Systems programming

Laboratory 3 A - ZeroMQ and Internet sockets

Learning objectives

Client server using ZeroMQ

Internet domain sockets

Data abstraction pattern

Internet domain sockets allow the communication of programs running in multiple computers using the Internet.

The way to establish communication using Internet domain sockets is similar with the use of unix domain sockets. The main different is the way the socket addresses (identification of the recipient socket) are defined.

For two processes to communicate with Internet domain sockets it is necessary to create one socket on each program: each communication participant should create a socket.

To receive data from another remote socket it is necessary to assign/bind an address. This can be done explicitly with the bind function or implicitly by the **sendto**. In internet domain sockets, after sending data with the **sendto** function, such socket is ready to receive messages. In this case the recipient of the message will get an address that was implicitly assigned/bind during the **sendto**.

1 Internet domain sockets addresses

ZeroMQ Unix domain socket use addresses composed of a string (file path). This is enough to allow processes in the same computer to use (bind and connect) those sockets, but for Internet communication a more complex address is needed.

In Internet domain sockets addresses are composed of two distinct parts:

- the IP address (for instance 146.193.41.15)
- the port (for instance 80)

by knowing the address of the server and the port of the program any other client on the internet can connect to it.

1.1 Internet domain TCP addresses

The generic representation of a ZermoMQ TCP Internet domain socket is **tcp://<addr>:<port> .** Depending whether this address is being used in the server (in the bind) or in the clients (doing connect), the **addr** part can change, while the port should be the same (for client and server).

The port should be unique on the computer running the server application and, for regular applications, usually ranges between 1024 and 49151. In our example this value is defined as 5555.

When performing the connect, the client should use one of the numeric addresses (using the **xxx.xxx.xxx.xxx** notation) assigned to the server computer on the **addr** part.

NI the bind, the server can use a wildcard (*****) in the **addr** part, meaning that if the computer running the server has several network cards, with multiple addresses, the server will be accessible in any of those addresses. If the server application is supposed to be accesses in a specific address , the **addr** part should contain it.

If the client uses the same address the server used in the bind (or the server used the wildcard) a connection will be established unless there is a network outrage. If the addresses do not match an error will be returned.

1.2 Available addresses in computers

Computers connected to the Internet can have multiple addresses.

One of such addresses is the **127.0.0.1**. All computers have this address and, if used in a client, the program will try to communicate with a server running on the same computer.

To accept remote connections it is necessary to tell the clients what are the available addresses of the computer that runs the server. To know such addresses there are a some commands to be executed on the server computer:

- in Linux – **ip** a or **ifconfig**
- MAC OS X – **ifconfig**
- Windows – **ipconfig**

These command should be executed in a terminal or command prompt window and will produce the following outputs.

ip a

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 . . .  qlen 1000
. . . .
```

```
    inet 127.0.0.1/8 scope host lo
. . . .
2: enp4s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> . . . qlen 1000
. . . .
    inet 146.193.41.15/24 brd 146.193.41.255 . . .  enp4s0f0
. . . .
```

ip a (Linux / WSL)

```
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
. . . .
   inet 127.0.0.1 netmask 0xff000000
. . . .
en0: flags=8863<UP,. . . . > mtu 1500
. . . .
   inet 146.193.41.40 netmask 0xffffff00 broadcast 146.193.41.255
. . . .
   status: active
```

ipconfig

```
Windows IP Configuration
Ethernet adapter vEthernet (WSL):
IPv4 Address. . . . . . . . . . . : 172.26.0.1
Ethernet adapter Ethernet 4:
IPv4 Address. . . . . . . . . . . : 192.168.56.1
Wireless LAN adapter Wi-Fi:
IPv4 Address. . . . . . . . . . . : 194.210.228.171
```

For different computers, the number of available addresses may change but the relevant ones are identified as **inet xxx.xxx.xxx.xxx** (in Linux/MAC OS X /WSL) or **Ipv4 Adress xxx.xxx.xxx.xxx** in windows.

Not all addresses are accessible remotely because of network or firewall configurations.

2 Exercise 3 - Remote control on the Internet

Modify the provided **Client-server-system** so that the server and clients run on different computers using **ZeroMQ** TCP sockets.

To test the system using different computers create a WIFI Hot Spot with a smartphone and connect both computers to that network.

To compile the provided code you need to install the **ncurses** library:

```
sudo apt-get install libncurses5-dev libncursesw5-dev
```

or

```
sudo apt-get install libncurses-dev
```

or

```
brew install ncurses
```

3 Exercise 4 - Response messages

Modify the system so that if a client tries to use a letter already assigned the server send as response to the CONNECT message and ERROR message., and the clinet terminates.

Modify further the system so that when a letter touches the border of the window, the client terminates and the letter is removed.

4 Exercise 4 - Data abstraction

Observe how all the functions used to implement communication between the client and the server hide completely the type of channel and the structure of the message: in the client and server main no reference is made to thos data structures.

Guarantee that the previous exercises maintain this data abstraction and try to apply the same principle to the code that manipulates the screen. TO understand the used ncurses functions, look at the following documentation:

https://invisible-island.net/ncurses/ncurses.html

https://invisible-island.net/ncurses/ncurses-intro.html

https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/