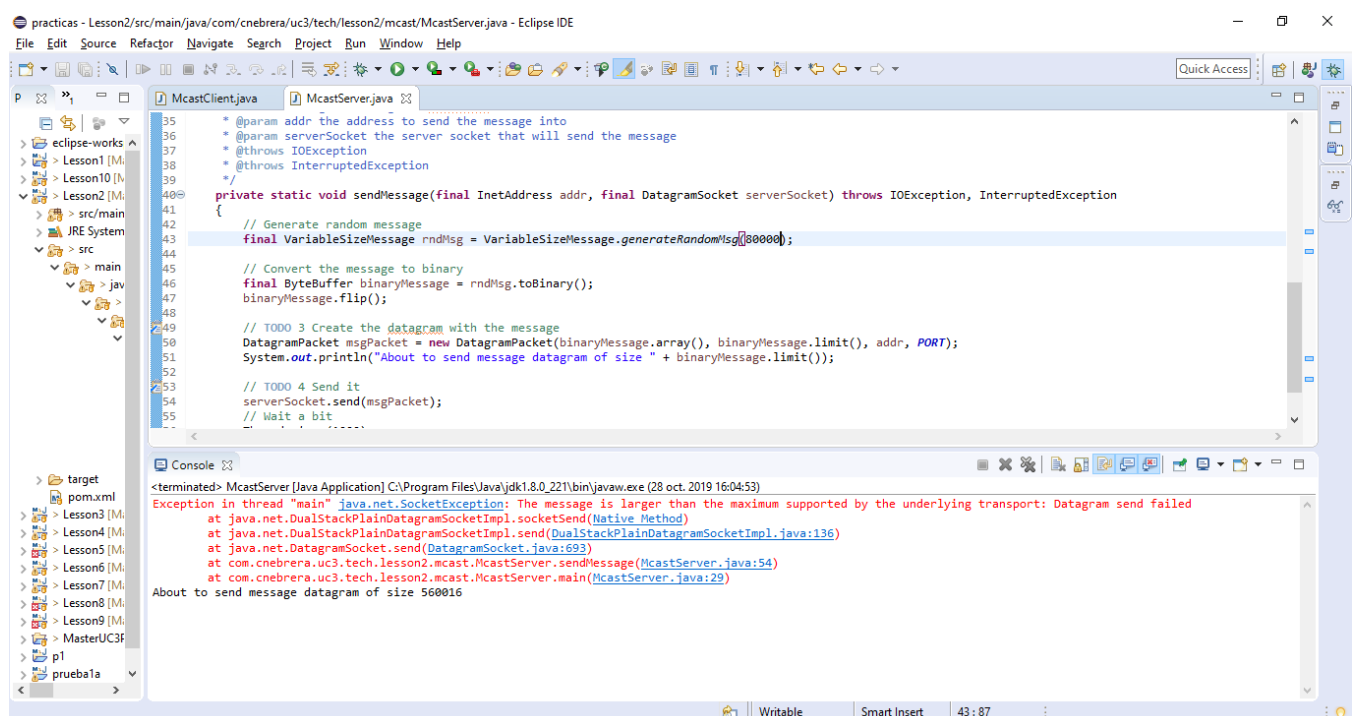
[illegible]

Mensaje enviado

[illegible]

## 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.



## Práctica 3: Analizando con Wireshark

### a. Flujo TCP normal

Primero abrimos únicamente el cliente, va a fallar porque no hay servidor. Deberíais ver dos entradas en la lista. Una primera del cliente en un puerto aleatorio llamando al puerto del servidor de tipo SYNC, y una respuesta de rechazo por parte del SO.

Capturing from Adapter for loopback traffic capture

Filter: tcp && tcp.port==6789

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	61158 → 6789 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
2	0.000086	127.0.0.1	127.0.0.1	TCP	44	6789 → 61158 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	0.691252	127.0.0.1	127.0.0.1	TCP	56	[TCP Retransmission] 61158 → 6789 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
4	0.691330	127.0.0.1	127.0.0.1	TCP	44	6789 → 61158 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	1.191162	127.0.0.1	127.0.0.1	TCP	56	[TCP Retransmission] 61158 → 6789 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
6	1.191289	127.0.0.1	127.0.0.1	TCP	44	6789 → 61158 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	1.693304	127.0.0.1	127.0.0.1	TCP	56	[TCP Retransmission] 61158 → 6789 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
8	1.693385	127.0.0.1	127.0.0.1	TCP	44	6789 → 61158 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	2.292416	127.0.0.1	127.0.0.1	TCP	56	[TCP Retransmission] 61158 → 6789 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
10	2.292536	127.0.0.1	127.0.0.1	TCP	44	6789 → 61158 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Frame 1: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0  
 Null/Loopback  
 Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
 0000 02 00 00 00 45 00 00 34 b8 8f 40 00 06 00 00 ...E..4..@..  
 0010 7f 00 00 01 7f 00 00 01 ee e6 1a 85 4d 22 42 cf .....N"B..

