

Práctica Bloque II

Alumno 1: Apellidos, Nombre: García Fernández, Isidro Javier

Titulación: Doble grado Ingeniería Informática y Matemáticas (Grupo D)

PC de la práctica: 012

Parte I: Protocolo UDP

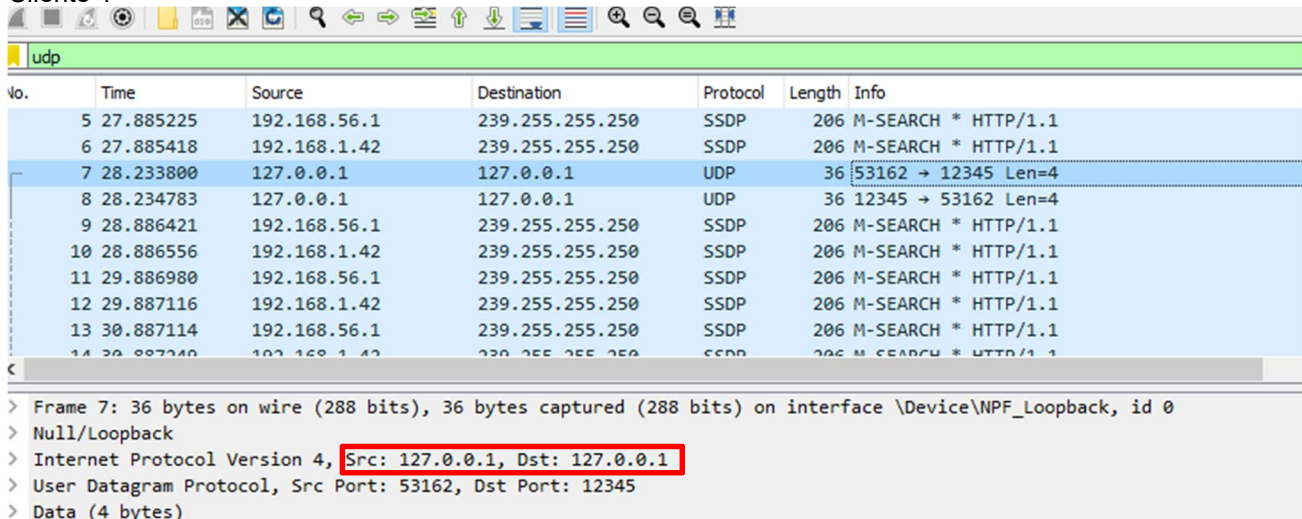
Recuerde que debe añadir una explicación del código (incluyendo todas las sentencias relacionadas con los sockets). Puede añadir esta explicación como comentarios en el código.

Usando la traza UDP1 (b2e1-3.pcapng):

Ejercicio 1.

- ¿Cuál es el puerto que usa el cliente?
Cliente 1: 53162
Cliente 2: 53163
- ¿Y el servidor? 12345
- ¿Qué tipo de puerto es cada uno de ellos?
Cuando el cliente envía, el cliente es Source Port y el servidor es Dst (Destinatario)
Cuando el cliente recibe, el cliente es Dst (Destinatario) y el servidor es Source Port

Cliente 1



No.	Time	Source	Destination	Protocol	Length	Info
5	27.885225	192.168.56.1	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
6	27.885418	192.168.1.42	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
7	28.233800	127.0.0.1	127.0.0.1	UDP	36	53162 → 12345 Len=4
8	28.234783	127.0.0.1	127.0.0.1	UDP	36	12345 → 53162 Len=4
9	28.886421	192.168.56.1	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
10	28.886556	192.168.1.42	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
11	29.886980	192.168.56.1	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
12	29.887116	192.168.1.42	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
13	30.887114	192.168.56.1	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
14	30.887249	192.168.1.42	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1

> Frame 7: 36 bytes on wire (288 bits), 36 bytes captured (288 bits) on interface \Device\NPF_{Loopback}, id 0

