

Lección 2 - Mensajería de baja latencia

Tecnologías del sector financiero

Asier Sampietro Alberdi

Práctica 1: Creación de comunicaciones TCP

1. Mensajes de tamaño fijo

Después de realizar las modificaciones, y ejecutando únicamente el cliente, obtenemos este resultado, debido a la imposibilidad de establecer un túnel de comunicación con el servidor.

```
[INFO] -----
[INFO] Building Lesson2 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Lesson2 ---
[WARNING]
java.net.ConnectException: Conexión rehusada (Connection refused)
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    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.SocksSocketImpl.connect(SocksSocketImpl.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:244)
    at com.cnebrera.uc3.tech.lesson2.tcp.TCPFixedSizeClient.main(TCPFixedSizeClient.java:18)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.codehaus.mojo.exec.ExecJavaMojo$1.run(ExecJavaMojo.java:282)
    at java.lang.Thread.run(Thread.java:748)
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 0.839 s
[INFO] Finished at: 2018-11-04T18:08:03+01:00
[INFO] Final Memory: 11M/212M
```

Si se ejecuta en el orden correcto, primero dejando el servidor a la espera de una solicitud de conexión y luego realizando dicha petición desde el cliente, se ejecutan ambos programas correctamente.

```
sampru@debian:/media/sampru/Datuak/eskola/tec-sec-fin/MasterUC3Practices/Lesson2$ mvn exec:java -Dexec.mainClass="com.cnebrera.uc3.tech.Lesson2.tcp.TCPFixedSizeServer"
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Lesson2 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Lesson2 ---
Message sent: FixSizeMessage{instrument=2022027713, price=0.779184125810782, volume=1864072645}
Message sent: FixSizeMessage{instrument=960886769, price=0.0585816156596513, volume=1096965549}
Message sent: FixSizeMessage{instrument=259268720, price=0.19589967408613385, volume=131065014}
Message sent: FixSizeMessage{instrument=1791359190, price=0.7354225988213174, volume=180214126}
Message sent: FixSizeMessage{instrument=1561164395, price=0.8522392803248179, volume=610853312}
Message sent: FixSizeMessage{instrument=1495910562, price=0.5781803998850776, volume=1070215141}
Message sent: FixSizeMessage{instrument=1494124278, price=0.5016116237906402, volume=2141679221}
Message sent: FixSizeMessage{instrument=1586051690, price=0.820030083923786, volume=949453678}
Message sent: FixSizeMessage{instrument=1911186834, price=0.6281846259527086, volume=613344596}
Message sent: FixSizeMessage{instrument=703870482, price=0.1415060655643754, volume=1320176392}
Message sent: FixSizeMessage{instrument=1167782437, price=0.49456400594094563, volume=1779404729}
Message sent: FixSizeMessage{instrument=832052187, price=0.8297553820845252, volume=1515493063}
^Csampru@debian:/media/sampru/Datuak/eskola/tec-sec-fin/MasterUC3PracticesLesson2$

sampru@debian:/media/sampru/Datuak/eskola/tec-sec-fin/MasterUC3Practices/Lesson2$ mvn exec:java -Dexec.mainClass="com.cnebrera.uc3.tech.Lesson2.tcp.TCPFixedSizeClient"
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Lesson2 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Lesson2 ---
Received message: FixSizeMessage{instrument=2022027713, price=0.779184125810782, volume=1864072645}
Received message: FixSizeMessage{instrument=960886769, price=0.0585816156596513, volume=1096965549}
Received message: FixSizeMessage{instrument=259268720, price=0.19589967408613385, volume=131065014}
Received message: FixSizeMessage{instrument=1791359190, price=0.7354225988213174, volume=180214126}
Received message: FixSizeMessage{instrument=1561164395, price=0.8522392803248179, volume=610853312}
Received message: FixSizeMessage{instrument=1495910562, price=0.5781803998850776, volume=1070215141}
Received message: FixSizeMessage{instrument=1494124278, price=0.5016116237906402, volume=2141679221}
Received message: FixSizeMessage{instrument=1586051690, price=0.820030083923786, volume=949453678}
Received message: FixSizeMessage{instrument=1911186834, price=0.6281846259527086, volume=613344596}
Received message: FixSizeMessage{instrument=703870482, price=0.1415060655643754, volume=1320176392}
Received message: FixSizeMessage{instrument=1167782437, price=0.49456400594094563, volume=1779404729}
Received message: FixSizeMessage{instrument=832052187, price=0.8297553820845252, volume=1515493063}
^Csampru@debian:/media/sampru/Datuak/eskola/tec-sec-fin/MasterUC3PracticesLesson2$
```

Los precios llegan correctamente hasta que se ha cerrado el servidor matando el proceso. El cliente se queda a la espera de más paquetes, por lo que no se da cuenta de que el túnel ha caído. De esta forma se puede matar el proceso del cliente después, sin tener que ver ningún mensaje de error que pueda mostrar algún otro orden.

