

UNIVERSITAT DE LLEIDA



Universitat de Lleida

GRAU EN ENGINYERIA INFORMÀTICA

XARXES

Pràctica 1, Programació d'aplicacions de xarxa

Autor:
Jordi Ricard Onrubia
Palacios

Professor:
Enric Guitart

21 d'abril de 2016

Resum

En el següent document es troba la informació necessària per a comprendre el funcionament del client i el servidor proposats per la pràctica. Per a la seva explicació es fan ús de diagrames d'estats on es mostra el funcionament dels protocols emprats. Per a mostrar la implantació del client i el servidor, s'explique l'estructuració juntament amb diagrames d'execució, on es mostren les funcions utilitzades i com s'executen entre elles.

Índex

1	Client	1
1.1	Protocol UDP	1
1.2	Protocol TCP	2
1.3	Estructura	3
2	Servidor	5
2.1	Protocol TCP	5
3	Anexe	6
3.1	Execució	6
3.2	Problemes i solucions trobats en el desenvolupament	6
3.3	Llistat de figures	7
4	Bibliografia	8

1 Client

1.1 Protocol UDP

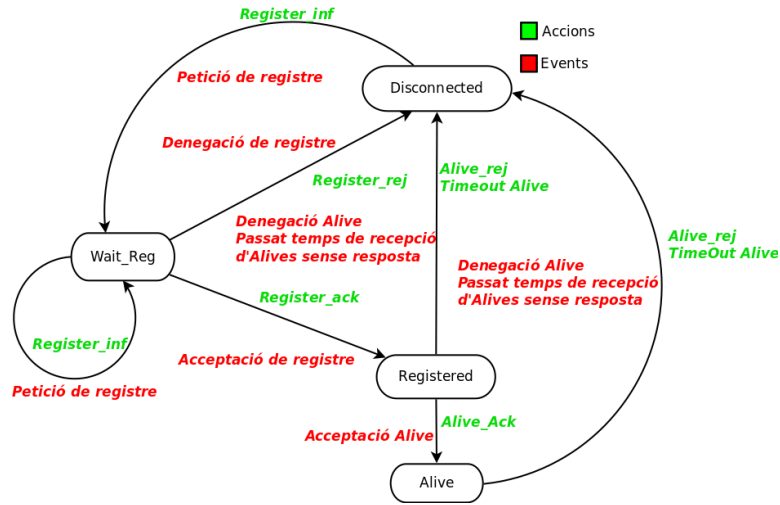


Figura 1: Protocol UDP

1.2 Protocol TCP

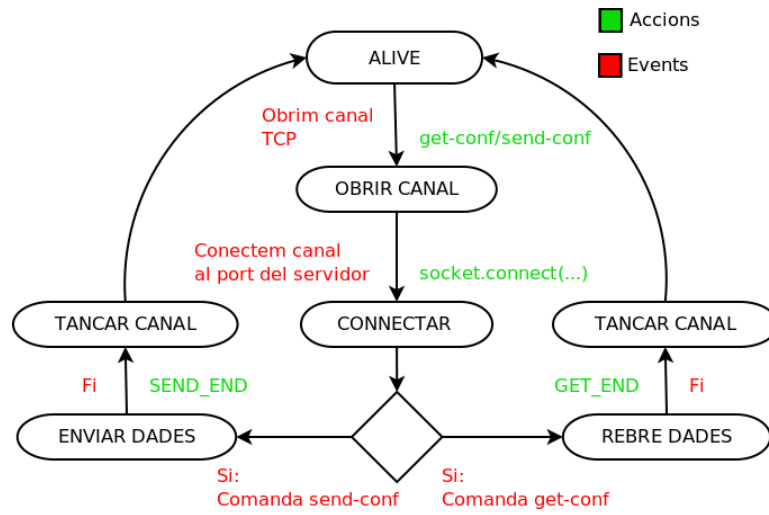


Figura 2: Client: Protocol TCP

1.3 Estructura

Programat en: Python versió 2.7.10.

El client està dividit en 2 fases diferents:

Primera fase: Registre

Per a la primera fase el client al iniciar-se fa un intent de registre al servidor mitjançant el protocol UDP, un cop finalitzant l'intent de registre passem a la següent fase.

Com es pot observar en la figura 3¹ la comunicació entre les funcions que intervenen en la primera fase són aquelles numerades de l'1 al 18.

Segona fase: Manteniment de comunicació i espera de comandes per teclat

En la segona fase intervenen processos el primer encarregat de mantenir la comunicació i el segon encarregat de rebre comandes per teclat, per tal que aquests processos facin les seves funcions previament caldrà comprovar que el registre ha estat realitzat correctament.