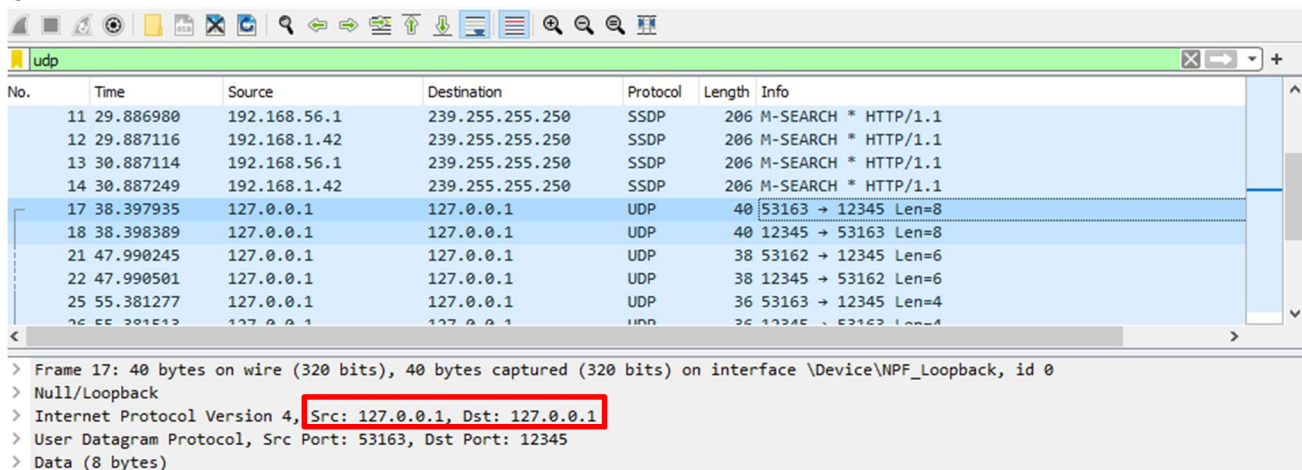
> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> User Datagram Protocol, Src Port: 53162, Dst Port: 12345

> Data (4 bytes)

Cliente 2



No.	Time	Source	Destination	Protocol	Length	Info
11	29.886980	192.168.56.1	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
12	29.887116	192.168.1.42	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
13	30.887114	192.168.56.1	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
14	30.887249	192.168.1.42	239.255.255.250	SSDP	206	M-SEARCH * HTTP/1.1
17	38.397935	127.0.0.1	127.0.0.1	UDP	40	53163 → 12345 Len=8
18	38.398389	127.0.0.1	127.0.0.1	UDP	40	12345 → 53163 Len=8
21	47.990245	127.0.0.1	127.0.0.1	UDP	38	53162 → 12345 Len=6
22	47.990501	127.0.0.1	127.0.0.1	UDP	38	12345 → 53162 Len=6
25	55.381277	127.0.0.1	127.0.0.1	UDP	36	53163 → 12345 Len=4
26	55.381513	127.0.0.1	127.0.0.1	UDP	36	12345 → 53163 Len=4

> Frame 17: 40 bytes on wire (320 bits), 40 bytes captured (320 bits) on interface \Device\NPF_{Loopback}, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> User Datagram Protocol, Src Port: 53163, Dst Port: 12345

> Data (8 bytes)

Salida por consola (Servidor):

```
ESTADO: Esperando recibir datos de cliente
Direccion socket del cliente: (IP): /127.0.0.1, (Puerto): 53162, (Datos): hola
ESTADO: Esperando recibir datos de cliente
Direccion socket del cliente: (IP): /127.0.0.1, (Puerto): 53163, (Datos): castanas
ESTADO: Esperando recibir datos de cliente
Direccion socket del cliente: (IP): /127.0.0.1, (Puerto): 53162, (Datos): verano
ESTADO: Esperando recibir datos de cliente
Direccion socket del cliente: (IP): /127.0.0.1, (Puerto): 53163, (Datos): mesa
ESTADO: Esperando recibir datos de cliente
Direccion socket del cliente: (IP): /127.0.0.1, (Puerto): 53162, (Datos): silla
ESTADO: Esperando recibir datos de cliente
Direccion socket del cliente: (IP): /127.0.0.1, (Puerto): 53163, (Datos): estandar
ESTADO: Esperando recibir datos de cliente
Direccion socket del cliente: (IP): /127.0.0.1, (Puerto): 53162, (Datos): canción
ESTADO: Esperando recibir datos de cliente
```