2. Mensajes de tamaño variable

Ejecutando el cliente y el servidor de mensajes de tamaño variable, y pasando un 8 a la función `VariableSizeMessage.generateRandomMsg()`, conseguimos el siguiente resultado:

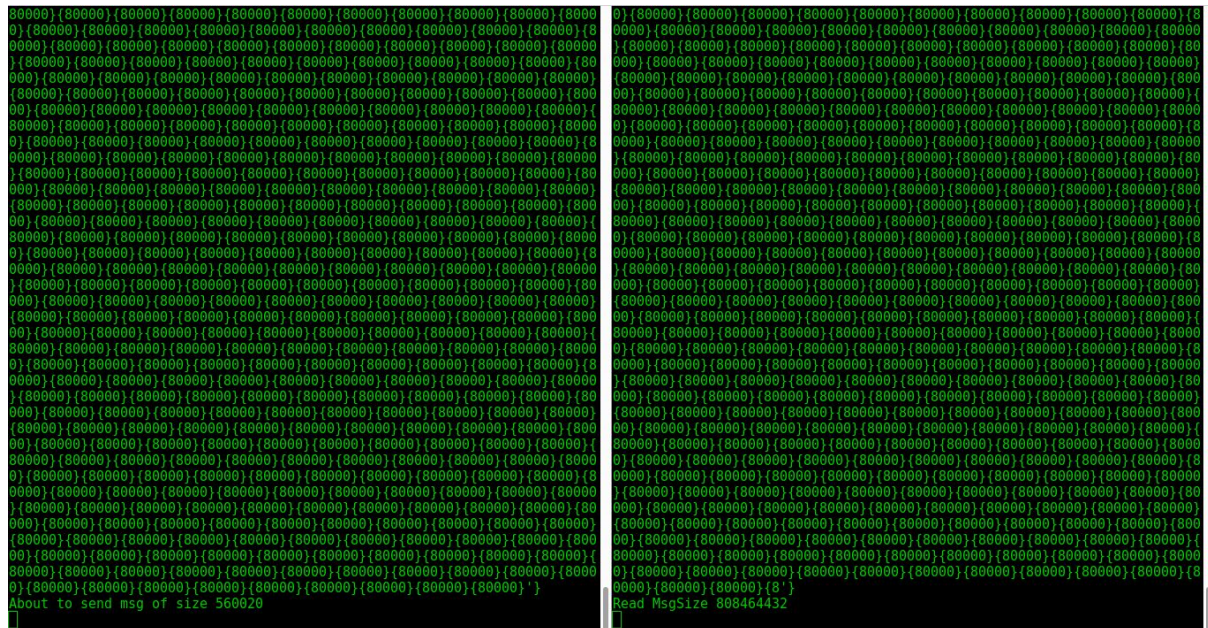
```
[INFO] -----
[INFO] Building Lesson2 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Lesson2 ---
About to send msg of size 44
Message sent: VariableSizeMessage{price=0.3326305703380463, volume=1676779082, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
About to send msg of size 44
Message sent: VariableSizeMessage{price=0.07065803512640578, volume=310798300, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
About to send msg of size 44
Message sent: VariableSizeMessage{price=0.6410843215904908, volume=1838316515, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
About to send msg of size 44
Message sent: VariableSizeMessage{price=0.35616341133175833, volume=640417672, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
About to send msg of size 44
Message sent: VariableSizeMessage{price=0.02791475958926104, volume=920905018, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
About to send msg of size 44
Message sent: VariableSizeMessage{price=0.22370939556177194, volume=125809507, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
About to send msg of size 44
Message sent: VariableSizeMessage{price=0.15608589096032743, volume=953519745, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
About to send msg of size 44
Message sent: VariableSizeMessage{price=0.13704370458866222, volume=2143473156, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
About to send msg of size 44
Message sent: VariableSizeMessage{price=0.6469131423152766, volume=1475089579, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }

[INFO] -----
[INFO] Building Lesson2 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Lesson2 ---
Read MsgSize 40
Msg received VariableSizeMessage{price=0.3326305703380463, volume=1676779082, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
Read MsgSize 40
Msg received VariableSizeMessage{price=0.07065803512640578, volume=310798300, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
Read MsgSize 40
Msg received VariableSizeMessage{price=0.6410843215904908, volume=1838316515, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
Read MsgSize 40
Msg received VariableSizeMessage{price=0.35616341133175833, volume=640417672, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
Read MsgSize 40
Msg received VariableSizeMessage{price=0.02791475958926104, volume=920905018, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
Read MsgSize 40
Msg received VariableSizeMessage{price=0.22370939556177194, volume=125809507, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
Read MsgSize 40
Msg received VariableSizeMessage{price=0.15608589096032743, volume=953519745, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
Read MsgSize 40
Msg received VariableSizeMessage{price=0.13704370458866222, volume=2143473156, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
Read MsgSize 40
Msg received VariableSizeMessage{price=0.6469131423152766, volume=1475089579, instrument='INST(8){8}{8}{8}{8}{8}{8}{8}' }
```


