Práctica 5

Redes y Sistemas Distribuidos
Grado de Ingeniería del Software (Grupo A)

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Funcionamiento de C/S TCP

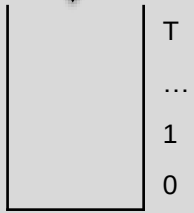
Socket Pasivo

Servidor

Puerto recepción de peticiones

```
s = new ServerSocket(P, T);
```

Cola de espera clientes



Funcionamiento de C/S TCP

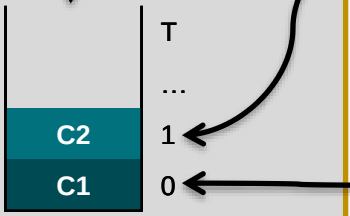
Servidor

Socket Pasivo

Puerto recepción de peticiones

```
s = new ServerSocket(P, T);
```

Cola de espera clientes



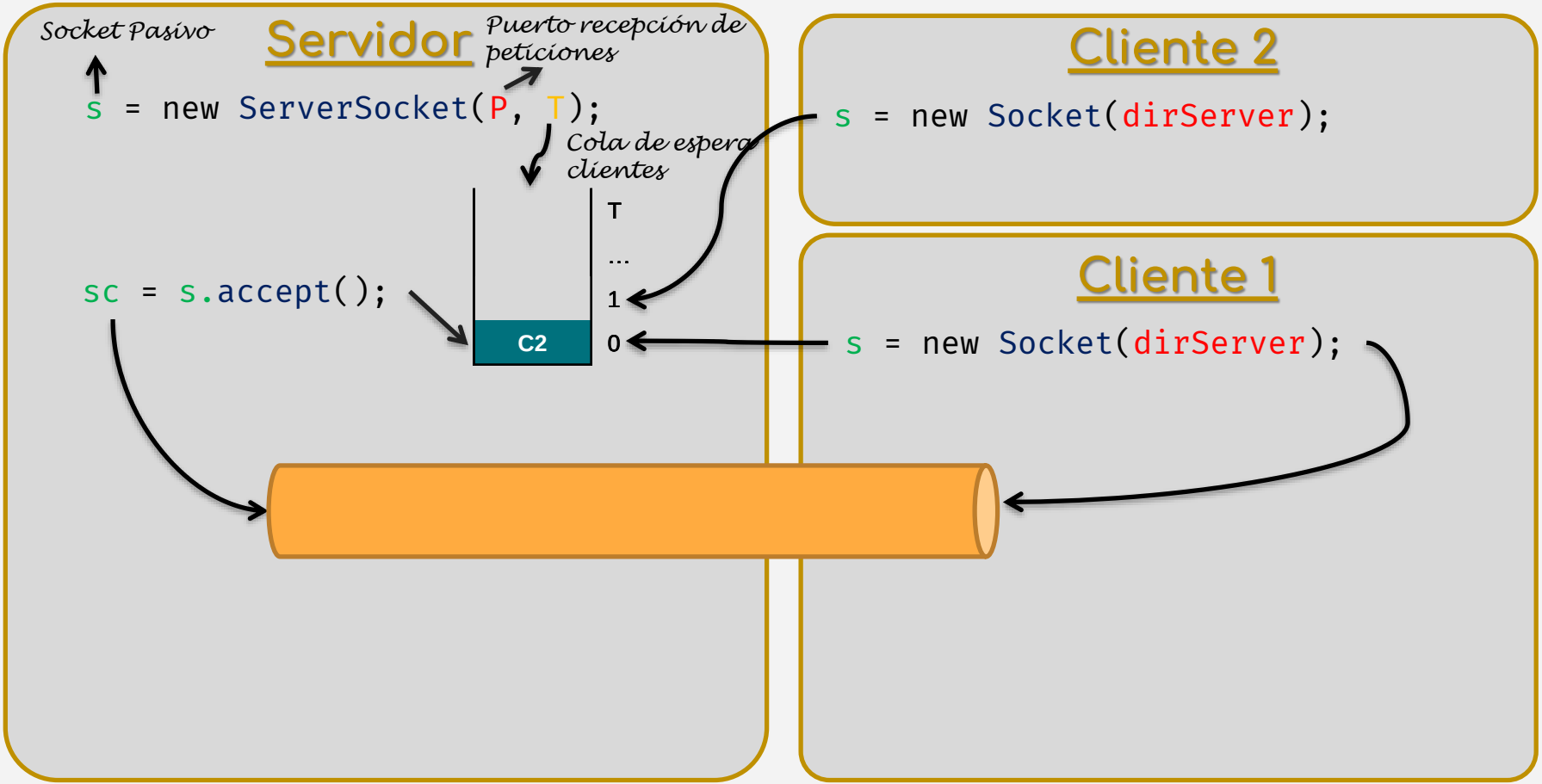
Cliente 2

```
s = new Socket(dirServer);
```