Salida por consola (Cliente1):

```
Introduzca un texto a enviar (END para acabar):
hola
STATUS: Waiting for the reply
Respuesta del servidor: aloh
Introduzca un texto a enviar (END para acabar):
verano
STATUS: Waiting for the reply
Respuesta del servidor: onarev
Introduzca un texto a enviar (END para acabar):
silla
STATUS: Waiting for the reply
Respuesta del servidor: allis
Introduzca un texto a enviar (END para acabar):
canción
STATUS: Waiting for the reply
Respuesta del servidor: n?icnac
Introduzca un texto a enviar (END para acabar):
END
STATUS: Closing client
STATUS: closed
```

Salida por consola (Cliente 2):

```
Introduzca un texto a enviar (END para acabar):
castanas
STATUS: Waiting for the reply
Respuesta del servidor: sanatsac
Introduzca un texto a enviar (END para acabar):
mesa
STATUS: Waiting for the reply
Respuesta del servidor: asem
Introduzca un texto a enviar (END para acabar):
estandarte
STATUS: Waiting for the reply
Respuesta del servidor: etradnatse
Introduzca un texto a enviar (END para acabar):
END
STATUS: Closing client
STATUS: closed
```

Ejercicio 2. Wireshark ofrece la opción de “Follow UDP stream”, pero en UDP no existe tal concepto.

- ¿Cómo es capaz Wireshark de decidir de un mensaje pertenece a un “flujo” u a otro?

Si observamos el puerto del emisor y del destinatario podemos identificar si el mensaje es enviado por/al cliente 1 o cliente 2.

Conversación cliente1 con servidor

Wireshark · Seguir flujo UDP (udp.stream eq 2) · b2e1-3.pcapng

holaalohveranoonarevsillaalliscanciññicnac

4 cliente pkts, 4 servidor pkts, 7 cambios.

Conversación completa (45 bytes) Show data as UTF-8 Secuencia 2

Buscar:

Conversación cliente2 con servidor

Wireshark · Seguir flujo UDP (udp.stream eq 3) · b2e1-3.pcapng

castanassanatsacmesaasemestandardteetradnatse

3 cliente pkts, 3 servidor pkts, 5 cambios.

Conversación completa (44 bytes) Show data as UTF-8 Secuencia 3

Buscar:

Ejercicio 3. Examine un mensaje que lleve tildes,

- ¿coincide el tamaño indicado en el campo longitud de la cabecera de UDP con la cantidad de letras enviadas?

udp.stream eq 2

No.	Time	Source	Destination	Protocol	Length	Info
7	28.233800	127.0.0.1	127.0.0.1	UDP	36	53162 → 12345 Len=4
8	28.234783	127.0.0.1	127.0.0.1	UDP	36	12345 → 53162 Len=4
21	47.990245	127.0.0.1	127.0.0.1	UDP	38	53162 → 12345 Len=6
22	47.990501	127.0.0.1	127.0.0.1	UDP	38	12345 → 53162 Len=6
28	67.744351	127.0.0.1	127.0.0.1	UDP	37	53162 → 12345 Len=5
29	67.744608	127.0.0.1	127.0.0.1	UDP	37	12345 → 53162 Len=5
33	85.100156	127.0.0.1	127.0.0.1	UDP	40	53162 → 12345 Len=8
34	85.100631	127.0.0.1	127.0.0.1	UDP	39	12345 → 53162 Len=7

> Frame 33: 40 bytes on wire (320 bits), 40 bytes captured (320 bits) on interface \Device\NPF_{...}, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> User Datagram Protocol, Src Port: 53162, Dst Port: 12345

> Data (8 bytes)

