David LE, Mehdi BOULHARTS, 3A RTS
Supervisor: Linda SENIGAGLIESI

TP2 – Java for networks

1 – Creating a UDP Client-Server

1.1 UDP Server

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out UDPServer 8080
UDPServer{port=8080, running=true}
[127.0.0.1:50946] I am the client
[127.0.0.1:50946] My name is Mehdi
[127.0.0.1:55962] hello

[127.0.0.1:58843] hello2
```

1.2 UDP Client

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out UDPClient localhost 8080
Tape des lignes et valide. Fin avec Ctrl+D (mac/linux) ou Ctrl+Z puis Entrée (Windows).
I am the client
My name is Mehdi
```

With netcat command :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % printf 'hello\n' | nc -u -w1 127.0.0.1 8080
```

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % printf 'hello2\n' | nc -u -w 1 127.0.0.1 8080
```

We can see that our server is operational and can receive messages from clients.

2 – Additional

2.1 UDP Reliability Challenges

Server :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out ReliableUDPServer 8081 0.2
ReliableUDPServer{port=8081, dropRate=0,20}
[/127.0.0.1:55042] seq=1 msg=msg-1-554
[/127.0.0.1:55042] seq=2 msg=msg-2-712
[/127.0.0.1:55042] seq=3 msg=msg-3-437
```

.
.
.

```
[/127.0.0.1:55042] seq=198 msg=msg-198-166
[/127.0.0.1:55042] seq=199 msg=msg-199-808
(sim drop) from /127.0.0.1:55042 seq=200
(sim drop) from /127.0.0.1:55042 seq=200
(sim drop) from /127.0.0.1:55042 seq=200
[/127.0.0.1:55042] seq=200 msg=msg-200-530
Stats: received=243, duplicates=0, simDropped=43
```

Client :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out ReliableUDPClient localhost 8081 200
Summary: sent=200, acked=200, failed(no-ACK)=0, retries≈0, loss≈0,00%
```

There are 200 messages transmitted from the client to the server with this command.

2.2 Datagram Packet Analysis

len(buffer) = 64 bytes

Server :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out UDPBufferServer 8082 64
UDPBufferServer{port=8082, buffer=64}
recv len=64 (buffer=64) [POSSIBLE TRUNCATION] from /127.0.0.1:61537
recv len=64 (buffer=64) [POSSIBLE TRUNCATION] from /127.0.0.1:55631
```

Client :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out PacketSizeClient 127.0.0.1 8082 64
sent 64 bytes to 127.0.0.1:8082
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out PacketSizeClient 127.0.0.1 8082 2048
sent 2048 bytes to 127.0.0.1:8082
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out PacketSizeClient 127.0.0.1 8082 70000
Exception in thread "main" java.net.SocketException: Message too long
        at java.base/sun.nio.ch.DatagramChannelImpl.send0(Native Method)
        at java.base/sun.nio.ch.DatagramChannelImpl.sendFromNativeBuffer(DatagramChannelImpl.java:1016)
        at java.base/sun.nio.ch.DatagramChannelImpl.send(DatagramChannelImpl.java:973)
        at java.base/sun.nio.ch.DatagramChannelImpl.send(DatagramChannelImpl.java:896)
        at java.base/sun.nio.ch.DatagramChannelImpl.blockingSend(DatagramChannelImpl.java:959)
        at java.base/sun.nio.ch.DatagramSocketAdaptor.send(DatagramSocketAdaptor.java:193)
        at java.base/java.net.DatagramSocket.send(DatagramSocket.java:667)
        at PacketSizeClient.main(PacketSizeClient.java:17)
```

