

# Memòria pràctica 7:

## Disseny web del control remot d'un led

Sergi Carol i Enric Lenard

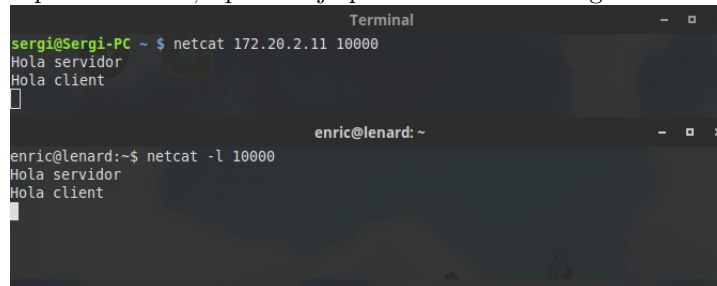
May 13, 2015

### 1 Netcat

#### 1.1 Tasca 2

Per tal de realitzar aquesta tasca primer em mirat la documentació de la eina **netcat**. Hem vist tot el seguit de opcions i hem observat que el que ens interessa fer és, en la màquina servidora, executar l'ordre netcat amb la flag **-l** i a continuació el port. Per tan la comanda seria la següent: **netcat -l 10000**. Mentre que a la màquina client voldrem especificar a quin servidor ens voldrem connectar, en el nostre cas el servidor es troba a la IP 172.20.2.11, per tan tindrem: **netcat 172.20.2.11 10000**.

Un cop executades aquestes dues comandes ja tenim les sockets enllaçades mitjançant el protocol TCP, i per tan ja podem enviar missatges.



```
Terminal
sergi@Sergi-PC ~ $ netcat 172.20.2.11 10000
Hola servidor
Hola client
[ ]

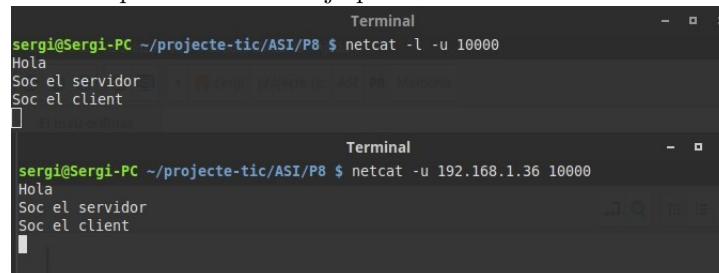
enric@lenard: ~
enric@lenard:~$ netcat -l 10000
Hola servidor
Hola client
```

Seria difícil executar comandes a un servidor, però no impossible, ja que existeixen tècniques de Remote shell o Backdoor. La primera és necessari que el servidor permeti la execució de comandes afegint més opcions a la comanda netcat, com per exemple la opció **-e** que permetria fer **netcat -l 10000 -e /bin/bash** que permetria que les comandes escrites pel client fossin executades a la màquina del servidor. Amb la nostra versió de netcat això no és possible, però el manual suggereix utilitzar la següent comanda: **rm -f /tmp/f; mkfifo**

`/tmp/f` i a continuació `cat /tmp/f — /bin/sh -i 2> & 1 — nc -l 127.0.0.1 1234 > /tmp/f`. Tot i així no es recomana fer servir aquestes opcions ja que deixen el sistema molt desprotegit.

## 1.2 Tasca 3

Ordre pel servidor `nc -l -u 10000` pel client `nc -u 172.20.2.11 10000`. Un cop executades aquestes comandes ja podem connectar-nos via UDP.



The image shows two terminal windows. The top window shows a netcat listener on port 10000 receiving a connection from 172.20.2.11. The bottom window shows a netcat client connecting to 192.168.1.36 on port 10000. Both show the exchange of 'Hola' and 'Soc el servidor'/'Soc el client' messages.

```
Terminal
sergi@Sergi-PC ~/projecte-tic/ASI/P8 $ netcat -l -u 10000
Hola
Soc el servidor
Soc el client
[ ]

Terminal
sergi@Sergi-PC ~/projecte-tic/ASI/P8 $ netcat -u 192.168.1.36 10000
Hola
Soc el servidor
Soc el client
[ ]
```

## 1.3 Tasca 4

En el cas de que el servidor sigui **TCP** el servidor només permetrà una connexió, ja que només existeix un process per atendre la connexió. En canvi, en el cas de **UDP** el servidor pot rebre els dos clients, ja que no és orientat a connexió i per tan allibera la connexió un cop el missatge s'ha entregat.

