



---

**XARXES**

**PRÀCTICA 1: L'APLICACIÓ ECO  
AMB SOCKETS TCP/IP**

**MEMÒRIA**

---

**Aniol Juanola Vilalta, u1978893, u1978893@campus.udg.edu, GEINF  
Jordi Badia Auladell, u1978902, u1978902@campus.udg.edu, GEINF**

**Girona, octubre de 2023**



## Continguts

1. Requisits mínims i millores .....	1
2. L'arquitectura en capes .....	1
3. La interfície aplicació-usuari .....	2
4. Els serveis de la capa d'aplicació i transport.....	2
5. Les interfícies de les capes .....	3
5.1 La interfície de la capa d'aplicació ECO .....	3
5.2 La interfície de la capa de transport TCP .....	3
5.3 Les interfícies de les capes d'Interxarxa IP i xarxa <i>Ethernet</i> .....	3
6. Els protocols de les capes d'aplicació i transport i els <i>sockets</i> TCP (estudi amb <i>Wireshark</i> i "ss") ...	3
6.1 El protocol de la capa d'aplicació d'ECO (estudi amb <i>Wireshark</i> ) .....	3
6.2 El protocol de la capa de transport TCP (estudi amb <i>Wireshark</i> ) .....	4
6.3 Els protocols de les capes d'Interxarxa IP i xarxa <i>Ethernet</i> .....	5
6.4 L'encapsulació de protocols (estudi amb <i>Wireshark</i> ) .....	5
6.5 Els <i>sockets</i> TCP de l'aplicació (estudi amb "ss") .....	6
7. Les millores .....	6
7.1 Comprovació que al S, <i>scon</i> i <i>sesc</i> són dos descriptors d'un mateix socket, i per tant tenen la mateixa adreça .....	6
7.2 El format del vostre missatge d'ECO .....	7
7.3 El format del vostre missatge d'ECO .....	7
8 Problemes i suggeriments .....	8
9 Treball en parella i dedicació .....	8
Bibliografia .....	8

En aquesta pràctica [1] s'ha dissenyat i construït l'aplicació ECO, una aplicació en xarxa (o aplicació distribuïda) inspirada en l'*Echo Protocol* [2]. Per a construir-la s'ha fet servir la interfície de sockets TCP/IP [3].

## 1. Requisits mínims i millores

Els requisits (R) mínims fets són els següents:

- R1. Construcció d'una aplicació ECO sobre TCP, seguint un model C-S.
- R2. La interfície aplicació-usuari del client (execució en un terminal).
- R3. La interfície aplicació-usuari (administrador) del servidor (execució en un terminal).
- R4. Programació en llenguatge C i estructura del codi en diversos fitxers, per separar "interfície aplicació-usuari + capa d'aplicació d'ECO" de la capa de transport TCP.
- R5. Estudi dels protocols d'aplicació i transport i dels sockets TCP.

Les millores senzilles (MS) fetes són les següents:

- MS1. Comprovació que al S, sconn i sesc són dos descriptors d'un mateix socket, i per tant tenen la mateixa @ (@IP i #port TCP)
- MS2. El format del vostre missatge d'ECO
- MS3. Missatges d'error de la interfície de sockets en 3 casos

## 2. L'arquitectura en capes

L'aplicació ECO segueix el model C-S i la seva arquitectura en forma de capes segueix el model de referència TCP/IP d'Internet, és a dir, el model de 3 capes d'Aplicació (A), Transport (T) i Xarxa (X), la darrera formada per les capes d'Interxarxa (I) i de xarxa. A la Fig. 1 es mostren les capes de l'aplicació ECO.

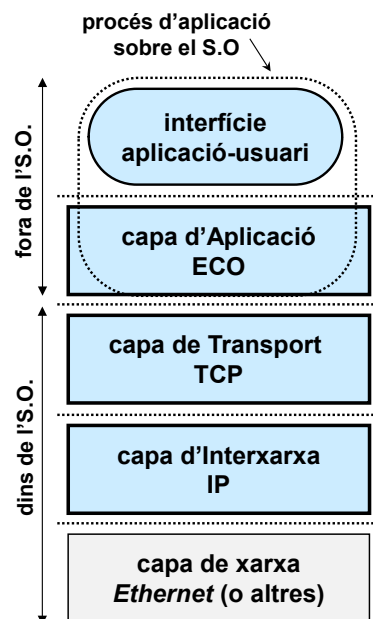


Figura 1: L'arquitectura en capes de l'aplicació ECO.