3. Mensajes de tamaño variable grandes

Lo anterior no sucede cuando los paquetes superan cierto tamaño, ya que se llena el buffer del túnel de comunicación, y el cliente se queda a la espera de la llegada de todo el mensaje para leerlo de una vez. Como el túnel no puede albergar más, y el cliente no va a leer nada hasta conseguir el mensaje entero, se queda atascado en un bucle infinito.

Tal y como indica el ejercicio, si modificamos el tamaño de paquete desde los 8 a los 80000, la esta casuística sucederá.



Como se puede ver, aunque el servidor sigue mandando mensajes, el cliente se ha quedado pillado al intentar leer un mensaje muy grande, por lo que aunque se sigan mandando mensajes, el canal de comunicación pondrá los mensajes a la cola y permanecerá saturado.

Para evitar esto, lo que se hará es lo siguiente: se enviará el mensaje entero, pero se irá leyendo por partes, asegurándose de que lleguen pequeñas partes y vaciando el canal para futuros fragmentos.



Aunque las imágenes no varían mucho, se ve que el mensaje de espera al mensaje después de leer el header no aparece, indicando que se ha leído todo el mensaje.

Práctica 2: Creación de comunicaciones multicast

1. Mensajes de tamaño variable

A diferencia de TCP, usando UDP multicast se puede ejecutar tanto el cliente como el servidor en el orden deseado. El servidor estará mandando mensajes constantemente, y cuando alguien se suscriba a la red leerá lo que haya mandado, por eso la cantidad de mensajes leídos y enviados difiere en la captura.


```
sampru@debian:/media/sampru/Datuak/eskola/tec-sec-fin/MasterUC3Practices/Lesson2$ mvn exec:java -Dexec.mainClass="com.cnebrera.uc3.tech.lesson2.mcast.McastServer"  
[INFO] Scanning for projects...  
[INFO]  
  
[INFO] -----  
[INFO] Building Lesson2 1.0-SNAPSHOT  
[INFO] -----  
  
[INFO]  
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Lesson2 ---  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
About to send message datagram of size 40  
^Csampru@debian:/media/sampru/Datuak/eskola/tec-sec-fin/MasterUC3Practices/Lesson2$
```

```

18, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
Msg received VariableSizeMessage(price=0.4474234292877326, volume=96070888
, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
Msg received VariableSizeMessage(price=0.4633217055158517, volume=16299635
64, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
Msg received VariableSizeMessage(price=0.9838678422070398, volume=12848749
12, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
^Csampru@debian:/media/sampru/Datuak/eskola/tec-sec-fin/MasterUC3Practices
esson2$ 
[INFO] -----
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Lesson2 ---
Msg received VariableSizeMessage(price=0.4055451052713305, volume=10923861
62, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
Msg received VariableSizeMessage(price=0.8835920277614994, volume=71053557
8, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
]
Msg received VariableSizeMessage(price=0.03225837736463866, volume=4561104
20, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
Msg received VariableSizeMessage(price=0.27641126584308706, volume=184733
1154, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
Msg received VariableSizeMessage(price=0.4055451052713305, volume=10923861
62, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
Msg received VariableSizeMessage(price=0.8835920277614994, volume=71053557
8, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
]
Msg received VariableSizeMessage(price=0.03225837736463866, volume=4561104
20, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
Msg received VariableSizeMessage(price=0.27641126584308706, volume=184733
1154, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
Msg received VariableSizeMessage(price=0.4055451052713305, volume=10923861
62, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
Msg received VariableSizeMessage(price=0.8835920277614994, volume=71053557
8, instrument='INST(8)(8)(8)(8)(8)(8)(8)(8)'
]

```

Como se puede observar, aunque el servidor esté mandando información constante, en el momento que se conecte el cliente será cuando empiece a recibirlos. En caso de parar el cliente, el servidor seguirá mandando mensajes. En caso de parar el servidor, el cliente se quedará a la espera de nuevos mensajes en la IP suscrita. Esto también permite la opción de conectarse varios clientes, al no ser una comunicación punto a punto.