## 1.4 Tasca 5

En el cas de **netcat** existeix un bug a l'aplicació que no permet que es connectin dos clients a la vegada, encara que el servei sigui UDP, ja que en el cas de TCP no ho permet per lo que hem mencionat anteriorment, però amb UDP que si que tindria no deixa.

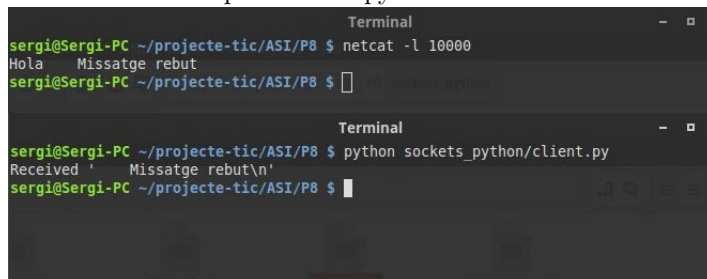
# 2 Sockets amb Python

## 2.1 Tasca 6

Hem comprovat els exemples proposats i el seu funcionament. Podem veure com tan el client com el servidor , un cop enviat el missatge tenquen les seves connexions.

## 2.2 Tasca 7

A continuació farem la comprovació de python amb netcat.



The image contains two terminal window screenshots. The top terminal shows a netcat listener on port 10000. It receives a connection and prints 'Hola Missatge rebut'. The bottom terminal shows a python client script 'sockets\_python/client.py' being executed. It prints 'Received ' Missatge rebut\n' and then returns to the shell prompt.

```
Terminal
sergi@Sergi-PC ~/projecte-tic/ASI/P8 $ netcat -l 10000
Hola Missatge rebut
sergi@Sergi-PC ~/projecte-tic/ASI/P8 $

Terminal
sergi@Sergi-PC ~/projecte-tic/ASI/P8 $ python sockets_python/client.py
Received ' Missatge rebut\n'
sergi@Sergi-PC ~/projecte-tic/ASI/P8 $
```

Veiem com el client envia un missatge al servidor netcat, el client python es queda obert fins que rep un missatge de confirmació del servidor, que en aquest cas tenim que escriure manualment.

## 2.3 Tasca 8

Per tal de passar de **TCP** a **UDP** tenim que canviar un parell de coses. Primer de tot tenim que canviar el tipus de socket, que passara de **SOCK\_STREAM** a **SOCK\_DGRAM**. Per tan tindrem la següent comanda:

```
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

A més a més ja que hem passat a un servei no orientat a connexió tenim que treure les següents línies al servidor:

```
s.listen(1)
conn= s.accept()
```

Ja que amb udp no tenim que estar esperant connexions, ni tampoc les tenim que acceptar ja que no esta orientat a connexió. Finalment una forma amigable de gestionar les rebudes de missatges es utilitzant la funció **recvfrom** que ens retorna el missatge rebut i a més a més la IP del emissor.

En el cas del client la única cosa que tenim que modificar és la socket de la forma que hem mencionat anteriorment.

## 2.4 Tasca 9

Cal tenir en compte que en aquest apartat el sistema es queda bloquejat mentre existeix un raw\_input, per tan els missatges es reben tard. Aixó queda millorat en tasques posteriors. L'arxiu utilitzat es *client\_udp.py* i *servidor\_udp.py*

```

sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_python $ python client_servidor.py 192
.168.1.36 10000
Escriu el que vols enviar: Hey
Escriu el que vols enviar:

import sys
HOST = sys.argv[1] # the remote host
PORT = int(sys.argv[2])
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect((HOST, PORT))
s.settimeout(1)
message = raw_input('Escriu el que vols enviar: ')
s.send(message)
s.close()

sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_python $ python client_servidor.py 100
00
soc el servidor
10000
Connected by ('192.168.1.36', 43583) Hola
Que tal
Connected by ('192.168.1.36', 47704) Hey
[]