Suposarem que la capa de xarxa és *Ethernet* ja que les nostres estacions tenen una interfície *Ethernet*, però de fet podria ser qualsevol altra tecnologia de xarxa (p.e., Wi-Fi, 5G, etc.).

### 3. La interfície aplicació-usuari

La interfície aplicació-usuari tant del C com del S d'ECO són molt senzilles. De manera resumida (veieu els detalls a [1]), en el C, en un terminal, l'usuari entra per teclat una línia, un cop en rep l'eco del S la mostra a la pantalla, i així continuament fins que l'usuari indica que no n'escriurà més; en el S, en un terminal, l'administrador l'engega, i cada cop que rep una línia del C la mostra a la pantalla. A la Fig. 2 es mostra la captura de pantalla d'una execució del C-S d'ECO.

```

Terminal - xarxes@X1: ~/Desktop/shared
15.10.23-xarxes@X1:~/Desktop/shared$ ./sqr
Entra el port d'escolta del servidor: 1212
Iniciant el servidor...
Socket inicialitzat a 0.0.0.0:1212
Rebuta nova connexió
- Socket local: 10.0.2.5:1212
- Socket remot: 32.234.113.183:23735
Rebut: Prova d'execució (18 bytes)
Rebut: segona prova d'enviament del missatge (38 bytes)
Rebut: missatge curt (14 bytes)
Connexió tancada amb el client (32.234.113.183:23735)

Rebuta nova connexió
- Socket local: 10.0.2.5:1212
- Socket remot: 32.234.113.183:23735
Rebut: segona connexió! (18 bytes)
Connexió tancada amb el client (32.234.113.183:23735)

^C
15.10.23-xarxes@X1:~/Desktop/shared$

Terminal - xarxes@X1: ~/Desktop/shared
15.10.23-xarxes@X1:~/Desktop/shared$ ./cli
IP servidor: 0.0.0.0:36688
Introdueix la direcció IP a la que et vols connectar:10.0.2.4
Introdueix el port al que et vols connectar:1212
IP servidor: 10.0.2.4:1212
Entra "STOP" per a parar la transmissió de missatges
Prova d'execució
Enviats 18 bytes
ECO: Prova d'execució

segona prova d'enviament del missatge
Enviats 38 bytes
ECO: segona prova d'enviament del missatge

missatge curt
Enviats 14 bytes
ECO: missatge curt

STOP
Vols establir una altra connexió? [y/n]
IP servidor: 0.0.0.0:40128
Introdueix la direcció IP a la que et vols connectar:10.0.2.4
Introdueix el port al que et vols connectar:1212
IP servidor: 10.0.2.4:1212
Entra "STOP" per a parar la transmissió de missatges
segona connexió!
Enviats 18 bytes
ECO: segona connexió!

STOP
Vols establir una altra connexió? [y/n]
15.10.23-xarxes@X1:~/Desktop/shared$

```

Figura 2: La interfície aplicació-usuari del C i S d'ECO.

A la funció `main()` del C i del S ECO s'han fet servir dos grups de funcions, i) les relatives a la interacció entre l'usuari i l'aplicació (via teclat, pantalla, etc.) i ii) les de la interfície de la capa de transport TCP. Les del segon grup es descriuen més avall, mentre que les del primer grup es descriuen a continuació.

Les funcions del `main()` relatives a la interacció entre l'usuari i l'aplicació són les següents:

- **`printf()`, llibreria `<stdio.h>`:** Permet mostrar text per pantalla (*stdout*); s'ha utilitzat per a mostrar menús i els diferents missatges i ecos corresponents.
- **`scanf()`, llibreria `<stdio.h>`:** Permet llegir de *stdin*; s'ha utilitzat per a llegir input de l'usuari com adreces IP i ports. Llegeix fins a que troba un salt de línia o un espai.
- **`strcmp()`, llibreria `<stdio.h>`:** Permet comparar *strings* (*char\**), de forma que retorna 0 si són iguals. S'ha usat per a comprovar l'input i acabar l'enviament de missatges des de l'executable del client.
- **`strcpy()`, llibreria `<stdio.h>`:** Permet copiar un string a un altre string.
- **`snprintf()`, llibreria `<stdio.h>`:** Similar al *printf()*, permet guardar l'output a un *char\**.

### 4. Els serveis de la capa d'aplicació i transport

Els serveis d'una capa són les tasques que la capa fa, la funcionalitat que proporciona.