2. Mensajes de tamaño variable grande

Tal y como indica el ejercicio, la gestión de tramas debe hacerse por parte de usuario, así que si no se ha regulado manualmente, el servidor no será capaz de mandar mensajes de grandes dimensiones y fallará al intentarlo.

```
[INFO] -----
[INFO] Building Lesson2 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Lesson2 ---
About to send message datagram of size 560016
[WARNING]
java.io.IOException: Mensaje demasiado largo (sendto failed)
    at java.net.PlainDatagramSocketImpl.send(Native Method)
    at java.net.DatagramSocket.send(DatagramSocket.java:693)
    at com.cnebrera.uc3.tech.lesson2.mcast.McastServer.sendMessage(McastServer.java:59)
    at com.cnebrera.uc3.tech.lesson2.mcast.McastServer.main(McastServer.java:31)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.codehaus.mojo.exec.ExecJavaMojo$1.run(ExecJavaMojo.java:282)
    at java.lang.Thread.run(Thread.java:748)
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO]
[INFO] Total time: 0.913 s
[INFO] Finished at: 2018-11-04T19:55:13+01:00
[INFO] Final Memory: 10M/218M
[INFO] -----
```

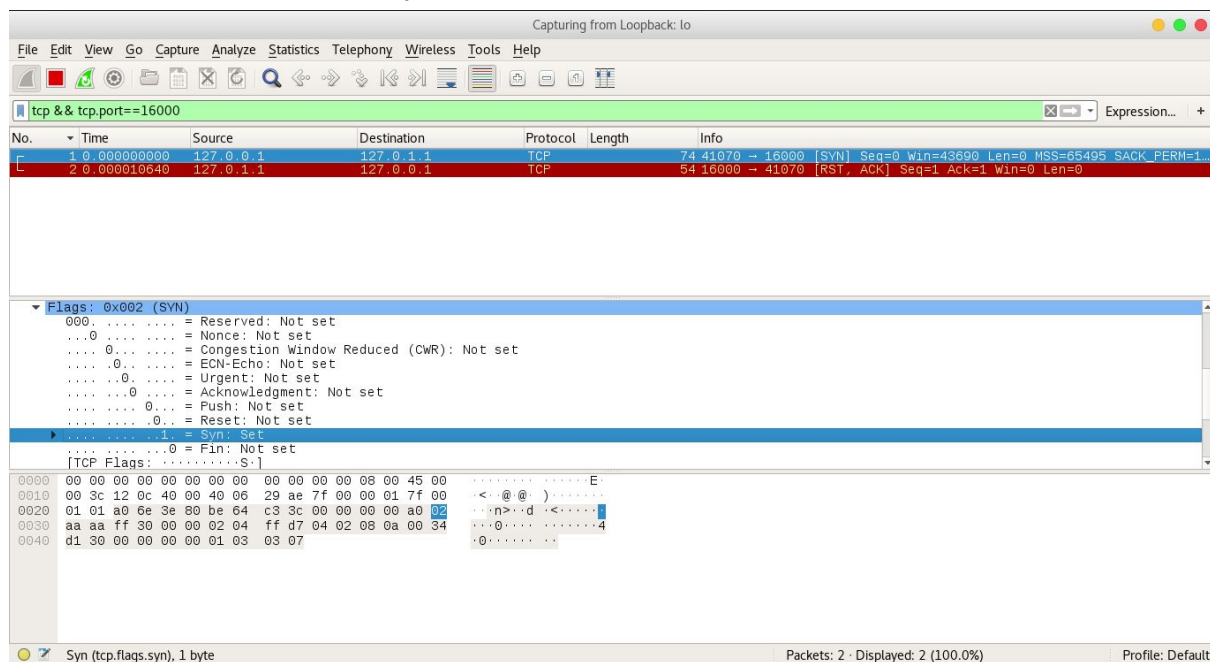
```
Lesson2$ mvn exec:java -Dexec.mainClass="com.cnebrera.uc3.tech.lesson2.mcas
r.McastClient"
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Lesson2 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Lesson2 ---
[
```