len(buffer) = 2048 bytes

Server :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out UDPBufferServer 8082 2048
UDPBufferServer{port=8082, buffer=2048}
recv len=2048 (buffer=2048) [POSSIBLE TRUNCATION] from /127.0.0.1:49890
recv len=64 (buffer=2048) from /127.0.0.1:63119
```

Client :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out PacketSizeClient 127.0.0.1 8082 70000
Exception in thread "main" java.net.SocketException: Message too long
        at java.base/sun.nio.ch.DatagramChannelImpl.send0(Native Method)
        at java.base/sun.nio.ch.DatagramChannelImpl.sendFromNativeBuffer(DatagramChannelImpl.java:1016)
        at java.base/sun.nio.ch.DatagramChannelImpl.send(DatagramChannelImpl.java:973)
        at java.base/sun.nio.ch.DatagramChannelImpl.send(DatagramChannelImpl.java:896)
        at java.base/sun.nio.ch.DatagramChannelImpl.blockingSend(DatagramChannelImpl.java:959)
        at java.base/sun.nio.ch.DatagramSocketAdaptor.send(DatagramSocketAdaptor.java:193)
        at java.base/java.net.DatagramSocket.send(DatagramSocket.java:667)
        at PacketSizeClient.main(PacketSizeClient.java:17)
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out PacketSizeClient 127.0.0.1 8082 2048
sent 2048 bytes to 127.0.0.1:8082
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out PacketSizeClient 127.0.0.1 8082 64
sent 64 bytes to 127.0.0.1:8082
```

In both cases, when the message from the client is too long, he is not transmitted to the server. The message can be truncated if he is short enough as we can see in the first example with 2048 bytes transmitted in packs of 64 bytes.

2.3 Connectionless Communication Demo

Try without launching server :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % printf 'cmd nc\n' | nc -u -w1 127.0.0.1 8080
read(net): Connection refused
```

```
15 3.491275     127.0.0.1          127.0.0.1          ICMP      60 Destination unreachable (Port unreachable)
```

No connection UDP without server (logic).

We launched 2 servers and then connected with client :
Servers :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out UDPServer 8080
UDPServer{port=8080, running=true}
[127.0.0.1:49304] cmd nc to 8080
```

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out UDPServer 8083
UDPServer{port=8083, running=true}
[127.0.0.1:53121] cmd nc to 8083
```

Client :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % printf 'cmd nc to 8080\n' | nc -u -w1 127.0.0.1 8080
^C%
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % printf 'cmd nc to 8083\n' | nc -u -w1 127.0.0.1 8083
```

In Wireshark :

```
1  0.000000     127.0.0.1          127.0.0.1          UDP       47 49304 → 8080 Len=15
18 18.531015    127.0.0.1          127.0.0.1          UDP       47 53121 → 8083 Len=15
```

So no “connection state” is shared, each datagram is routed independently to the target port.

Here we launched server after client begin to send 200 messages at this this one :

Server :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out ReliableUDPServer 8080 0.2
ReliableUDPServer{port=8080, dropRate=0,20}
[/127.0.0.1:58307] seq=7 msg=msg-7-228
[/127.0.0.1:58307] seq=8 msg=msg-8-942
[/127.0.0.1:58307] seq=9 msg=msg-9-273
(sim drop) from /127.0.0.1:58307 seq=10
[/127.0.0.1:58307] seq=10 msg=msg-10-390
[/127.0.0.1:58307] seq=11 msg=msg-11-790
```

```
[/127.0.0.1:58307] seq=71 msg=msg-71-364
[/127.0.0.1:58307] seq=72 msg=msg-72-404
(sim drop) from /127.0.0.1:58307 seq=73
(sim drop) from /127.0.0.1:58307 seq=73
^C%
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out ReliableUDPServer 8080 0.2
ReliableUDPServer{port=8080, dropRate=0,20}
[/127.0.0.1:58307] seq=80 msg=msg-80-422
(sim drop) from /127.0.0.1:58307 seq=81
[/127.0.0.1:58307] seq=81 msg=msg-81-617
[/127.0.0.1:58307] seq=82 msg=msg-82-995
```