0000 02 00 00 00 45 00 00 24 4a 11 00 00 00 11 00 00E..\$ J.....
0010 7f 00 00 01 7f 00 00 01 cf aa 30 39 00 10 12 f109....
0020 63 61 6e 63 69 c3 b3 6ecanciññ

udp.stream eq 2

No.	Time	Source	Destination	Protocol	Length	Info
7	28.233800	127.0.0.1	127.0.0.1	UDP	36	53162 → 12345 Len=4
8	28.234783	127.0.0.1	127.0.0.1	UDP	36	12345 → 53162 Len=4
21	47.990245	127.0.0.1	127.0.0.1	UDP	38	53162 → 12345 Len=6
22	47.990501	127.0.0.1	127.0.0.1	UDP	38	12345 → 53162 Len=6
28	67.744351	127.0.0.1	127.0.0.1	UDP	37	53162 → 12345 Len=5
29	67.744608	127.0.0.1	127.0.0.1	UDP	37	12345 → 53162 Len=5
33	85.100156	127.0.0.1	127.0.0.1	UDP	40	53162 → 12345 Len=8
34	85.100631	127.0.0.1	127.0.0.1	UDP	39	12345 → 53162 Len=7

> Frame 34: 39 bytes on wire (312 bits), 39 bytes captured (312 bits) on interface \Device\NPF_{...}, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> User Datagram Protocol, Src Port: 12345, Dst Port: 53162

> Data (7 bytes)

0000 02 00 00 00 45 00 00 23 4a 12 00 00 00 11 00 00E..# J.....
0010 7f 00 00 01 7f 00 00 01 30 39 cf aa 00 0f 58 3109....X1
0020 6e f3 69 63 6e 61 63ñ-icnac

No.

- ¿Por qué?

Parece que los caracteres con tilde necesitan 2 bytes para representarse. (2 puntos)

Al enviar el mensaje de vuelta, el servidor lo envía con 1 bytes (1 punto). Es por ello que sale un carácter irreconocible.

Usando la traza UDP2 (b2e4.pcapng):

Ejercicio 4.

The screenshot shows the Wireshark interface with a packet capture of a UDP message. The packet list shows a UDP packet from 127.0.0.1 to 127.0.0.1 on port 60748. The packet details show the UDP header and the data field containing the text 'hola'.

- ¿Por qué consigue enviar si no hay ningún servidor activo?

En UDP no es necesaria una conexión previa entre cliente y servidor. Directamente el cliente envía el mensaje y espera respuesta.

- ¿Recibe alguna respuesta?

No. El servidor está inactivo. No puede realizar su función (en este caso, de invertir el mensaje).

- En caso afirmativo, indique qué significa esa respuesta y si es tratada o no.

No hay respuesta. El servidor está inactivo. El cliente se queda esperando una respuesta.

Sin traza:

Ejercicio 5. Asegure que captura la excepción de la creación del socket UDP y que muestra (método `getMessage()`) el error que se produce (modifique el código si no lo hacía). Intente abrir dos veces el servidor con los mismos parámetros,

- ¿qué error indica que se produce?

Address already in use: Cannot bind

```

28 DatagramSocket server = null;
29
30 /** COMPLETAR crear e inicializar el socket del servidor
31 try {
32     server = new DatagramSocket(port);
33 } catch (SocketException e) {
34     //System.err.println("Error: No se pudo escuchar al puerto" + port);
35     System.err.println(e.getMessage());
36     System.exit(1);
37 }
38 // Funcion PRINCIPAL del servidor
39 while (true)
40 {
41     System.out.println("ESTADO: Esperando recibir datos de cliente");
42     /** COMPLETAR: crear e inicializar un datagrama VACIO para recibir la
43     byte[] datosRecibidos = new byte[500];
44     DatagramPacket recepcion = new DatagramPacket(datosRecibidos, datosRecibidos.length);
45     /** COMPLETAR: Recibir datagrama
46     try {
47         server.receive(recepcion);
48     } catch (IOException e) {
49         System.err.println("Servidor no puede recibir el datagrama");
50     }
51 }

```

The screenshot shows a Java IDE with a code snippet for a UDP server. The code attempts to create a DatagramSocket on a specific port, but it throws an IOException with the message 'Address already in use: Cannot bind'.

- ¿Qué debería hacer para solucionar ese error y tener dos servidores del mismo tipo en su equipo?

El problema es que ambos servidores se inician con mismo puerto. Es por ello que el primero se ejecuta sin problema. En el momento en el que ejecutamos otro servidor con el mismo puerto, salta la excepción. Para poder abrir un nuevo servidor debemos asociarle un puerto distinto.