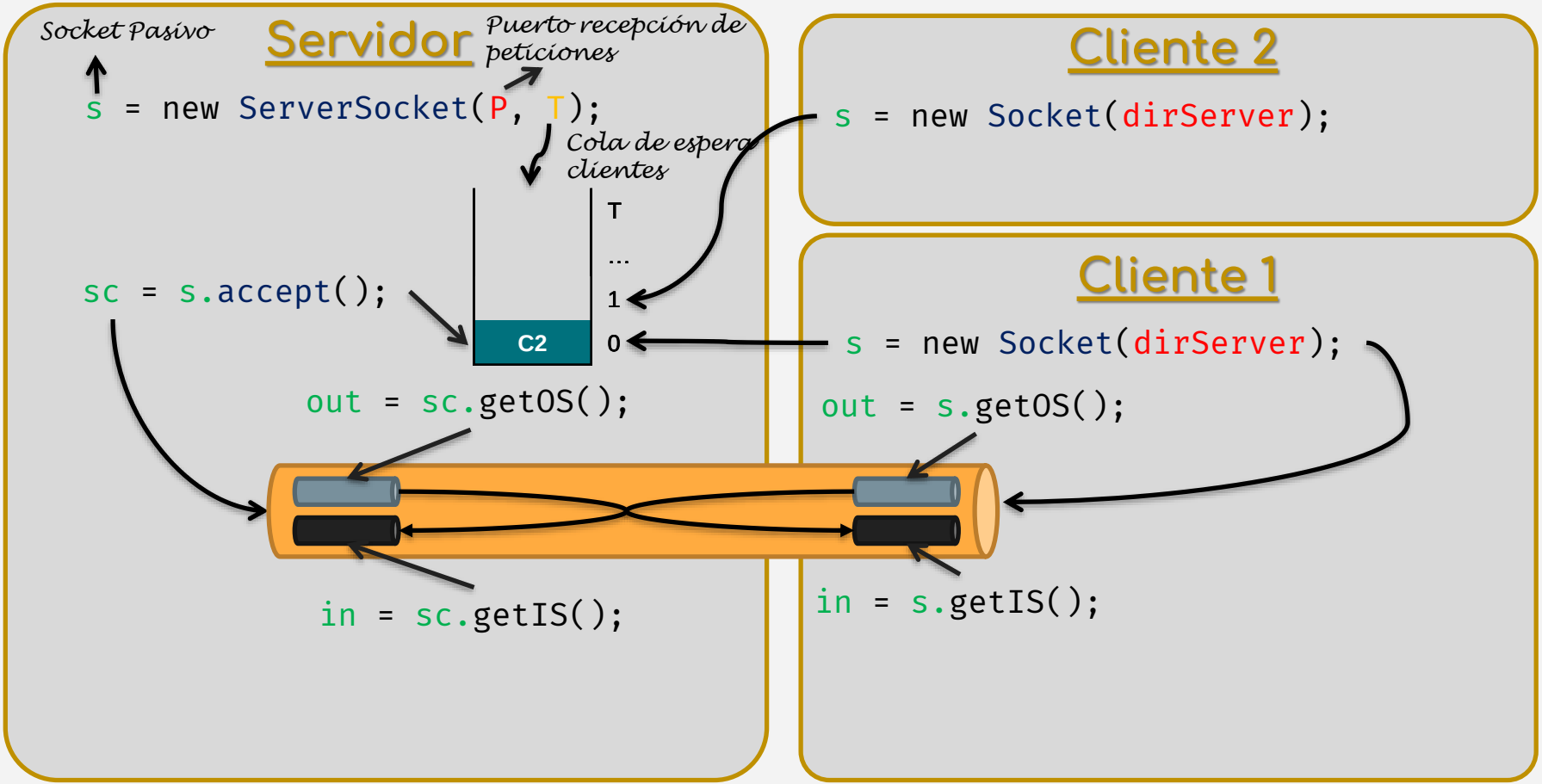
Cliente 1

```
s = new Socket(dirServer);
```