Adapter for loopback traffic capture: <live capture in progress> | Packets: 10 · Displayed: 10 (100.0%) | Profile: Default

Si repetimos con el servidor arrancado veremos que además de los mensajes de sincronización iniciales ahora además tenemos mensajes de datos, ACK de control y mensajes de actualización de tamaño de ventana.

Capturing from Adapter for loopback traffic capture

Filter: tcp && tcp.port==6789

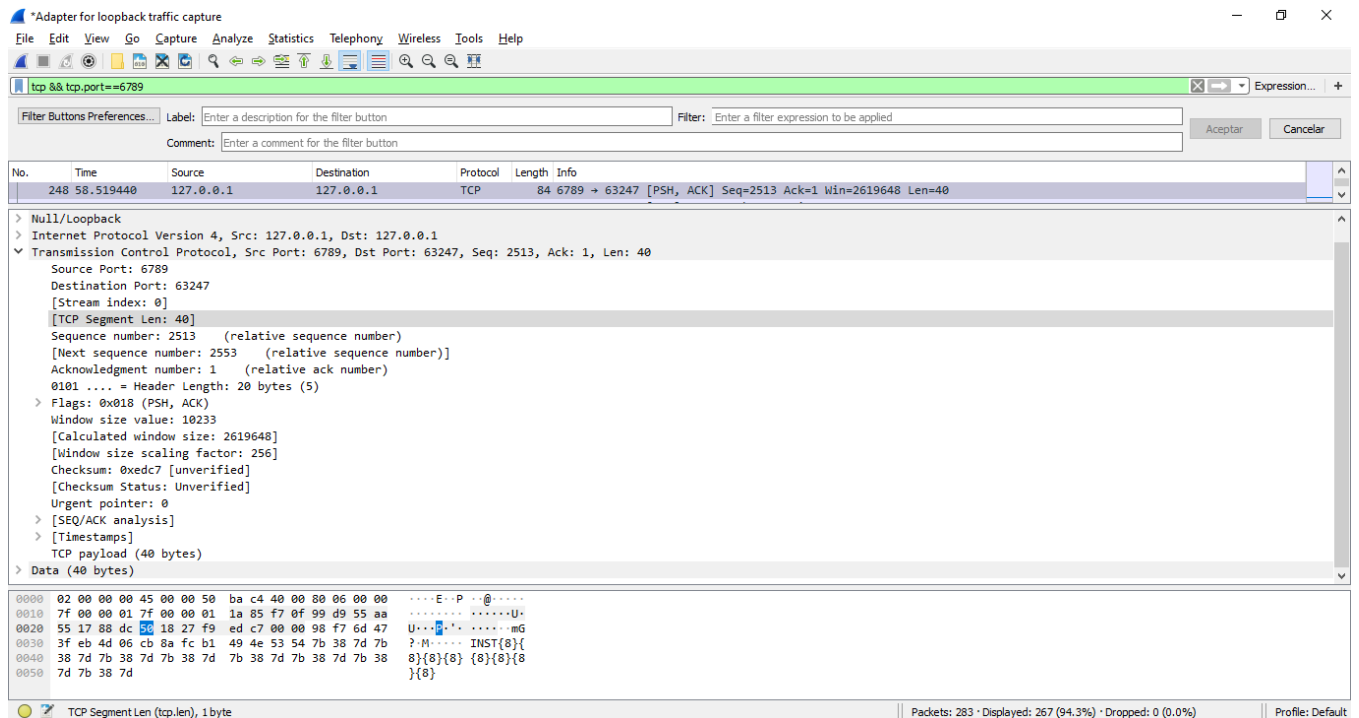
No.	Time	Source	Destination	Protocol	Length	Info
16	19.332902	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=81 Win=2619648 Len=0
17	20.340051	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=81 Ack=1 Win=2619648 Len=16
18	20.340163	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=97 Win=2619648 Len=0
19	21.390063	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=97 Ack=1 Win=2619648 Len=16
20	21.390177	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=113 Win=2619648 Len=0
21	22.439663	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=113 Ack=1 Win=2619648 Len=16
22	22.439782	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=129 Win=2619648 Len=0
23	23.475380	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=129 Ack=1 Win=2619648 Len=16
24	23.475489	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=145 Win=2619648 Len=0
25	24.506659	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=145 Ack=1 Win=2619648 Len=16
26	24.506884	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=161 Win=2619392 Len=0
27	25.538390	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=161 Ack=1 Win=2619648 Len=16
28	25.538492	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=177 Win=2619392 Len=0
29	26.541613	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=177 Ack=1 Win=2619648 Len=16
30	26.541726	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=193 Win=2619392 Len=0
31	27.593676	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=193 Ack=1 Win=2619648 Len=16
32	27.593782	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=209 Win=2619392 Len=0
33	28.607134	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=209 Ack=1 Win=2619648 Len=16
34	28.607247	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=225 Win=2619392 Len=0
35	29.640635	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=225 Ack=1 Win=2619648 Len=16
36	29.640735	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=241 Win=2619392 Len=0
37	30.642272	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=241 Ack=1 Win=2619648 Len=16
38	30.642387	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=257 Win=2619392 Len=0
39	31.673888	127.0.0.1	127.0.0.1	TCP	60	6789 → 63224 [PSH, ACK] Seq=257 Ack=1 Win=2619648 Len=16
40	31.673978	127.0.0.1	127.0.0.1	TCP	44	63224 → 6789 [ACK] Seq=1 Ack=273 Win=2619392 Len=0