Els serveis de la capa d'aplicació d'ECO són:

- Permet escollir a quin servidor i a quin port es vol connectar el client.
- Permet enviar un missatge a un servidor i aquest el reenvia cap al client altra vegada per a mostrar-lo per pantalla.
- Permet tancar la connexió i obrir-ne una de nova sense reiniciar el programa.

Els serveis de la capa de transport TCP són:

- Inicialitzar un *socket* d'escolta (pel servidor).
- Inicialitzar una connexió a un servidor en mode escolta (pel client).
- Tancar la connexió actual.
- Rebre i enviar missatges.
- Donar un *socket*, retornar la IP i port del propi *socket* i del *peer* connectat.

Els serveis de les capes d'Interxarxa IP i de xarxa *Ethernet* són:

- El d'IP és portar paquets entre dues estacions d'Internet i el d'*Ethernet* portar paquets entre dues estacions d'una xarxa *Ethernet*.

## 5. Les interfícies de les capes

La interfície d'una capa són el conjunt de "mètodes" o "funcions" amb les quals es poden accedir als serveis de la capa.

En aquesta pràctica s'ha construït l'aplicació ECO a partir de la interfície de la capa de transport TCP. En la construcció, però, no s'ha definit una interfície de la capa d'aplicació ECO.

### 5.1 La interfície de la capa d'aplicació ECO

En aquesta pràctica no s'ha definit una interfície de la capa d'aplicació d'ECO.

### 5.2 La interfície de la capa de transport TCP

En lloc de fer servir la interfície "original" de *sockets* TCP, a sobre seu s'ha construït una "nova" interfície. Les funcions d'aquesta "nova" interfície de la capa TCP, tTCP, són les següents:

- **TCP\_CreaSockClient:** Donada una adreça IP i un port, crea un *socket* i el retorna.
- **TCP\_CreaSockServidor:** Donada una IP i un port, crea un *socket* d'escolta i el retorna.
- **TCP\_DemanaConnexio:** Donat un *socket*, una IP i un port, es demana una connexió mitjançant el *socket* prèviament creat i es retorna el descriptor de la connexió establerta.
- **TCP\_AceptaConnexio:** Donat un *socket* d'escolta, el servidor espera a rebre una connexió i, un cop establerta i acceptada, retorna el descriptor d'aquesta nova connexió.
- **TCP\_Envia:** Donat un descriptor, envia una seqüència de bytes pel descriptor.
- **TCP\_Rep:** Donat un descriptor, rep una seqüència de bytes pel descriptor.
- **TCP\_TancaSock:** Tanca el *socket* o descriptor passat per paràmetre.
- **TCP\_TrobaAdrSockRem:** Donat un *socket* connectat, retorna l'adreça del *socket* remot amb qui està connectat.
- **TCP\_TrobaAdrSockLoc:** Donat un *socket* connectat, retorna l'adreça local del propi *socket*.
- **T\_ObteTextRes:** Donat un codi d'error, retorna el missatge d'error que descriu el codi.
- **obtenirIpSock:** Retorna un string en format "IP:port" de l'adreça local d'un *socket* connectat.
- **obtenirIpPeer:** Retorna un string en format "IP:port" de l'adreça remota amb qui està connectat el *socket*.

### 5.3 Les interfícies de les capes d'Interxarxa IP i xarxa *Ethernet*

La interfície de la capa d'Interxarxa IP no la coneixem exactament, es troba dins del S.O., però podem suposar que seria de l'estil `IP_Envia()` i `IP_Rep()`.

La interfície de la capa de xarxa *Ethernet* tampoc no la coneixem exactament, es troba dins del S.O., però podem suposar que seria de l'estil `Eth_Envia()` i `Eth_Rep()`.

## 6. Els protocols de les capes d'aplicació i transport i els sockets TCP (estudi amb *Wireshark* i "ss")

El protocol d'una capa és el conjunt de regles que governen la comunicació, regles que defineixen el diàleg de missatges entre les parts (o entitats) d'una capa d'una manera clara i precisa. La definició d'un protocol comprèn 3 aspectes: el nom i significat dels missatges, el seu format i la seva seqüència temporal.

### 6.1 El protocol de la capa d'aplicació d'ECO (estudi amb *Wireshark*)

En aquesta pràctica no s'ha definit un protocol de la capa d'aplicació d'ECO que totes les aplicacions haguessin de complir, sinó que cadascú ha pogut decidir com fer-ho.