```
[/127.0.0.1:58307] seq=196 msg=msg-196-799
[/127.0.0.1:58307] seq=197 msg=msg-197-282
[/127.0.0.1:58307] seq=198 msg=msg-198-610
[/127.0.0.1:58307] seq=199 msg=msg-199-541
(sim drop) from /127.0.0.1:58307 seq=200
[/127.0.0.1:58307] seq=200 msg=msg-200-276
Stats: received=141, duplicates=0, simDropped=20
```

Client :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out ReliableUDPClient localhost 8080 200
Summary: sent=200, acked=187, failed(no-ACK)=13, retries≈0, loss≈6,50%
```

In Wireshark :

62	5.846830	127.0.0.1	127.0.0.1	ICMP	60 Destination unreachable (Port unreachable)
63	6.048798	127.0.0.1	127.0.0.1	UDP	46 58307 → 8080 Len=14
64	6.048869	127.0.0.1	127.0.0.1	ICMP	60 Destination unreachable (Port unreachable)
65	6.250273	127.0.0.1	127.0.0.1	UDP	46 58307 → 8080 Len=14
66	6.250335	127.0.0.1	127.0.0.1	ICMP	60 Destination unreachable (Port unreachable)
67	6.451671	127.0.0.1	127.0.0.1	UDP	46 58307 → 8080 Len=14
68	6.454041	127.0.0.1	127.0.0.1	UDP	37 8080 → 58307 Len=5
69	6.454686	127.0.0.1	127.0.0.1	UDP	46 58307 → 8080 Len=14
70	6.455265	127.0.0.1	127.0.0.1	UDP	37 8080 → 58307 Len=5
71	6.455516	127.0.0.1	127.0.0.1	UDP	46 58307 → 8080 Len=14
72	6.456113	127.0.0.1	127.0.0.1	UDP	37 8080 → 58307 Len=5

215	9.737004	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15
216	9.938668	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15
217	10.140408	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15
218	10.342112	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15
219	10.342174	127.0.0.1	127.0.0.1	ICMP	60 Destination unreachable (Port unreachable)
220	10.543867	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15
221	10.543925	127.0.0.1	127.0.0.1	ICMP	60 Destination unreachable (Port unreachable)
222	10.745826	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15
223	10.745881	127.0.0.1	127.0.0.1	ICMP	60 Destination unreachable (Port unreachable)

285	16.392369	127.0.0.1	127.0.0.1	ICMP	60 Destination unreachable (Port unreachable)
286	16.593958	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15
287	16.594012	127.0.0.1	127.0.0.1	ICMP	60 Destination unreachable (Port unreachable)
288	16.795219	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15
289	16.795265	127.0.0.1	127.0.0.1	ICMP	60 Destination unreachable (Port unreachable)
290	16.996774	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15
291	17.000857	127.0.0.1	127.0.0.1	UDP	37 8080 → 58307 Len=5
292	17.001357	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15
293	17.202835	127.0.0.1	127.0.0.1	UDP	47 58307 → 8080 Len=15

So no negotiation or state recovery, the server does not maintain connections with clients using UDP.

2.4 UDP Multicast Exploration

Server :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out UDPMulticastReceiver 230.0.0.1 8888
Joined multicast group 230.0.0.1:8888
[multicast recv from /10.10.24.227:56820] hi everyone
[multicast recv from /10.10.24.227:56820] hi everyone
```

Client :

```
(base) mehdi@MacBook-Air-de-Mehdi tp-chat-udp % java -cp out UDPMulticastSender 230.0.0.1 8888 "hi everyone"
Sent multicast 'hi everyone' to 230.0.0.1:8888
```

In Wireshark :

1	0.000000	10.10.24.227	230.0.0.1	UDP	43	53321 → 8888 Len=11

We can see that one message is sent to a group of sockets and so we have multiple receivers (all sockets initialized).

Conclusion :
UDP is connectionless: there is no handshake or persistent server state. Datagrams sent before the server starts (or while it’s down) are simply lost, and each packet is routed independently to its target IP:port (no “session”). Starting/stopping the server doesn’t restore anything, only packets sent while it’s listening are observed.