Práctica 3: Analizando con Wireshark

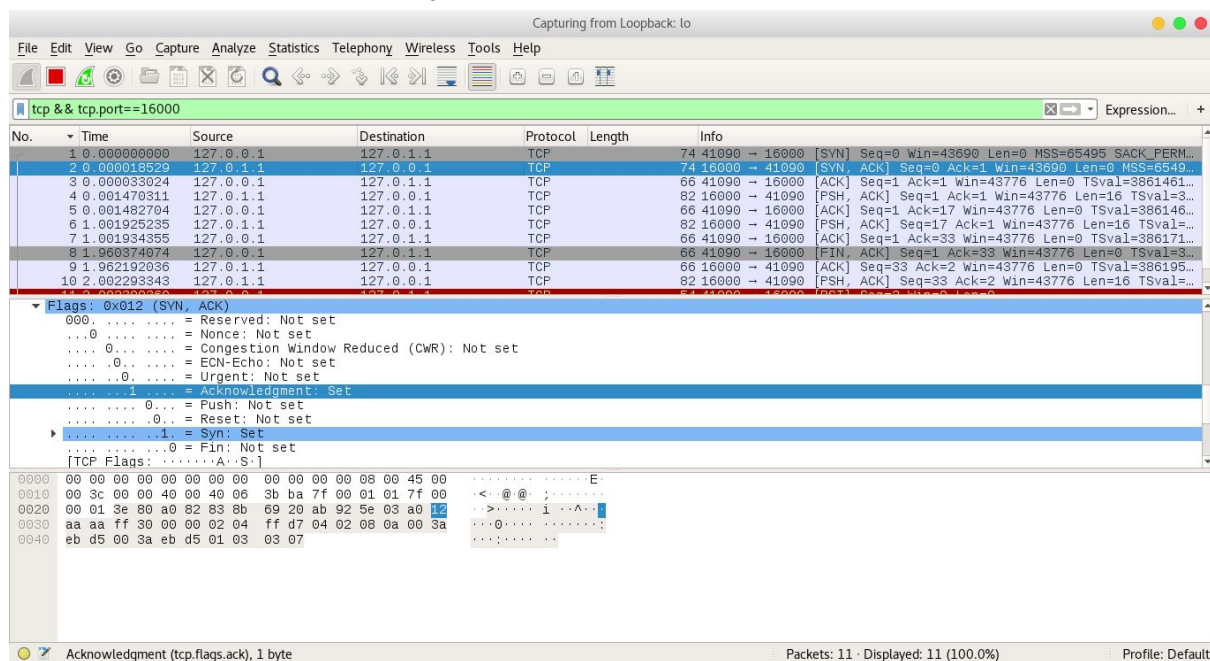
1. Flujo TCP normal

Si abrimos únicamente el cliente, fallará porque no será posible abrir el socket solicitado. El sistema responderá con un rechazo.

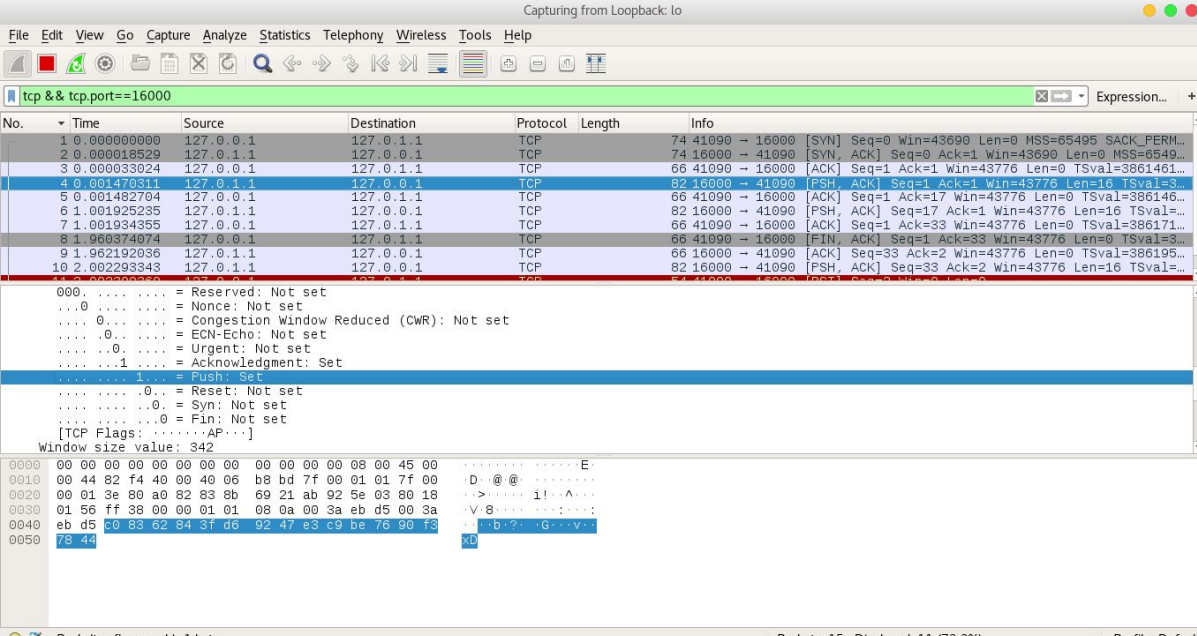
Se verán un paquete de SYN y un RST avisando de que no ha sido posible.



Si se lanza con el servidor en marcha, empezará a generar tráfico. Aquí se observa que en vez de un RST, recibe un mensaje de ACK para confirmar que se ha establecido el socket.



Sin embargo, y debido al medio de transporte por bytes que se ha elegido, no se puede leer nada de lo que viaje en el mensaje, ya que está en binario.