En el protocol d'ECO només hi ha un missatge, el que porta la línia, que anomenem LIN.

La seqüència temporal dels missatges d'ECO corresponent a l'estudi fet amb *Wireshark* (veure la captura adjunta) es mostra a la Fig. 3.

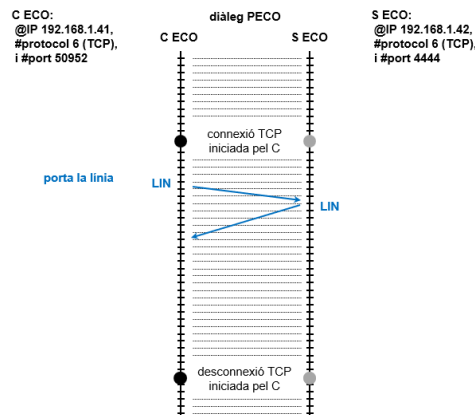


Figura 3: Seqüència temporal del protocol d'ECO.

## 6.2 El protocol de la capa de transport TCP (estudi amb *Wireshark*)

El nom i significat dels missatges TCP és el següent:

- Els paquets "SYN", "SYN+ACK" i "ACK" són els missatges de petició d'inici de connexió i resposta.
- Els paquets "PSH+ACK" i "ACK" són els missatges de transport del missatge de la capa d'aplicació.
- Els paquets "FIN", "FIN+ACK" i "ACK" són els missatges de petició de fi de connexió i resposta.

El format dels missatges TCP [4] es mostra a la Fig. 4 (el tipus de paquet s'indica amb els bits o *flags*, principalment SYN, FIN i ACK).

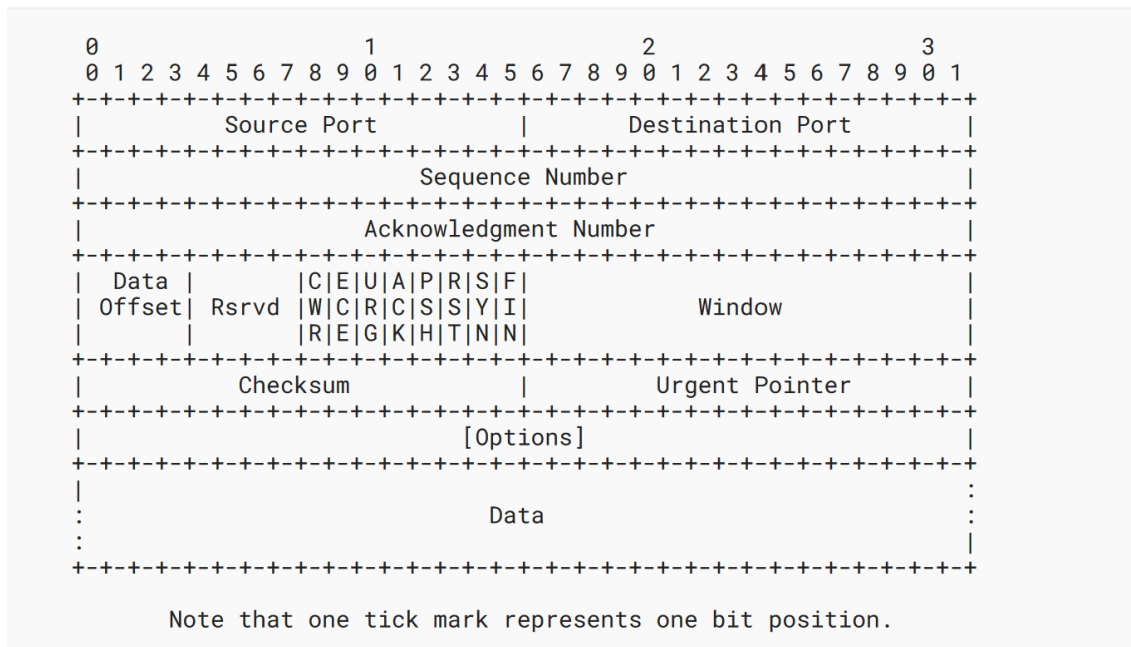


Figura 4: Format dels missatges del protocol TCP (imatge extreta de [4]).

La seqüència temporal dels missatges TCP corresponent a l'estudi fet amb *Wireshark* (veure la captura adjunta) es mostra a la Fig. 5.