Parte II: Protocolo TCP

Recuerde que debe añadir una explicación del código (incluyendo todas las sentencias relacionadas con los sockets). Puede añadir esta explicación como comentarios en el código.

Usando la traza TCP1 (b2e6-9.pcapng):

Ejercicio 6. Identifique una trama de la comunicación y use la opción "Follow TCP stream" para ver el intercambio de información entre cliente y servidor.

- ¿Cuál es el puerto que usa el cliente?
Puerto Cliente: 61369
- ¿Y el servidor?
Puerto Servidor: 12345
- Muestra una captura de pantalla con dicha información.

Wireshark - Seguir flujo TCP (tcp.stream eq 0) · b2e6-9.pcapng

tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	61369 → 12345 [SYN, Seq=0 Win=65535
2	0.000068	127.0.0.1	127.0.0.1	TCP	56	12345 → 61369 [SYN, ACK] Seq=0 Ack=1
3	0.000105	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=1 Ack=1 Win=
4	0.004144	127.0.0.1	127.0.0.1	TCP	56	12345 → 61369 [PSH, ACK] Seq=1 Ack=1
5	0.004184	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=1 Ack=13 Win=
6	5.028182	127.0.0.1	127.0.0.1	TCP	52	61369 → 12345 [PSH, ACK] Seq=1 Ack=1
7	5.028219	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [ACK] Seq=13 Ack=9 Win=
8	5.028749	127.0.0.1	127.0.0.1	TCP	52	12345 → 61369 [PSH, ACK] Seq=13 Ack=
9	5.028785	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=9 Ack=21 Win=
10	6.778563	127.0.0.1	127.0.0.1	TCP	49	61369 → 12345 [PSH, ACK] Seq=9 Ack=2
11	6.778600	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [ACK] Seq=21 Ack=14 Wi
12	6.778877	127.0.0.1	127.0.0.1	TCP	50	12345 → 61369 [PSH, ACK] Seq=21 Ack=
13	6.778904	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=14 Ack=27 Wi
14	6.779878	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [FIN, ACK] Seq=27 Ack=
15	6.779904	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=14 Ack=28 Wi

> Frame 1: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{Loopback}, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 61369, Dst Port: 12345, Seq: 0, Len: 0

Ejercicio 7.

- ¿Cuál es el número de secuencia que se usa el cliente TCP hacia el servidor?

Número de secuencia de cliente a servidor: 595739486

tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	61369 → 12345 [SYN] Seq=0 Win=65535

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

✓ Transmission Control Protocol, Src Port: 61369, Dst Port: 12345, Seq: 0, Len: 0

Source Port: 61369

Destination Port: 12345

[Stream index: 0]

[TCP Segment Len: 0]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 595739486

[Next Sequence Number: 1 (relative sequence number)]

- ¿Y las respuestas del servidor al cliente?
- Número de secuencia de servidor a cliente: 4227358416

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	61369 → 12345 [SYN] Seq=0 Win=65535
2	0.000068	127.0.0.1	127.0.0.1	TCP	56	12345 → 61369 [SYN, ACK] Seq=0 Ack=1
3	0.000105	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=1 Ack=1 Win=
4	0.004144	127.0.0.1	127.0.0.1	TCP	56	12345 → 61369 [PSH, ACK] Seq=1 Ack=1
5	0.004184	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=1 Ack=13 Win=
6	5.028182	127.0.0.1	127.0.0.1	TCP	52	61369 → 12345 [PSH, ACK] Seq=1 Ack=1
7	5.028219	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [ACK] Seq=13 Ack=9 Win=
8	5.028749	127.0.0.1	127.0.0.1	TCP	52	12345 → 61369 [PSH, ACK] Seq=13 Ack=9
9	5.028785	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=9 Ack=21 Win=
10	6.778563	127.0.0.1	127.0.0.1	TCP	49	61369 → 12345 [PSH, ACK] Seq=9 Ack=21
11	6.778600	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [ACK] Seq=21 Ack=14 Win=
12	6.778877	127.0.0.1	127.0.0.1	TCP	50	12345 → 61369 [PSH, ACK] Seq=21 Ack=14
13	6.778904	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=14 Ack=27 Win=
14	6.779878	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [FIN, ACK] Seq=27 Ack=14
15	6.779904	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=14 Ack=28 Win=