Benvingut a la terminal

The following files are in the working directory:
client.py          client_tcp.py      servidor_tcp.py
client_servidor_multiple.py  client_udp.py      servidor_udp.py
client_servidor_multiple.py~ client_udp.py~     sockets.py~
client_servidor.py servidor.py

sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_python $ python client_servidor.py 192.168.1.36 10000
Escriu el que vols enviar: Hola
Escriu el que vols enviar: H
Received 'Que tal' From ('192.168.1.36', 10000)
Escriu el que vols enviar:

```

Podem veure com els dos clients poden enviar missatges al servidor, i el servidor pot responde missatges al últim client que ha enviat un missatge.

## 2.5 Tasca 12

En aquesta última tasca hem ajuntat les tasques 10 i 11, ja que les dues acaben ajuntades amb el mateix codi. Tot i així l'arxiu **client\_servidor\_udp.py** conte el codi realitzat en la tasca 10 i 11.

En aquest cas hem creat u codi amb protocol UDP, per tan, que permet multiples connexions, a més a més gràcies a la eina **select** hem pogut solucionar el problema mencionat anteriorment que produia que el sistema es quedés encallat a la funció `ra_input`.

La funcio `select` ens permet estar pendent de varis *events*, com per exemple rebre un missatge a través de una socket o que s'escrigui per el `stdin`. La forma de utilitzar aquesta funció en el nostre cas és al següent:

```

socket_list = [sys.stdin, s]
read_sockets, write_sockets, error_sockets = select.select(socket_list, [], [])
for sock in read_sockets:
    #incoming message from remote server
    if sock == s:
        data, addr = sock.recvfrom(1024)
        if not data:
            print '\nDisconnected from chat server'
            sys.exit()
        else:
            prompt_him(data)

```

```

#user entered a message
else :
    msg = sys.stdin.readline()
    s.send(msg)

```

En el primera linia creem una llista de dos elements, el primer es tracta del **stdin**, osigui, el *standar input*. El segon element es la socket en si.

La funcio select permet retorna tres coses. Si hi ha hagut una lectura, una escriptura o un missatge de error. A continuacio del select iterem cada element que retorna la funcio select de lectura, en el cas de que l'element retornat sigui el socket voldra dir que hem rebut un missatge a través del socket i per tan tenim que tractar-lo. En el cas de que no sigui el socket voldra dir que ha sigut un stdin, i per tan que tenim que enviar un missatge.

D'aquesta forma hem solucionat el problema del raw\_input. Com que el nostre servidor només accepta connexions UDP no es necessari que creem processos per cada missatge rebut.

The image displays three terminal windows illustrating the operation of a Python socket server and its interaction with multiple clients.

- Top Left Terminal:** Shows the execution of `python client_servidor_multiple.py 192.168.1.36 10000`. The server prompts for a client name, receives "Client 1", and then enters a loop where it repeatedly sends "Hola client 1 soc el servidor" to the client.
- Top Right Terminal:** Shows the execution of `python client_servidor_multiple.py 10000`. The client repeatedly sends messages to the server, which responds with "Soc el servidor". The messages sent by the client are: "Him>: python sockets\_python/client\_udp.py 192.168.1.36 10000", "Him>:", "Him>: Hey", "Him>: Client 1", "Him>: Client 2", "Him>: Torno a ser el Client 1", and "Him>: Hola client 1 soc el servidor".
- Bottom Terminal:** Shows the execution of `python client_servidor_multiple.py 192.168.1.36 10000`. The client sends "Hey" and "Client 2", to which the server responds with "Him>: Soc el servidor".

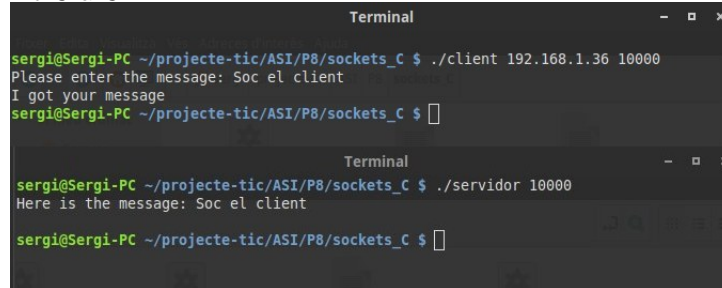
Un exemple del seu funcionament.

## 3 Sockets amb C

### 3.1 Tasca 13

Un cop obtinguts els exemples hem procedit a comprovar el seu funcionament, tenim que tenir en compte que aquest codis estan fets per TCP.

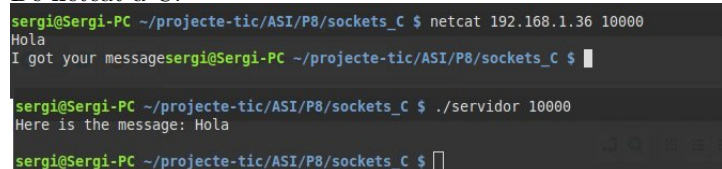
De C a C:



```
Terminal
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $ ./client 192.168.1.36 10000
Please enter the message: Soc el client
I got your message
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $

Terminal
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $ ./servidor 10000
Here is the message: Soc el client
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $
```

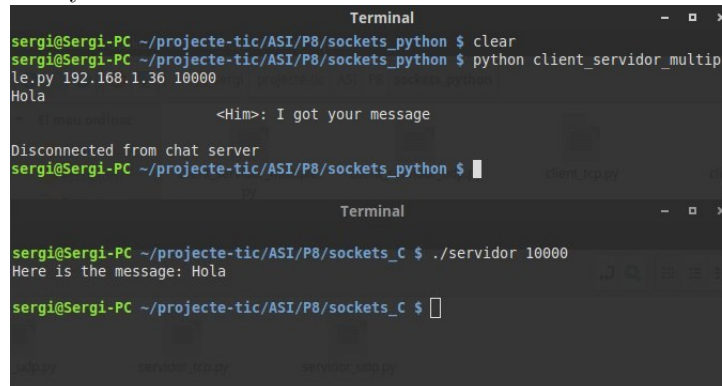
De netcat a C:



```
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $ netcat 192.168.1.36 10000
Hola
I got your messages
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $

sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $ ./servidor 10000
Here is the message: Hola
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $
```

De Python a C:



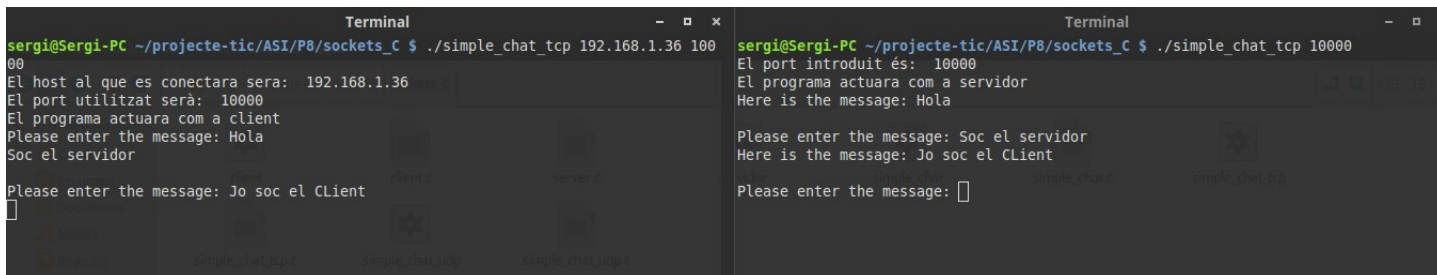
```
Terminal
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_python $ clear
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_python $ python client_servidor_multip
le.py 192.168.1.36 10000
Hola
<Him>: I got your message
Disconnected from chat server
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_python $

Terminal
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $ ./servidor 10000
Here is the message: Hola
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $
```

## 3.2 Tasca 14

En aquesta tasca hem ajuntat les dues aplicacions anteriors per tal de tenir una aplicació de xat en un sol fitxer, a més a més hem realitzat algunes modificacions, com per exemple el programa ja no s'acaba un cop s'ha rebut el missatge, sino que el tan el servidor com el client es queden oberts, a més a més el servidor ara pot escriure missatges cap el client també.

De client TCP C a servidor TCP C



```
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $ ./simple_chat_tcp 192.168.1.36 100
00
El host al que es connectara sera: 192.168.1.36
El port utilitzat sera: 10000
El programa actuara com a client
Please enter the message: Hola
Soc el servidor

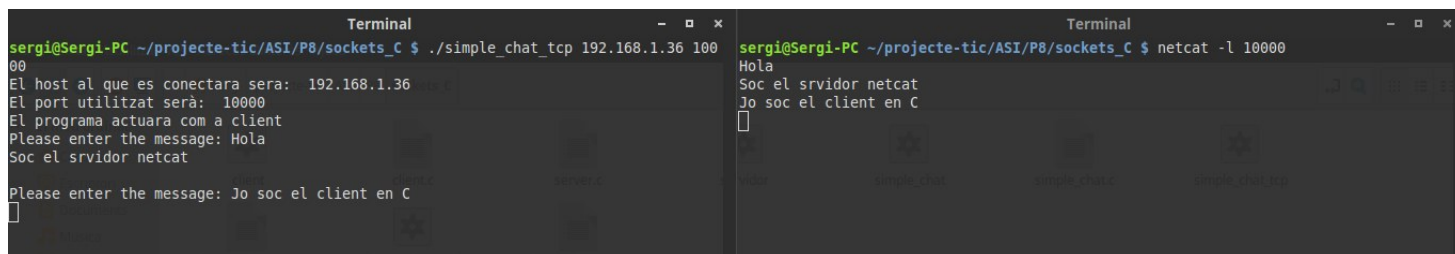
Please enter the message: Jo soc el Client
[]

sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $ ./simple_chat_tcp 10000
El port introduit és: 10000
El programa actuara com a servidor
Here is the message: Hola