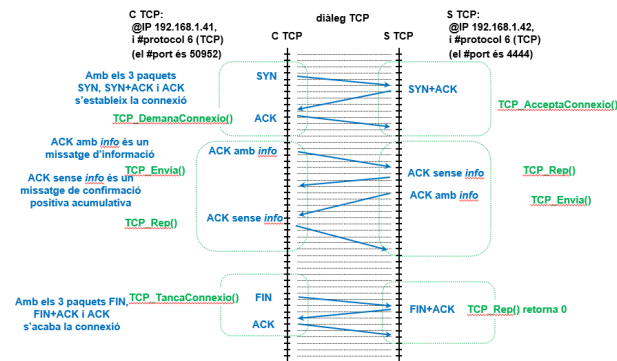


Figura 5: Seqüència temporal del protocol TCP.

### 6.3 Els protocols de les capes d'Interxarxa IP i xarxa Ethernet

En el protocol IP només hi ha un tipus de paquet, el paquet IP. El format del paquet IP versió 4 (IPv4) [5] es mostra a la Fig. 6.

Version <sub>4 bits</sub>	IHL <sub>4</sub>	DSCP <sub>6</sub>	ECN <sub>2</sub>	Total Length <sub>16</sub>	
Identification <sub>16</sub>			Flags <sub>3</sub>	Fragment Offset <sub>13</sub>	
Time To Live <sub>8</sub>		Protocol <sub>8</sub>		Header Checksum <sub>16</sub>	
Source Address <sub>32</sub>					
Destination Address <sub>32</sub>					
Options <sub>m ((IHL - 5) x 32)</sub>					
Data <sub>n (total length x 8)</sub>					

Figura 6: Format del paquet IPv4.

La seqüència temporal dels paquets IP no cal dibuixar-la: un paquet IP porta a dins un paquet TCP, és 1 a 1, i la seqüència, doncs, seria "igual" a la de TCP (però posant-hi IP).

En el protocol *Ethernet* [6] només hi ha un tipus de paquet, el paquet *Ethernet*. El format del paquet MAC d'*Ethernet* es mostra a la Fig. 7.

6 bytes	6	2	46-1500	4
Destination Address	Source Address	type / length	data	CRC

Figura 7: Format del paquet MAC d'*Ethernet*.

La seqüència temporal dels paquets *Ethernet* no cal dibuixar-la: un paquet *Ethernet* porta a dins un paquet IP, és 1 a 1, i la seqüència, doncs, seria "igual" a la d'IP (però posant-hi *Ethernet*).

### 6.4 L'encapsulació de protocols (estudi amb Wireshark)

De l'estudi fet amb *Wireshark* (veure la captura adjunta) hem escollit un dels missatges d'ECO i hem estudiat l'encapsulació dels seus protocols. Com que les línies dels missatges d'ECO són "petites", TCP no fragmenta el missatge d'ECO, de manera que només hi ha un paquet TCP d'informació. La llista de missatges de protocols i la seva longitud (escrita com "c+i", on "c" és la longitud de la capçalera i "i" la de la informació) és la següent:

- ECO LIN: 5 bytes
- TCP ACK amb *info*: 32+5 (= 37) bytes



- IP: 20+37 (= 57) bytes
- MAC d'Ethernet \*: 14+4+57 (= 75) bytes

(\* Com que Wireshark no mostra el camp CRC de 4 bytes del paquet MAC d'Ethernet, cal afegir-lo)

## 6.5 Els sockets TCP de l'aplicació (estudi amb "ss")

En el cas d'estudi, els sockets TCP de l'aplicació en diferents instants de l'execució van ser els següents:

1. **Amb el C i S apagats:** el C i el S mostren els sockets d'escolta oberts per defecte, en aquest cas TCP 22 i UDP 68 a totes les IPs (0.0.0.0).
2. **Un cop s'ha engegat el C i el S però encara no s'han connectat:** el C es manté amb l'estat "1" mentre que el servidor té un nou socket d'escolta a totes les IPs (0.0.0.0) en el port establert per l'administrador del sistema (4444 en aquest exemple).
3. **Durant l'enviament de línies:** el C té un nou descriptor TCP (established) amb adreça local 192.168.1.41 : 44107 (client) i remota 192.168.1.42 : 4444 (servidor). El S té el mateix descriptor però amb l'adreça local i remota girades, lògicament.
4. **Un cop acabada la connexió:** el C té la connexió TCP de l'estat "3" en *time-wait*, és a dir, està esperant el temps prudencial abans de tancar el socket definitivament. El S tanca la connexió de l'estat "3" i es troba a l'estat "2".
5. **Un cop s'acaba l'execució del C i del S:** el C ja ha tancat el socket en *time-wait* i es troba en l'estat inicial "1", mentre que el S té el socket d'escolta en *time-wait* i ha d'esperar el temps prudencial. Un cop passat aquest temps, es troba en l'estat "1" altra vegada.