Wireshark packet capture showing a TCP push message. The packet list shows a PSH flag set. The packet details show the TCP flags and the push data. The packet bytes show the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	41090 → 16000 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM...
2	0.000018529	127.0.0.1	127.0.0.1	TCP	74	16000 → 41090 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=6549...
3	0.000033024	127.0.0.1	127.0.0.1	TCP	66	41090 → 16000 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=3861461...
4	0.001470311	127.0.0.1	127.0.0.1	TCP	82	16000 → 41090 [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=16 TSval=3...
5	0.001482704	127.0.0.1	127.0.0.1	TCP	66	41090 → 16000 [ACK] Seq=1 Ack=17 Win=43776 Len=0 TSval=386146...
6	1.001925235	127.0.0.1	127.0.0.1	TCP	82	16000 → 41090 [PSH, ACK] Seq=17 Ack=1 Win=43776 Len=16 TSval=...
7	1.001934355	127.0.0.1	127.0.0.1	TCP	66	41090 → 16000 [ACK] Seq=1 Ack=33 Win=43776 Len=0 TSval=386171...
8	1.969374074	127.0.0.1	127.0.0.1	TCP	66	41090 → 16000 [FIN, ACK] Seq=1 Ack=33 Win=43776 Len=0 TSval=3...
9	1.962192036	127.0.0.1	127.0.0.1	TCP	66	16000 → 41090 [ACK] Seq=33 Ack=2 Win=43776 Len=0 TSval=386195...
10	2.002293343	127.0.0.1	127.0.0.1	TCP	82	16000 → 41090 [PSH, ACK] Seq=33 Ack=2 Win=43776 Len=16 TSval=...

000. = Reserved: Not set
...0 = Nonce: Not set
...0 = Congestion Window Reduced (CWR): Not set
...0 = ECN-Echo: Not set
...0 = Urgent: Not set
...1 = Acknowledgment: Set
...1 = Push: Set
...0 = Reset: Not set
...0 = Syn: Not set
...0 = Fin: Not set
[TCP Flags:AP...]
Window size value: 342

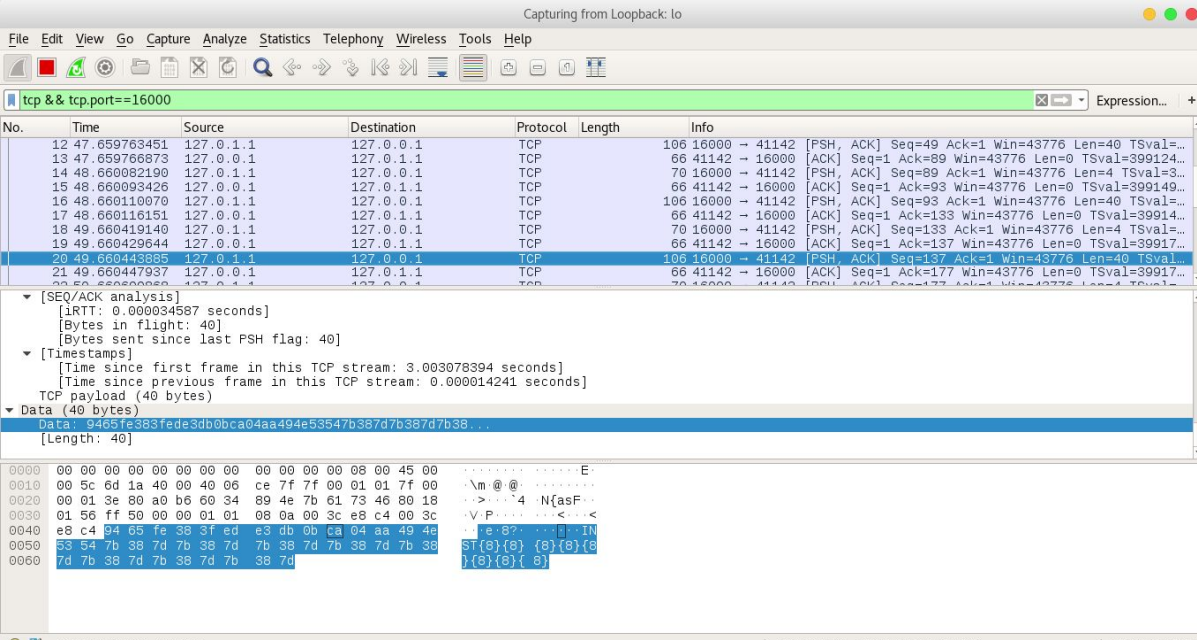
0000 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00E:
0010 00 44 82 f4 40 00 40 06 b8 bd 7f 00 01 01 7f 00 ..D.@@.....
0020 00 01 3e 80 a0 b6 60 34 89 4e 7b 61 73 46 80 18 ..>...1!-A-..
0030 01 56 ff 38 00 00 01 01 08 0a 00 3a eb d5 00 3a ..V.B.....
0040 eb d5 00 83 62 84 3f d6 92 47 e3 c9 be 76 90 76 ..b?..G...v
0050 78 44 ..x

Push (tcp.flags.push), 1 byte

Packets: 15 · Displayed: 11 (73.3%)

Profile: Default

En cambio, cuando pasamos a los mensajes variables que se mandan en formato “legible”, Wireshark es capaz de interpretar los datos y mostrarlos en pantalla.