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 12345, Dst Port: 61369, Seq: 0, Ack: 1, Len: 0

Source Port: 12345
Destination Port: 61369
[Stream index: 0]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 4227358416
[Next Sequence Number: 1 (relative sequence number)]

Ejercicio 8. Indique los segmentos relacionados con las siguientes actividades y qué métodos de Socket y ServerSocket son responsables del intercambio de estos segmentos:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	61369 → 12345 [SYN] Seq=0 Win=65535 Len=0
2	0.000068	127.0.0.1	127.0.0.1	TCP	56	12345 → 61369 [SYN, ACK] Seq=0 Ack=1 Win=
3	0.000105	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=1 Ack=1 Win=26196
4	0.004144	127.0.0.1	127.0.0.1	TCP	56	12345 → 61369 [PSH, ACK] Seq=1 Ack=1 Win=
5	0.004184	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=1 Ack=13 Win=2619
6	5.028182	127.0.0.1	127.0.0.1	TCP	52	61369 → 12345 [PSH, ACK] Seq=1 Ack=13 Win=
7	5.028219	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [ACK] Seq=13 Ack=9 Win=2619
8	5.028749	127.0.0.1	127.0.0.1	TCP	52	12345 → 61369 [PSH, ACK] Seq=13 Ack=9 Win=
9	5.028785	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=9 Ack=21 Win=2619
10	6.778563	127.0.0.1	127.0.0.1	TCP	49	61369 → 12345 [PSH, ACK] Seq=9 Ack=21 Win=
11	6.778600	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [ACK] Seq=21 Ack=14 Win=261
12	6.778877	127.0.0.1	127.0.0.1	TCP	50	12345 → 61369 [PSH, ACK] Seq=21 Ack=14 Wi=
13	6.778904	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=14 Ack=27 Win=261
14	6.779878	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [FIN, ACK] Seq=27 Ack=14 Wi=
15	6.779904	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=14 Ack=28 Win=261
16	6.780095	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [FIN, ACK] Seq=14 Ack=28 Wi=
17	6.780128	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [ACK] Seq=28 Ack=15 Win=261

a) Inicialización de la conexión. (1)

Segmentos: 1, 2

Métodos:

- Servidor:
 - ServerSocket server = new ServerSocket(puerto);
 - SocketClient client = server.accept();
- Cliente:
 - Socket serviceSocket = new Socket (serverName, serverPort);

b) Envío de datos. (2)

Segmentos: 3-13

Métodos:

- Cliente
 - Envío de datos:
 - `PrintWriter out = new PrintWriter(serviceSocket.getOutputStream());`
 - `out.println(userInput);`
 - `out.flush();`
 - Recepción de datos:
 - `BufferedReader in = new BufferedReader(new InputStreamReader(serviceSocket.getInputStream()));`
 - `String line = in.readLine();`
- Servidor
 - Envío de datos:
 - `PrintWriter out = new PrintWriter(client.getOutputStream(), true);`
 - `out.println(line)`
 - Recepción de datos:
 - `BufferedReader in = new BufferedReader(new InputStreamReader(client.getInputStream()));`
 - `String line = in.readLine();`

c) Finalización de la conexión. (3)

Segmentos: 14, 15, 16, 17

Métodos:

- Cliente
 - `ServiceSocket.close();` (cierre de socket)
 - `in.close();` (cierre de buffers)
 - `out.close();`
- Servidor
 - `in.close();` (cierre de buffers)
 - `out.close();`
 - `client.close();` (cierre de socket)

Ejercicio 9. ¿Cuántos números de secuencia se consumen en cada lado (cliente y servidor) durante el inicio y cierre de la conexión?