A la Fig. 8 es mostra la captura de pantalla de l'execució de la comanda "ss" a l'instant 3 a les dues estacions.

```
15.10.23-xarxes@X1:~$ ss -natu
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port
udp UNCONN 0 0 *:68 *:*
```

Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
tcp	LISTEN	0	128	*:22	*:*
tcp	LISTEN	0	128	:::22	:::*
tcp	LISTEN	0	10	*:4444	*:*
tcp	ESTAB	0	0	192.168.1.42:4444	192.168.1.41:44107

Figura 8: Execució de la comanda ss" a l'instant 3 al servidor.

```
15.10.23-xarxes@X1:~$ ss -natu
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port
udp UNCONN 0 0 *:68 *:*
```

Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
tcp	LISTEN	0	128	*:22	*:*
tcp	LISTEN	0	128	:::22	:::*
tcp	ESTAB	0	0	192.168.1.41:44107	192.168.1.42:4444

Figura 9: Execució de la comanda ss" a l'instant 3 al client.

## 7. Les millores

En aquesta secció es descriuen les millores que s'han fet.

### 7.1 Comprovació que al S, sconn i sesc són dos descriptors d'un mateix socket, i per tant tenen la mateixa adreça

Per a comprovar que al servidor *sconn* i *sesc* són dos descriptors d'un mateix socket, hem afegit el codi que mostra per pantalla l'adreça de *sconn* i *sesc* cada cop que s'accepta una connexió nova. Es pot trobar aquest codi al fitxer "p1-serECO-M1.c".

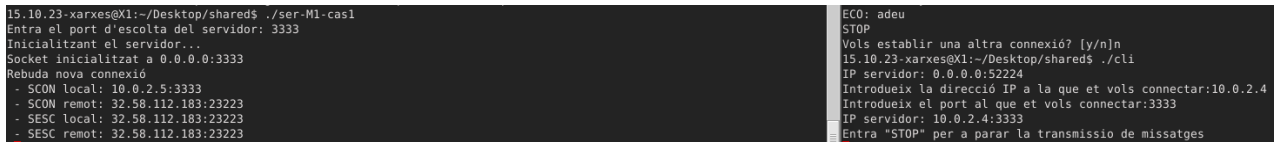
1. **Cas 1:** Com que la IP del servidor és la 10.0.2.4/24, en aquest cas es crida *TCP\_CreaSockServidor("10.0.2.4", port\_s)*, d'aquesta forma es força l'ús de l'interfície de xarxa *eth1*. Resultat de l'execució:

```
15.10.23-xarxes@X1:~/Desktop/shared$ gcc -o ser-M1-cas1 p1-serECO-M1.c p1-tTCP.c
15.10.23-xarxes@X1:~/Desktop/shared$ ./ser-M1-cas1
Entra el port d'escolta del servidor: 3333
Iniciant el servidor...
Socket inicialitzat a 10.0.2.4:3333
Rebuta nova connexió
- SCON local: 10.0.2.5:3333
- SCON remot: 32.58.110.183:22711
- SESC local: 32.58.110.183:22711
- SESC remot: 32.58.110.183:22711
```

```
link/ether 08:00:27:84:c2:f6 brd ff:ff:ff:ff:ff:ff
inet 10.0.2.5/24 brd 10.0.2.255 scope global eth2
inet6 fe80::a00:27ff:fe84:c2f6/64 scope link
valid_lft forever preferred_lft forever
15.10.23-xarxes@X1:~/Desktop/shared$ ./cli
IP servidor: 0.0.0.0:49152
Introdueix la direcció IP a la que et vols connectar:10.0.2.4
Introdueix el port al que et vols connectar:3333
IP servidor: 10.0.2.4:3333
Entra "STOP" per a parar la transmissió de missatges
```

Figura 10: Execució del cas 1 de la millora 1.