Wireshark packet capture showing a TCP push message with a data payload. The packet list shows a PSH flag set. The packet details show the TCP flags and the push data. The packet bytes show the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
12	47.659763451	127.0.0.1	127.0.0.1	TCP	106	16000 → 41142 [PSH, ACK] Seq=49 Ack=1 Win=43776 Len=40 TSval=...
13	47.659766873	127.0.0.1	127.0.0.1	TCP	66	41142 → 16000 [ACK] Seq=1 Ack=89 Win=43776 Len=0 TSval=399124...
14	48.660082190	127.0.0.1	127.0.0.1	TCP	70	16000 → 41142 [PSH, ACK] Seq=89 Ack=1 Win=43776 Len=4 TSval=...
15	48.660093426	127.0.0.1	127.0.0.1	TCP	66	41142 → 16000 [ACK] Seq=1 Ack=93 Win=43776 Len=0 TSval=399149...
16	48.660110070	127.0.0.1	127.0.0.1	TCP	106	16000 → 41142 [PSH, ACK] Seq=93 Ack=1 Win=43776 Len=40 TSval=...
17	48.660116151	127.0.0.1	127.0.0.1	TCP	66	41142 → 16000 [ACK] Seq=1 Ack=133 Win=43776 Len=0 TSval=39914...
18	49.660419140	127.0.0.1	127.0.0.1	TCP	70	16000 → 41142 [PSH, ACK] Seq=133 Ack=1 Win=43776 Len=4 TSval=...
19	49.660429644	127.0.0.1	127.0.0.1	TCP	66	41142 → 16000 [ACK] Seq=1 Ack=137 Win=43776 Len=0 TSval=39917...
20	49.660443885	127.0.0.1	127.0.0.1	TCP	106	16000 → 41142 [PSH, ACK] Seq=137 Ack=1 Win=43776 Len=40 TSval=...
21	49.660447937	127.0.0.1	127.0.0.1	TCP	66	41142 → 16000 [ACK] Seq=1 Ack=177 Win=43776 Len=0 TSval=39917...

[SEQ/ACK analysis]
[RTT: 0.000034587 seconds]
[Bytes in flight: 40]
[Bytes sent since last PSH flag: 40]
[Timestamps]
[Time since first frame in this TCP stream: 3.003078394 seconds]
[Time since previous frame in this TCP stream: 0.000014241 seconds]
TCP payload (40 bytes)
Data (40 bytes)
Data: 9465fe383fede3db0bca04aa494e53547b387d7b387d7b38...
[Length: 40]

0000 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00E:
0010 00 5c 6d 1a 40 00 40 06 ce 7f 7f 00 01 01 7f 00 ..\m.@@.....
0020 00 01 3e 80 a0 b6 60 34 89 4e 7b 61 73 46 80 18 ..>...4-N(asF..
0030 01 56 ff 50 00 00 01 01 08 0a 00 3c e8 c4 00 3c ..V.P.....<..
0040 e8 c4 04 65 fe 38 3f ed e3 db 0b ca 04 aa 49 4e ..e.8?....IN..
0050 53 54 7b 38 7d 7b 38 7d 7b 38 7d 7b 38 7d 7b 38 ..S({}{ }({}{ }({}{ }
0060 7d 7b 38 7d 7b 38 7d 7b 38 7d ..}{ }({}{ }{ }

Data (data.data), 40 bytes

Packets: 37 · Displayed: 35 (94.6%)

Profile: Default

2. Flujo TCP bloqueado

En la tercera casuística analizada en los ejercicios anteriores, se ha visto que el cliente entra en un loop infinito y se queda bloqueado, se llena la ventana de comunicaciones y no se consigue mandar nada mas. Aquí vemos los mensajes de control que se intercambian entre cliente y servidor, avisando de la ventana llena y de la imposibilidad de lectura por parte del cliente. También se pueden observar mensajes de keep-alive para avisar de que la comunicación sigue en marcha.