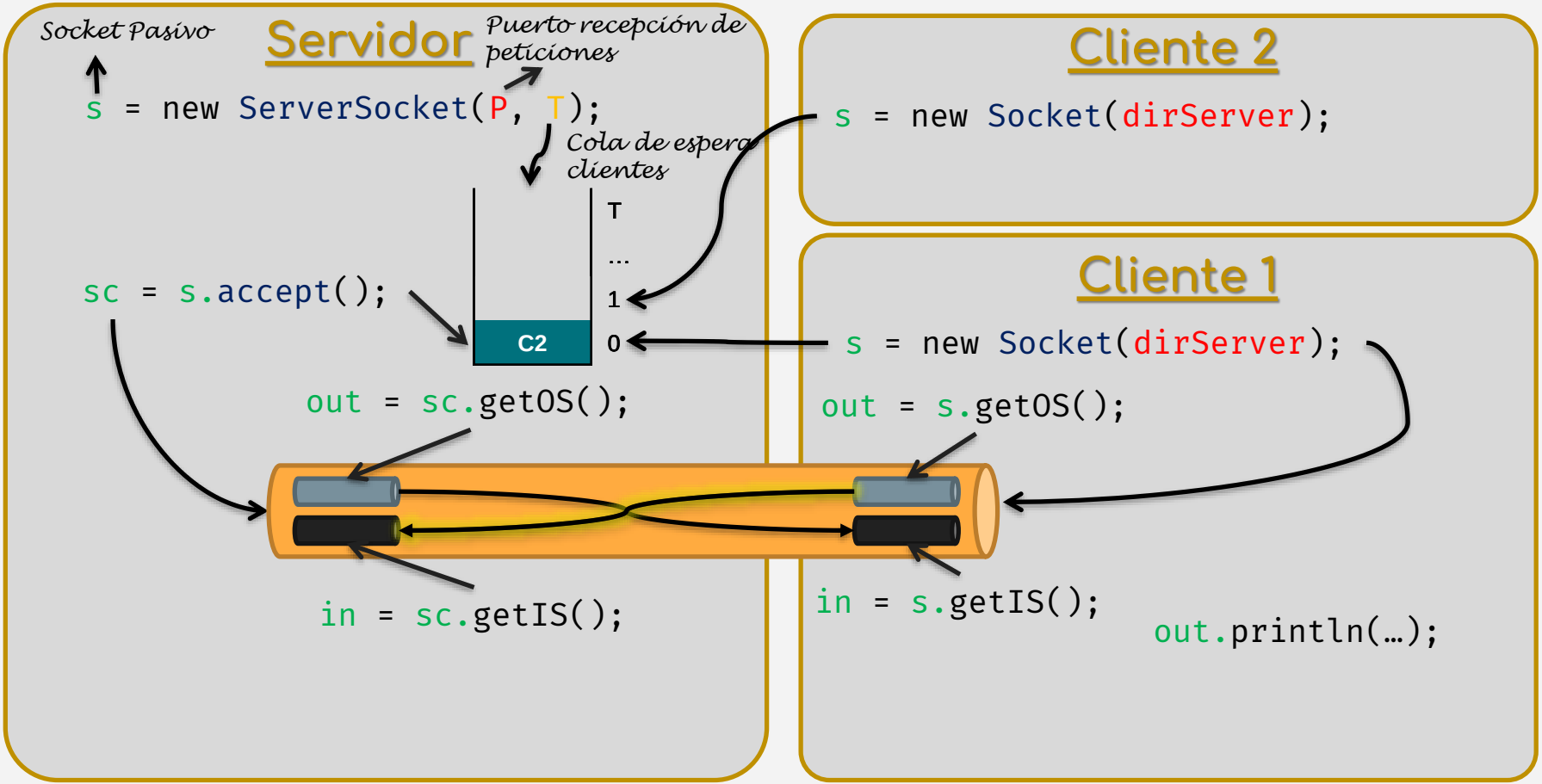
Funcionamiento de C/S TCP



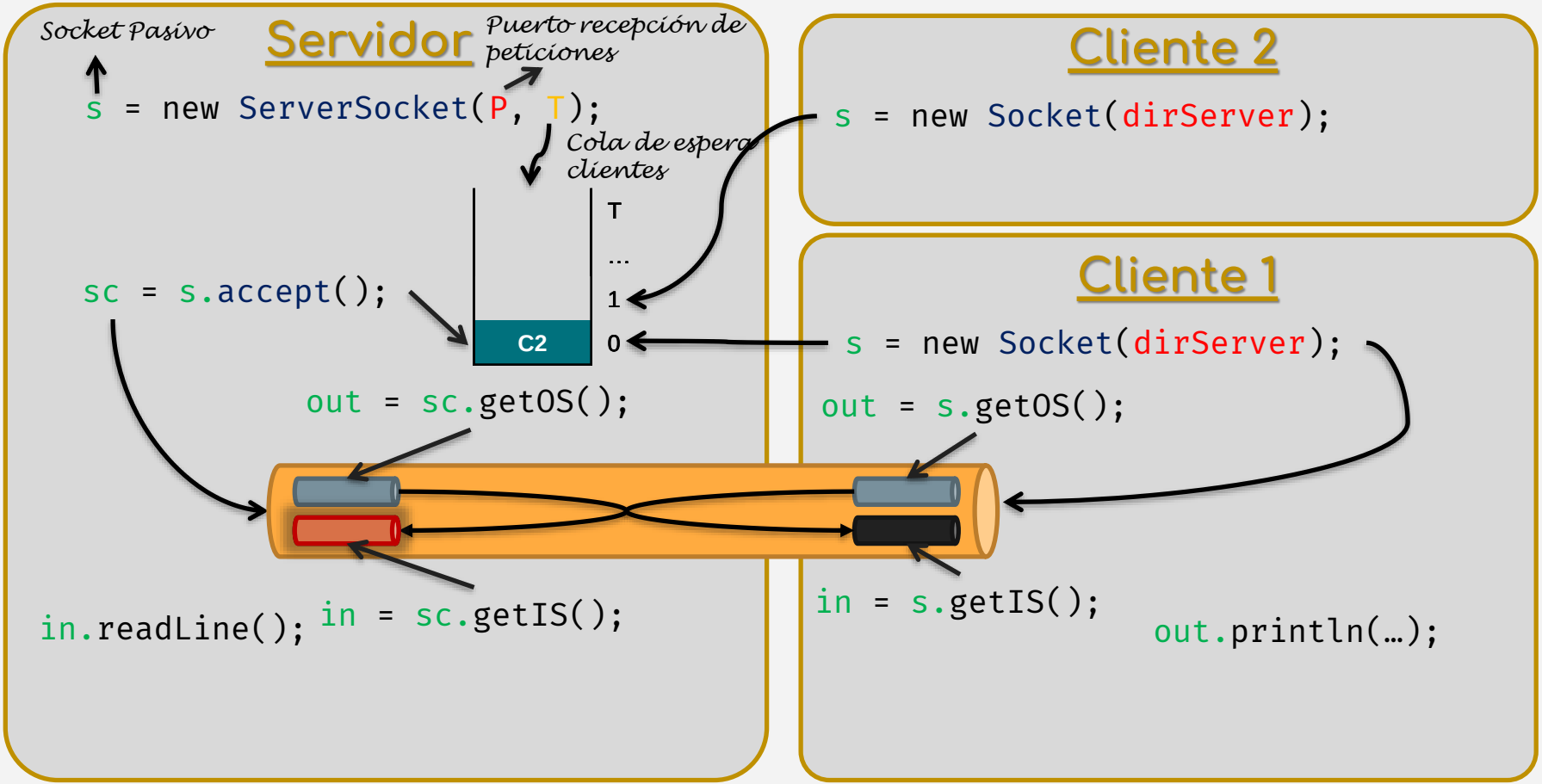
Funcionamiento de C/S TCP



Funcionamiento de C/S TCP



Funcionamiento de C/S TCP



Funcionamiento de C/S TCP

Servidor

Socket Pasivo

Puerto recepción de peticiones

```
s = new ServerSocket(P, T);
```

Cola de espera clientes

T
...
1
0

C2

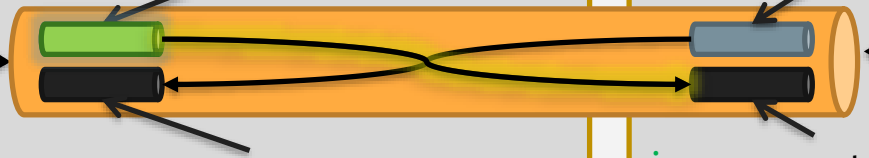
```
sc = s.accept();
out = sc.getOS();
in = sc.getIS();
in.readLine();
out.println(…);
```

Cliente 2

```
s = new Socket(dirServer);
```

Cliente 1

```
s = new Socket(dirServer);
out = s.getOS();
in = s.getIS();
out.println(…);
```



Funcionamiento de C/S TCP

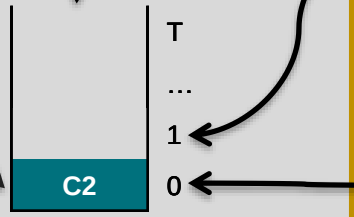
Servidor

Socket Pasivo

Puerto recepción de peticiones

```
s = new ServerSocket(P, T);
```

Cola de espera clientes



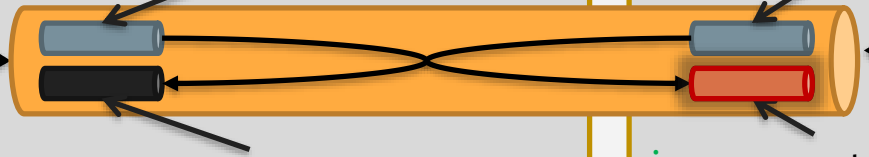

```
sc = s.accept();
out = sc.getOS();
in = sc.getIS();
in.readLine();
out.println(…);
```