Frame 4: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0  
 Null/Loopback  
 Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
 0000 02 00 00 00 45 00 00 34 b8 99 40 00 06 00 00 ...E..4..@..  
 0010 7f 00 00 01 7f 00 00 01 f6 f8 1a 85 56 f4 09 ab .....V...

Adapter for loopback traffic capture: <live capture in progress> | Packets: 40 · Displayed: 37 (92.5%) | Profile: Default

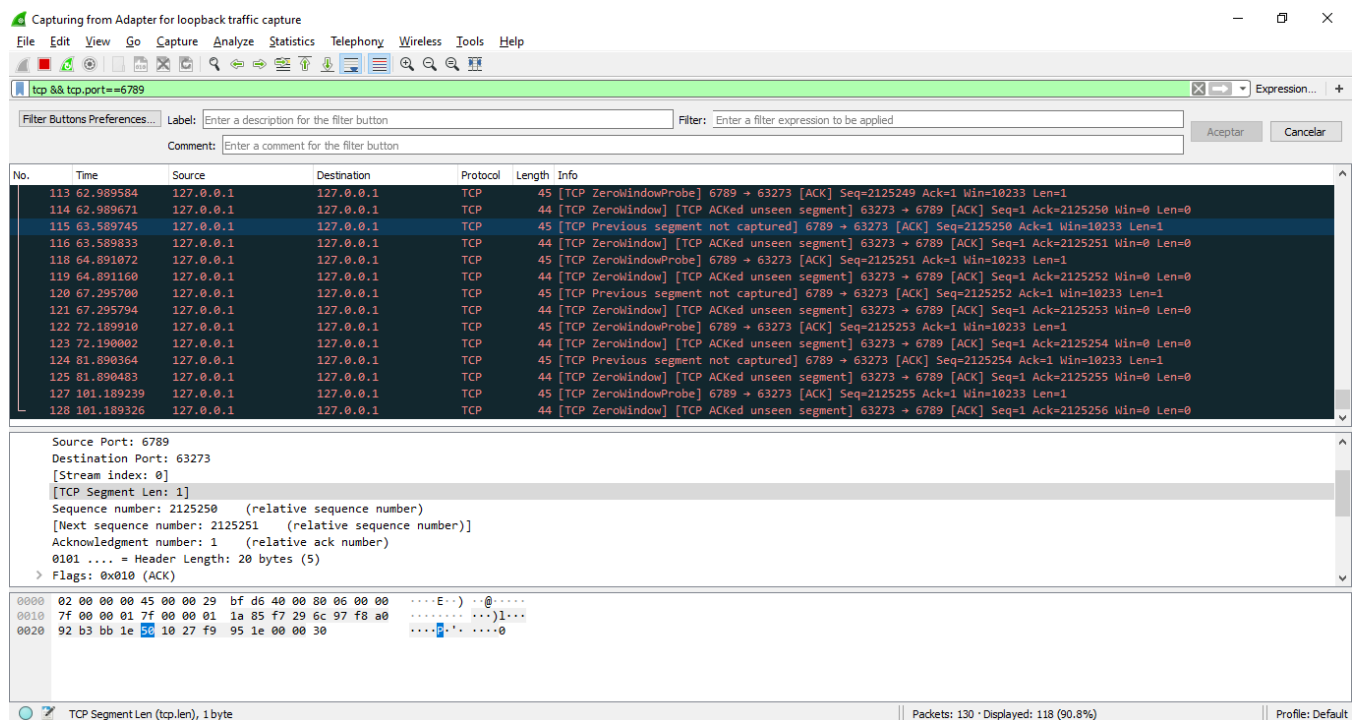


Vamos a repetir lo mismo con los mensajes de tamaño variable, esta vez pequeños de tamaño 8 como estaba en la práctica original. Si volvemos a ir a los mensajes de datos, veremos que esta vez al estar enviando Strings en UTF-8 wireshark es capaz de decodificar parte del mensaje



### b. Flujo TCP bloqueado

Esta vez vamos a repetir los mensajes de tamaño variable grandes que dejaban bloqueado el servidor y el cliente.



Vamos a repetir el análisis, pero con un flujo de mensajería multicast utilizando la práctica anterior de multicast.

[illegible]