La comunicació entre les funcions que intervenen en la comprovació correcta del registre són aquelles numerades del 19 al 24.

La comunicació entre les funcions que intervenen en l'espera de comandes per teclat és aquella numerada amb el 25 on es crea la bifurcació entre els processos i aquelles numerades del 26 al 40.

La comunicació entre les funcions que intervenen en el manteniment de la comunicació és aquella numerada amb el 25 on es crea la bifurcació entre els processos i aquelles numerades del 41 al 45.

¹En aquest esquema no es mostren les operacions opcionals com el mode debug, el canvi de l'arxiu d'arrancada o el de configuració encara que hi estan presents, ja que només mostra les funcions principals i imprescindibles.

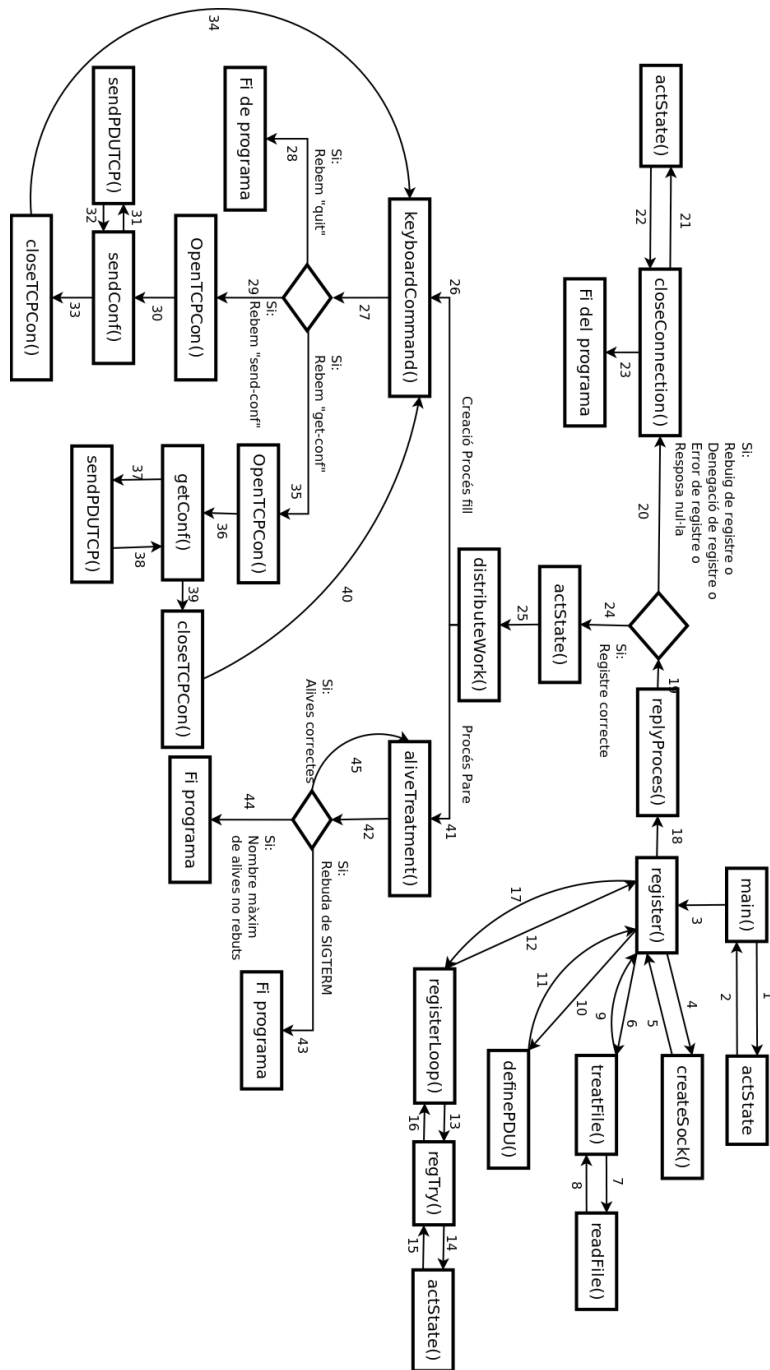


Figura 3: Client

2 Servidor

Compilat en: Linux versió 4.2.0 gcc versió 5.2.1.

Comanda utilitzada: gcc -ansi -pedantic -Wall.