- Confirmem, doncs, que l'adreça IP i el port són els mateixos tal i com es pot veure a la imatge.
2. **Cas 2:** Aquest cas és idèntic al cas 1 perquè utilitzarà la ip de l'interfície *eth1*. Resultat de l'execució:



```

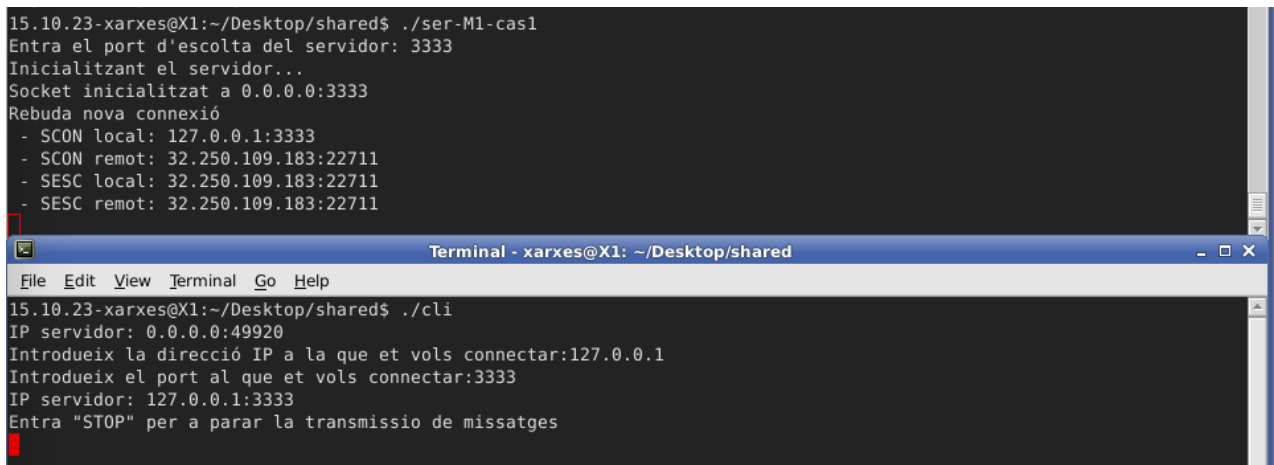
15.10.23-xarxes@X1:~/Desktop/shared$ ./ser-M1-cas1
Entra el port d'escolta del servidor: 3333
Iniciant el servidor...
Socket inicialitzat a 0.0.0.0:3333
Rebuda nova connexió
- SCON local: 10.0.2.5:3333
- SCON remot: 32.58.112.183:23223
- SESC local: 32.58.112.183:23223
- SESC remot: 32.58.112.183:23223

ECO: adeu
STOP
Vols establir una altra connexió? [y/n]
15.10.23-xarxes@X1:~/Desktop/shared$ ./cli
IP servidor: 0.0.0.0:52224
Introdueix la direcció IP a la que et vols connectar:10.0.2.4
Introdueix el port al que et vols connectar:3333
IP servidor: 10.0.2.4:3333
Entra "STOP" per a parar la transmissió de missatges

```

Figura 11: Execució del cas 2 de la millora 1.

3. **Cas 3:** Aquest cas és diferent als dos anteriors perquè utilitzarà la interfície loopback (127.0.0.1), i per tant sortiran IPs i ports iguals però diferents als dels casos anteriors. Resultat de l'execució:



```

15.10.23-xarxes@X1:~/Desktop/shared$ ./ser-M1-cas1
Entra el port d'escolta del servidor: 3333
Iniciant el servidor...
Socket inicialitzat a 0.0.0.0:3333
Rebuda nova connexió
- SCON local: 127.0.0.1:3333
- SCON remot: 32.250.109.183:22711
- SESC local: 32.250.109.183:22711
- SESC remot: 32.250.109.183:22711

Terminal - xarxes@X1: ~/Desktop/shared
15.10.23-xarxes@X1:~/Desktop/shared$ ./cli
IP servidor: 0.0.0.0:49920
Introdueix la direcció IP a la que et vols connectar:127.0.0.1
Introdueix el port al que et vols connectar:3333
IP servidor: 127.0.0.1:3333
Entra "STOP" per a parar la transmissió de missatges

```

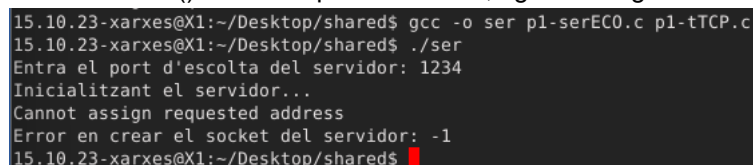
Figura 12: Execució del cas 3 de la millora 1.