Cliente 2

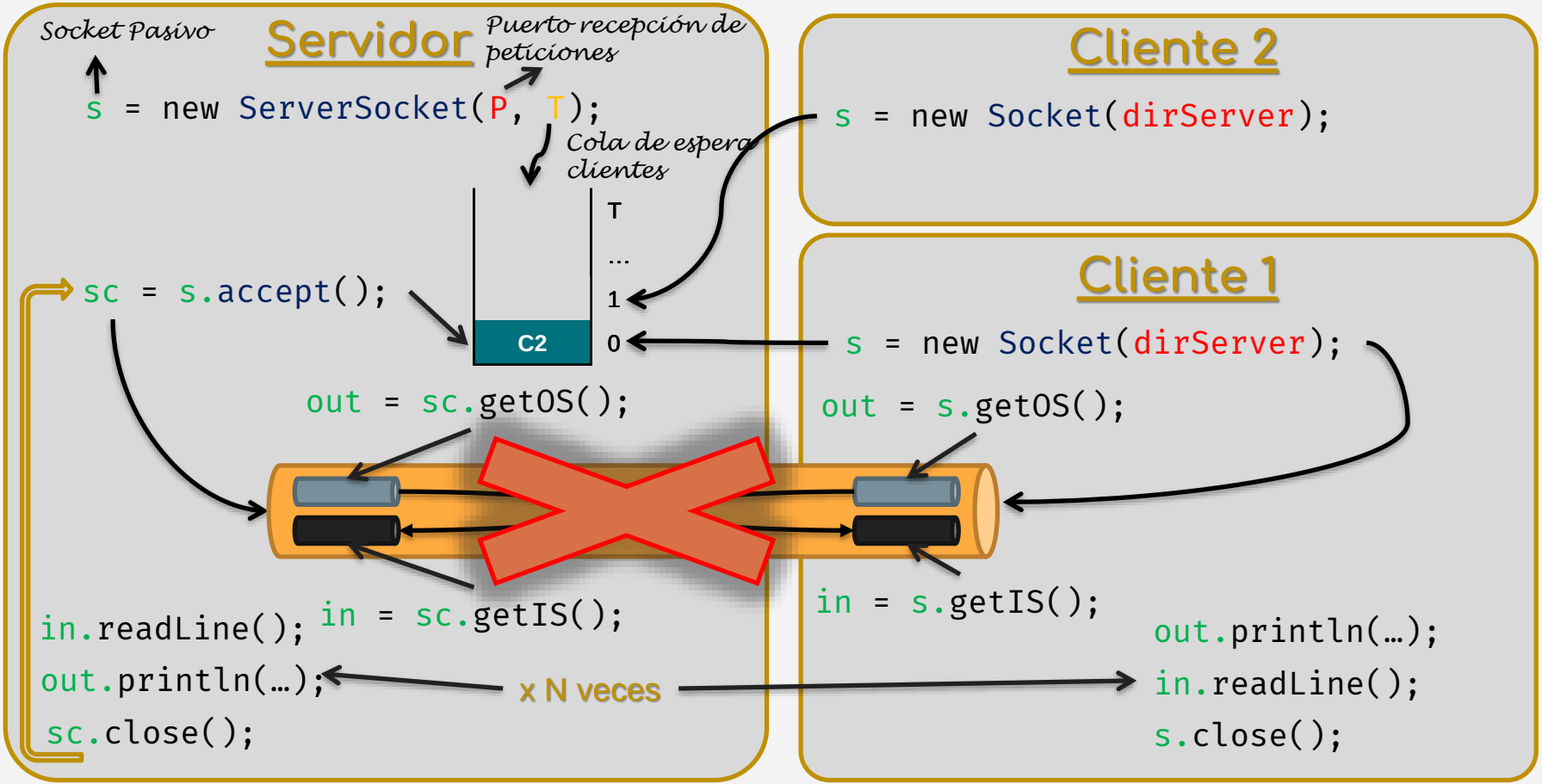
```
s = new Socket(dirServer);
```

Cliente 1

```
s = new Socket(dirServer);
out = s.getOS();
in = s.getIS();
out.println(…);
in.readLine();
```



Funcionamiento de C/S TCP



ServerSocket

Socket para un servidor orientado a la conexión

Constructores:

- `ServerSocket(int puerto)`: crea un socket pque recibe peticiones por el puerto indicado (0 = cualquier disponible). Cola de 50.
- `ServerSocket(int puerto, int count)`: igual que el previo pero con el tamaño de la cola (máximo número de peticiones en espera)
- `ServerSocket()`: crea un socket no vinculado (se debe usar `bind` con un puerto antes de usarse)

Métodos:

- `Socket accept()`: saca una petición de la cola de peticiones (si no hay se bloquea) y crea un socket conectado al cliente
- `bind(SocketAddress sa)`: vincula un socket a un puerto
- `close()`: libera el puerto y los recursos

Socket

Socket (conectado) para una comunicación TCP

Constructores:

- `Socket(String ip, int puerto)`: crea un socket TCP y lo conecta a la dirección IP y puertos indicados
- `Socket(InetAddress ip, int puerto)`: igual que el previo

Métodos:

- `InputStream getInputStream()`: devuelve el stream para la recepción de datos
- `InputStream getOutputStream()`: devuelve el stream para el envío de datos
- `close()`: libera el puerto y los recursos

Envío y recepción a través de Socket

`getInputStream` y `getOutputStream` obtienen los flujos asociados

Recepción:

```
BufferedReader r = new BufferedReader(
                       new InputStreamReader(s.getInputStream())
                   );
```

- `BufferedReader` dispone de los métodos `read` (recibe uno o varios bytes) y `readLine` (recibe líneas completas y devuelve un `String`)

Envío:

```
PrintWriter s = new PrintWriter(socket.getOutputStream(),true);
```

- `PrintWrite` dispone de los métodos `write` (envía un conjunto de líneas) y `println / print` (envía `String` completas con / sin '`\n`')

Protocolo de aplicación

Mismo servicio que con el protocolo UDP (capitalización de textos)

Cliente:

- Al introducir el usuario TERMINAR, enviará un END y esperará el OK del servidor

Servidor:

- Cuando el cliente acaba enviará un END y el servidor responderá OK
- Cola de espera de clientes tamaño 1