El servidor no utilitza Protocol UDP més que per a controlar els estats del client per tant el protocol és el mateix que l'indicat en la figura 1

2.1 Protocol TCP

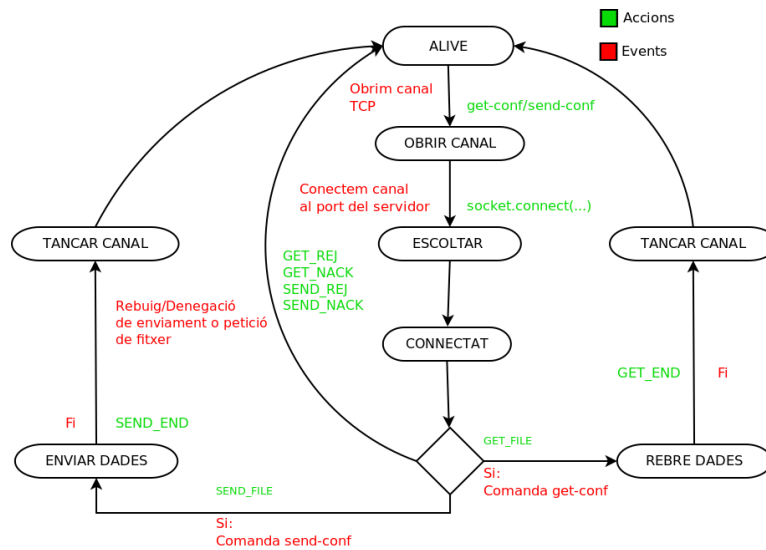


Figura 4: Servidor: Protocol TCP

3 Anexe

3.1 Execució

Client:

Execució:

- Executar el fitxer client.py de la següent manera: `python client.py`
- És important que el fitxer cons.py estigui en el mateix arxiu que client.py, ja que aquest conté totes les constants que han de ser utilitzades en l'execució del client.

Servidor:

Compilació:

- Executar el fitxer Makefile inclòs de la següent manera: `./Makefile`.
- Si el fitxer Makefile no pot ser executat per falta de permisos executar la comanda: `chmod +x Makefile`.
- És important que el fitxer server.c estigui en el mateix arxiu que el Makefile.

Execució:

- Executar el fitxer server generat per Makefile després de la seva execució de la següent manera: `./server`
- Si el fitxer server no pot ser executat per falta de permisos executar la comanda: `chmod +x server`.

3.2 Problemes i solucions trobats en el desenvolupament

Client

1. Bloqueig del programa per part del `recvfrom`, `recvfrom` és una funció què s'encarrega de rebre el missatge rebut pel socket, la solució aplicada va ser posar el timeout a 0 fent-la així no bloquejant.
2. Senyal `SIGTERM` ignorada, la solució aplicada va ser enviar-la contínuament amb un bucle infinit.
3. Llançament de excepcions per part de Python a l'hora de detectar per teclat un `ctrl-c`, la solució aplicada va ser tractar la lectura del teclat amb un `try except` ignorant les interrupcions de teclat.

Servidor

1. Problema d'accedir i actualitzar les dades dels clients des d'altres processos. La solució ha estat fer un mmap utilitzant la llibreria sys/man.h.
2. Si els dos primers "ALIVES" la temporització no arriba a agafar el tercer "ALIVE" en cas que fos enviat, ja que aquest arriba just en el moment en què el servidor ha desconnectat al client. La solució ha estat aplicar un temporitzador per tal de què el servidor només comprovi els estats dels clients cada segon en lloc de contínuament, a més a més, es permet que el temps en què haguera d'arribar "l'ALIVE" sigui més gran que l'especificat, exactament 1 segon més.

3.3 Llistat de figures

Índex de figures

1	Protocol UDP	1
2	Client: Protocol TCP	2
3	Client	4
4	Servidor: Protocol TCP	5

4 Bibliografia

Importar constants d'un altre fitxer en Python 2.
<http://zetcode.com/lang/python/packages/>
Structs en Python 2.
<https://docs.python.org/2/library/struct.html>
UDP sockets en Python 2.
<https://wiki.python.org/moin/UdpCommunication>
Funcions de la llibreria Time de Python 2.
<https://docs.python.org/2/library/time.html>
TCP sockets en Python 2.
<https://wiki.python.org/moin/TcpCommunication>
Signals en Python 2.
<https://docs.python.org/2/library/signal.html>
UDP sockets en C.
<https://www.cs.rutgers.edu/~pxk/417/notes/sockets/udp.html>
TCP sockets en C
http://www.linuxhowtos.org/C_C++/socket.htm
Compartir memòria en C.
<http://man7.org/linux/man-pages/man2/mmap.2.html>
Funcions de la llibreria Time de C.
<http://www.cplusplus.com/reference/ctime/>