- Inicio:
 - Cliente: 0
 - Servidor: 0
- Cierre:
 - Cliente: 27
 - Servidor: 14

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	61369 → 12345 [SYN, Seq=0 Win=65535 Len=0...
2	0.000068	127.0.0.1	127.0.0.1	TCP	56	12345 → 61369 [SYN, ACK] Seq=0 Ack=1 Win=...
3	0.000105	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=1 Ack=1 Win=26196...
4	0.004144	127.0.0.1	127.0.0.1	TCP	56	12345 → 61369 [PSH, ACK] Seq=1 Ack=1 Win=...
5	0.004184	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=1 Ack=13 Win=2619...
6	5.028182	127.0.0.1	127.0.0.1	TCP	52	61369 → 12345 [PSH, ACK] Seq=1 Ack=13 Win=...
7	5.028219	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [ACK] Seq=13 Ack=9 Win=2619...
8	5.028749	127.0.0.1	127.0.0.1	TCP	52	12345 → 61369 [PSH, ACK] Seq=13 Ack=9 Win=...
9	5.028785	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=9 Ack=21 Win=2619...
10	6.778563	127.0.0.1	127.0.0.1	TCP	49	61369 → 12345 [PSH, ACK] Seq=9 Ack=21 Win=...
11	6.778600	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [ACK] Seq=21 Ack=14 Win=261...
12	6.778877	127.0.0.1	127.0.0.1	TCP	50	12345 → 61369 [PSH, ACK] Seq=21 Ack=14 Wi...
13	6.778904	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=14 Ack=27 Win=261...
14	6.779878	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [FIN, ACK] Seq=27 Ack=14 Wi...
15	6.779904	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [ACK] Seq=14 Ack=28 Win=261...
16	6.780095	127.0.0.1	127.0.0.1	TCP	44	61369 → 12345 [FIN, ACK] Seq=14 Ack=28 Wi...
17	6.780128	127.0.0.1	127.0.0.1	TCP	44	12345 → 61369 [ACK] Seq=28 Ack=15 Win=261...

Usando la traza TCP2 (b2e10.pcapng):**Ejercicio 10.**


- ¿Recibe algún tipo de respuesta el intento de conexión del cliente?

No, el servidor no está iniciado, por lo que se rechaza el intento de conexión del cliente.

No recibe respuesta de ningún servidor, pues no está iniciado. No puede recibir ninguna señal para conectarse con él.

- En caso afirmativo ¿tiene alguna característica especial?

Podemos ver que el cliente intenta conectarse en numerosas ocasiones con un posible servidor que espera encontrarse. En cuanto no consigue conectarse, vuelve a transmitir una señal para intentar conectarse con él.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	61370 → 12345 [SYN] Seq=0 Win=65535 Len=0...
2	0.000026	127.0.0.1	127.0.0.1	TCP	44	12345 → 61370 [RST, ACK] Seq=1 Ack=1 Win=...
3	0.502943	127.0.0.1	127.0.0.1	TCP	56	[TCP Retransmission] 61370 → 12345 [SYN] ...
4	0.502972	127.0.0.1	127.0.0.1	TCP	44	12345 → 61370 [RST, ACK] Seq=1 Ack=1 Win=...
5	1.005358	127.0.0.1	127.0.0.1	TCP	56	[TCP Retransmission] 61370 → 12345 [SYN] ...
6	1.005415	127.0.0.1	127.0.0.1	TCP	44	12345 → 61370 [RST, ACK] Seq=1 Ack=1 Win=...
7	1.517370	127.0.0.1	127.0.0.1	TCP	56	[TCP Retransmission] 61370 → 12345 [SYN] ...
8	1.517416	127.0.0.1	127.0.0.1	TCP	44	12345 → 61370 [RST, ACK] Seq=1 Ack=1 Win=...
9	2.029524	127.0.0.1	127.0.0.1	TCP	56	[TCP Retransmission] 61370 → 12345 [SYN] ...
10	2.029595	127.0.0.1	127.0.0.1	TCP	44	12345 → 61370 [RST, ACK] Seq=1 Ack=1 Win=...

Usando la traza TCP3 (b2e11-12.pcapng):**Ejercicio 11.**

- ¿Se logran conectar los 3 clientes?

El primer cliente sí se conecta sin problema, el segundo y tercer cliente se quedan en espera sin poder utilizar el servicio del Servidor.

- En caso de alguno no se haya podido conectar, ¿se le indica de alguna forma que la cola está llena?

