No Classroom lab: Sockets programming

We wish to have a registry of the students attending the lab. To do so, we want a server to which students can communicate and Works has follows:

- The server receives a UDP datagram in port 8080, sent from a client X (the Student). In this datagram the client send a string representing an integer (e.g. 9090)
- Using a new socket, the server establish a TCP connection with client host X with the port sent in the previous UDP message.
- Using the TCP connection, the client sends the student's name to the server. When the server receives the student's name, it is shown in the screen.

You are request to write the code using C of a client able to interact with the server described previously. The client is called using the following parameters in the command line:

$ client < SERVER_HOST_NAME>

The server's code is provided as a jar file, and can be executed executing:

$ java –jar servidor.jar

Capture the messages exchanged between the client and the server with tcpdump (or wireshark). If both, client and server, are run in the same host, you must capture the packets using the loopback interface ("lo" in linux):

tcpdump -vs 1500 -ni lo

Execute the following command immediately after communicating the client and server. Identify the TCP socket that was created due to the connection, check that it is in state TIME_WAIT.

netstat –nat

Evaluation

Nothing must be delivered. The session will be evaluated the same day of the final lab exam, by means of a quiz. In the following you have some examples of the type of questions that you will find in the quiz.

Example of the type of questions that might be used in the evaluation of the sockets lab:

We have a system with an only networking interface with IP address 150.12.27.42. We execute the code::

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MIDABUFFER 256

1 int main(int argc, char *argv[])
2 {
3   char     buf[MIDABUFFER];
4   int      sdfS, sfdR, addrlen;
5   struct   sockaddr_in sinS, sin2;
6   int      llegit;

8   memset(&sinS,0,sizeof(sinS)); memset(&sin2,0,sizeof(sin2));
9   if((sdfS = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
10    perror("socket");
11    exit(1);
12  }
13  sinS.sin_family = AF_INET;
14  sinS.sin_addr.s_addr = inet_addr("127.0.0.1");
15  sinS.sin_port = htons(8080);
16  if(bind(sdfS, (struct sockaddr *)&sinS, sizeof(sinS)) == -1) {
17    perror("bind"); exit(1);
18  }
19  if(listen(sdfS, 10) == -1) {
20    perror("listen"); exit(1);
21  }
22  addrlen = sizeof(sin2);
23  if((sfdR = accept(sdfS, (struct sockaddr *)&sin2, &addrlen)) == -1) {
24    perror("accept"); exit(1);
25  }
26  while(1) {
27    if((llegit = read(sfdR, buf, MIDABUFFER)) == -1) {
28      perror("send"); exit(1);
29    }
30    if(llegit > 0) {
31      printf("Rx (%d): %s", llegit, buf);
32    } else {
33      break;
34    }
35  }
36  close(sfdR) ; close(sdfS) ;
37 }
```

Say which of the following statements are true regarding the previous code of the server:

1) It uses TCP sockets (true)
2) When the code is executed it will be blocked in line 23 until a client is connected (true)
3) Upon executing the code, it is possible to get the following dump (true)

```
> netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN
```

4) Upon a client is connected, it is possible to get the following dump (true)

```
> netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:8080          127.0.0.1:55848         ESTABLISHED
```

5) The client will have to execute the system call “connect” to establish the connection with the server. (true)
6) Upon executing the server code twice, the following dump will be obtained: (false)

```
> netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN
```