The image shows a Wireshark packet capture window titled "Capturing from Loopback: lo". The filter bar shows "tcp && tcp.port==16000". The packet list shows several TCP packets, including Keep-Alive and ZeroWindow messages. The packet details pane shows the selected packet (No. 222) as an Ethernet II frame, followed by an Internet Protocol Version 4 packet, and a TCP packet. The TCP packet details show a "ZeroWindow" flag, indicating that the receiver's buffer is full and no data can be sent.

No.	Time	Source	Destination	Protocol	Length	Info
220	23.514896623	127.0.0.1	127.0.0.1	TCP	65549	16000 → 41702 [ACK] Seq=4938652 Ack=1 Win=43776 Len=65483 TSv...
221	23.558794257	127.0.0.1	127.0.0.1	TCP	66	41702 → 16000 [ACK] Seq=1 Ack=5004135 Win=14336 Len=0 TSval=4...
222	23.786803178	127.0.0.1	127.0.0.1	TCP	14402	[TCP window Full] 16000 → 41702 [PSH, ACK] Seq=5004135 Ack=1 ...
223	23.786822707	127.0.0.1	127.0.0.1	TCP	66	[TCP ZeroWindow] 41702 → 16000 [ACK] Seq=1 Ack=5018471 Win=0 ...
224	24.010798133	127.0.0.1	127.0.0.1	TCP	66	[TCP Keep-Alive] 16000 → 41702 [ACK] Seq=5018470 Ack=1 Win=43...
225	24.010811174	127.0.0.1	127.0.0.1	TCP	66	[TCP ZeroWindow] 41702 → 16000 [ACK] Seq=1 Ack=5018471 Win=0 ...
226	24.482795555	127.0.0.1	127.0.0.1	TCP	66	[TCP Keep-Alive] 16000 → 41702 [ACK] Seq=5018470 Ack=1 Win=43...
227	25.378795634	127.0.0.1	127.0.0.1	TCP	66	[TCP Keep-Alive] 16000 → 41702 [ACK] Seq=5018470 Ack=1 Win=43...
228	25.378810578	127.0.0.1	127.0.0.1	TCP	66	[TCP ZeroWindow] 41702 → 16000 [ACK] Seq=1 Ack=5018471 Win=0 ...
229	27.174792655	127.0.0.1	127.0.0.1	TCP	66	[TCP Keep-Alive] 16000 → 41702 [ACK] Seq=5018470 Ack=1 Win=43...

Frame 222: 14402 bytes on wire (115216 bits), 14402 bytes captured (115216 bits) on interface 0
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
0100 = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
0000 00.. = Differentiated Services Codepoint: Default (0)
.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 14388
Identification: 0x0bc9 (3017)
Flags: 0x4000, Don't fragment
0..... = Reserved bit: Not set

Ethernet (eth), 14 bytes

Packets: 360 · Displayed: 211 (58.6%)

Profile: Default

3. Flujo Multicast

Al usar UPC, todos los mensajes de control y comunicación se dejan de lado, mostrando únicamente los paquetes de mensajes que manda el servidor. Estos paquetes son los que interpreta el cliente, y podemos ver el string de la cadena que se manda.

The image shows a Wireshark packet capture analysis of a UDP packet. The packet list at the top shows a UDP packet from 192.168.0.21 to 224.0.0.3, port 37340 to 8888. The packet details pane shows the Ethernet II header, Internet Protocol Version 4 header, and User Datagram Protocol header. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
81	19.279729981	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40
85	20.280036464	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40
90	21.280307989	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40
94	22.280826953	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40
97	23.281571148	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40
102	24.282150674	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40
123	25.282700243	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40
125	26.283309294	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40
127	27.283883681	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40
129	28.284409465	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40
131	29.284909655	192.168.0.21	224.0.0.3	UDP	82	82 37340 → 8888 Len=40

Source: 192.168.0.21
Destination: 224.0.0.3

User Datagram Protocol, Src Port: 37340, Dst Port: 8888

Source Port: 37340
Destination Port: 8888
Length: 40
Checksum: 0x19cb [unverified]
[Checksum Status: Unverified]
[Stream index: 11]

Data (40 bytes)

0000 81 00 5e 00 00 03 40 f0 2f e5 df 1d 08 00 45 00 ...
0010 00 44 2f e8 40 00 01 11 a9 00 c0 a8 00 15 e0 00 ...
0020 00 03 91 dc 22 b8 00 30 19 bc 80 2d 33 7e 3f e5 ...
0030 67 52 9a 08 3a 1a 49 4e 53 54 7b 38 7d 7b 38 7d ...
0040 7b 38 7d 7b 38 7d 7b 38 7d 7b 38 7d 7b 38 7d 7b ...
0050 38 7d

Data (data.data), 40 bytes

Packets: 329 · Displayed: 15 (4.6%)

Profile: Default

Como se puede observar, el servidor, alojado en la red local, manda paquetes a la dirección multicast establecida, habilitando a toda la red para capturar esos mensajes.