El segundo y tercer cliente se quedan en lista de espera (en cola). No se excede el límite de usuarios en espera de usar el servicio del Servicio.

```
ESTADO: Esperando cliente
ESTADO: Cliente conectado desde: /127.0.0.1:61374
ESTADO: Recibido desde el cliente END
ESTADO: Cerrando conexión con el cliente
ESTADO: Esperando cliente
ESTADO: Cliente conectado desde: /127.0.0.1:61375
ESTADO: Recibido desde el cliente END
ESTADO: Cerrando conexión con el cliente
ESTADO: Esperando cliente
ESTADO: Cliente conectado desde: /127.0.0.1:61376
ESTADO: Recibido desde el cliente END
ESTADO: Cerrando conexión con el cliente
ESTADO: Esperando cliente
```


Ejercicio 12. ¿Los clientes en espera (es decir los que están en la cola) tiene inicializada la conexión o esa inicialización se hace cuando se sacan de la cola (con el método accept)?

El segundo y tercer cliente, que se encuentran en cola, no tienen inicializada la conexión con el servidor, pues el cliente que tiene el servidor en uso es el primer cliente. Una vez el primer cliente cierra conexión, el servidor acepta la conexión con el segundo cliente. El tercer cliente sigue en la cola de espera. Cuando el segundo cliente cierra la conexión con el servidor, el servidor acepta la conexión con el tercer cliente. Cuando el tercer cliente cierra la conexión con el servidor, el servidor se queda a la espera de conectarse con otro cliente que solicite conexión.

1	0.000000	127.0.0.1	127.0.0.1	TCP	56	61374 → 12345	[SYN] Seq=0 Win=6553...
2	0.000085	127.0.0.1	127.0.0.1	TCP	56	12345 → 61374	[SYN, ACK] Seq=0 Ack...
3	0.000129	127.0.0.1	127.0.0.1	TCP	44	61374 → 12345	[ACK] Seq=1 Ack=1 Wi...
4	0.006945	127.0.0.1	127.0.0.1	TCP	56	12345 → 61374	[PSH, ACK] Seq=1 Ack...
5	0.006979	127.0.0.1	127.0.0.1	TCP	44	61374 → 12345	[ACK] Seq=1 Ack=13 W...
6	6.553402	127.0.0.1	127.0.0.1	TCP	56	61375 → 12345	[SYN] Seq=0 Win=6553...
7	6.553489	127.0.0.1	127.0.0.1	TCP	56	12345 → 61375	[SYN, ACK] Seq=0 Ack...
8	6.553531	127.0.0.1	127.0.0.1	TCP	44	61375 → 12345	[ACK] Seq=1 Ack=1 Wi...
9	10.792203	127.0.0.1	127.0.0.1	TCP	56	61376 → 12345	[SYN] Seq=0 Win=6553...
10	10.792312	127.0.0.1	127.0.0.1	TCP	56	12345 → 61376	[SYN, ACK] Seq=0 Ack...
11	10.792371	127.0.0.1	127.0.0.1	TCP	44	61376 → 12345	[ACK] Seq=1 Ack=1 Wi...
12	24.386750	127.0.0.1	127.0.0.1	TCP	49	61374 → 12345	[PSH, ACK] Seq=1 Ack...
13	24.386788	127.0.0.1	127.0.0.1	TCP	44	12345 → 61374	[ACK] Seq=13 Ack=6 W...
14	24.387099	127.0.0.1	127.0.0.1	TCP	50	12345 → 61374	[PSH, ACK] Seq=13 Ac...
15	24.387124	127.0.0.1	127.0.0.1	TCP	44	61374 → 12345	[ACK] Seq=6 Ack=19 W...
16	24.388024	127.0.0.1	127.0.0.1	TCP	44	12345 → 61374	[FIN, ACK] Seq=19 Ac...
17	24.388047	127.0.0.1	127.0.0.1	TCP	44	61374 → 12345	[ACK] Seq=6 Ack=20 W...
18	24.388332	127.0.0.1	127.0.0.1	TCP	44	61374 → 12345	[FIN, ACK] Seq=6 Ack...
19	24.388368	127.0.0.1	127.0.0.1	TCP	44	12345 → 61374	[ACK] Seq=20 Ack=7 W...