Please enter the message: Soc el servidor
Here is the message: Jo soc el Client

Please enter the message: []
```

De client C a servidor netcat

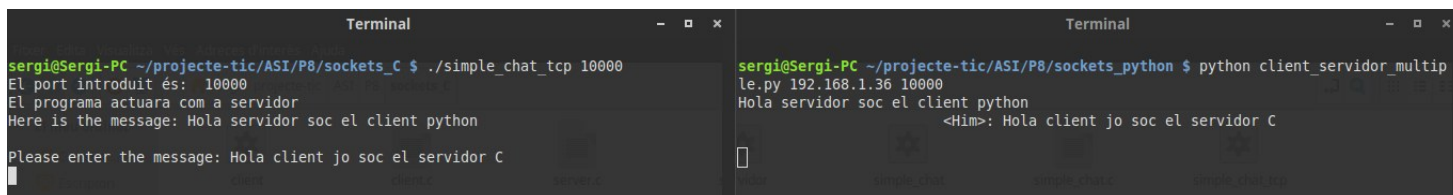


```
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $ ./simple_chat_tcp 192.168.1.36 100
00
El host al que es connectara sera: 192.168.1.36
El port utilitzat sera: 10000
El programa actuara com a client
Please enter the message: Hola
Soc el srvidor netcat

Please enter the message: Jo soc el client en C
[]

sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $ netcat -l 10000
Hola
Soc el srvidor netcat
Jo soc el client en C
[]
```

De servidor C a client python



```
sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C $ ./simple_chat_tcp 10000
El port introduit és: 10000
El programa actuara com a servidor
Here is the message: Hola servidor soc el client python

Please enter the message: Hola client jo soc el servidor C
[]

sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_python $ python client_servidor_multip
le.py 192.168.1.36 10000
Hola servidor soc el client python
<Him>: Hola client jo soc el servidor C

[]
```

### 3.3 Opcional: Servidor-Client en UDP

A més hem modificat el codi anterior de forma que es puguin realitzar connexions del tipus **UDP**. Per fer-ho hem hagut de modificar el servidor i el client de forma semblant a com ho hem fet amb el servidor python.

De servidor C a client C UDP

Terminal	Terminal
<pre>sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C \$ ./simple_chat_udp 192.168.1.36 10000 El host al que es connectara sera: 192.168.1.36 El port utilitzat sera: 10000 El programa actuara com a client Please enter the message: Soc el client Jo el server  Please enter the message: exit Tencent Conexio sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C \$</pre>	<pre>sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C \$ ./simple_chat_udp 10000 El port introduit és: 10000 El programa actuara com a servidor Data: Soc el client Please enter the message: Jo el server Data: exit Please enter the message: exit Tencent Conexio out sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C \$</pre>

De client C a python UDP

Terminal	Terminal
<pre>sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C \$ ./simple_chat_udp 192.168.1.36 10000 El host al que es connectara sera: 192.168.1.36 El port utilitzat sera: 10000 El programa actuara com a client Please enter the message: Hola servidor soc un client udp Hola client soc un servidor python  Please enter the message: </pre>	<pre>sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_python \$ python client_servidor_multip le.py 10000  &lt;Him&gt;: Hola servidor soc un client udp  Hola client soc un servidor python </pre>

De servidor C a client netcat UDP

Terminal	Terminal
<pre>sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_C \$ ./simple_chat_udp 10000 El port introduit és: 10000 El programa actuara com a servidor Data: Hola soc el client netcat Please enter the message: Jo el servidor udp Data: Hola servidor Please enter the message: </pre>	<pre>sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_python \$ netcat -u 192.168.1.36 10000 ^C sergi@Sergi-PC ~/projecte-tic/ASI/P8/sockets_python \$ netcat -u 192.168.1.36 10000 Hola soc el client netcat Jo el servidor udp Hola servidor </pre>