## 7.2 El format del vostre missatge d'ECO

Respecte als missatges, s'envia el text llegit per part del client + el caràcter '\0' de final de línia. S'ha proposat aquesta solució per a simplificar la feina a l'hora de printar el missatge mitjançant *printf()*, que busca el caràcter de final de seqüència '\0'. Es pot veure reflectit a les línies 71 i 72 del fitxer "*p1-cliECO.c*", on es substitueix el '\n' del *read*.

## 7.3 El format del vostre missatge d'ECO

1. **Cas 1:** Quan es fa el *bind()* d'una IP que no existeix, figura el següent missatge:



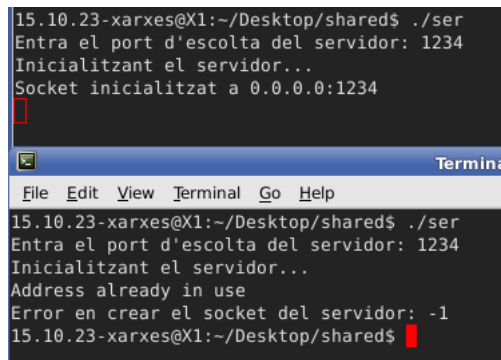
```

15.10.23-xarxes@X1:~/Desktop/shared$ gcc -o ser p1-serECO.c p1-tTCP.c
15.10.23-xarxes@X1:~/Desktop/shared$ ./ser
Entra el port d'escolta del servidor: 1234
Iniciant el servidor...
Cannot assign requested address
Error en crear el socket del servidor: -1
15.10.23-xarxes@X1:~/Desktop/shared$

```

Figura 13: Execució del cas 1 de la millora 3.

2. **Cas 2:** Quan s'executen dos servidors simultàniament en el mateix port, figura el següent missatge:

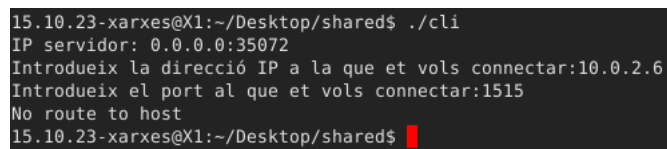


```
15.10.23-xarxes@X1:~/Desktop/shared$ ./ser
Entra el port d'escolta del servidor: 1234
Iniciant el servidor...
Socket inicialitzat a 0.0.0.0:1234

15.10.23-xarxes@X1:~/Desktop/shared$ ./ser
Entra el port d'escolta del servidor: 1234
Iniciant el servidor...
Address already in use
Error en crear el socket del servidor: -1
15.10.23-xarxes@X1:~/Desktop/shared$
```

**Figura 14:** Execució del cas 2 de la millora 3.

3. **Cas 3:** Quan el client s'intenta connectar a un servidor inexistent, figura el següent missatge:



```
15.10.23-xarxes@X1:~/Desktop/shared$ ./cli
IP servidor: 0.0.0.0:35072
Introdueix la direcció IP a la que et vols connectar:10.0.2.6
Introdueix el port al que et vols connectar:1515
No route to host
15.10.23-xarxes@X1:~/Desktop/shared$
```

**Figura 15:** Execució del cas 3 de la millora 3.

## 8 Problemes i suggeriments

No hi ha hagut problemàtiques greus en el desenvolupament de la pràctica més enllà d'errors sintàctics en el codi, o contratemps ocasionats per la falta de costum amb algunes eines de treball. L'ajuda del professor i la claredat en les seves explicacions ha ajudat a solventar aquests petits afers molt ràpid.

## 9 Treball en parella i dedicació

Hem aprofitat les hores de classe per a picar el codi i resoldre els petits problemes que ens sorgien amb l'ajuda del professor de pràctiques. L'informe l'hem redactat posteriorment a casa de forma conjunta per videotrucada.

## Bibliografia

- [1] Lluís Fàbrega, *Pràctica 1: L'aplicació ECO amb sockets TCP/IP*, curs 2023-24, UdG, 2023.
- [2] J. Postel, *RFC 862 - Echo Protocol*, 1983. Disponible a: <<https://www.rfc-editor.org/rfc/rfc862>>.
- [3] Lluís Fàbrega, *La interfície de sockets de C a UNIX*, curs 2023-24, UdG, 2023.