Universitat de Girona

XARXES

PRÀCTICA 3: L'APLICACIÓ UEB AMB *SOCKETS* TCP/IP − PART II

MEMÒRIA

Aniol Juanola Vilalta, u1978893, u1978893@campus.udg.edu, GEINF
Jordi Badia Auladell, u1978902, u1978902@campus.udg.edu, GEINF
Dimarts 15-16, GP3, Joaquim Puiggalí Tesouro

Girona, desembre de 2023

Continguts

L'aplicació UEB és una aplicació en xarxa (o aplicació distribuïda) Client-Servidor (C-S) inspirada en l'aplicació *web* "real", però amb importants simplificacions. A l'anterior pràctica [1] es van dissenyar i construir les seves parts bàsiques i en aquesta [2] s'hi han afegit noves funcionalitats, com l'ús a la interfície aplicació-usuari del C dels identificadors dels documents (incloent-hi noms DNS) o fer que el S sigui concurrent. 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. El S concurrent.
- R2. Ús de noms DNS
- R3. Ús de "doc id"

Les millores senzilles (MS) i complexes (MC) fetes són les següents:

- MS5. Acabar l'execució del S "suaument", sense ^C (CTRL+C)
- MC4. Peticions d'obtenir una "carpeta"

2 Els requisits mínims

En aquesta secció es descriuen els requisits mínims que s'han fet.

2.1 Ús de noms DNS a la interfície aplicació-usuari del C

La idea de l'ús dels noms DNS és per a simplificar el fet de recordar les adreces IP del servidor al qual et vols comunicar. Es vol aconseguir que, mitjançant un nom simple com *"ser"* es pugui accedir al servidor sense necessitat de saber la seva adreça IP. Això no és molt útil en l'entorn controlat d'aquesta pràctica, dins la xarxa NAT Network del VirtualBox, però en el món real amb servidors amb IPs dinàmiques o amb varis servidors, és interessant tenir un *nom* amb què referir-se a cadascun d'ells.

El codi consta d'una única funció afegida a la part del client (p3-aDNSc.c) anomenada **DNSc_ResolDNSaIP**(). Aquesta funció s'encarrega de, donat un array de caràcters "NomDNS", retornar un altre array que conté l'adreça IP resolta. El funcionament utilitza la crida de sistema *gethostbyname()* que fa una crida al servidor DNS configurat pel client. Aquesta crida retorna un *struct hostent*, que es converteix a un *struct in_addr* del qual n'extraiem la IP. El codi ha estat proporcionat pel professor de pràctiques en un dels documents del moodle.

Per a comprovar el funcionament del codi, simplement hem modificat el fitxer /etc/hosts de la màquina client i li hem afegit una entrada. Això ens permet modificar les entrades del DNS manualment per no haver de hostejar un servidor DNS propi. Llavors només ha calgut implementar l'altra millora senzilla (2.2) per a comprovar-ne el funcionament.

```
20.12.23-xarxes@X1:~/Desktop/shared$ cat /etc/hosts
127.0.0.1       localhost
127.0.0.1       X1

# Les @IPs de l'aula tenen assignat un nom DNS; en canvi,
# les @IPs de "casa" no. Si calgués tenir-ne, es pot fer
# com si en tinguessin fent servir aquest fitxer: només
# cal escriure aquí parelles [@IP,nom], amb les @IP de
# les estacions i els noms que es vulguin. P.e., això
# ja està fet per les 4 @IPs dels perfils "Casa PC-a",
# "Casa PC-b", etc., amb els noms PC-a, PC-b, etc.
10.100.100.101  PC-a
10.100.100.102  PC-b
10.100.100.103  PC-c
10.100.100.104  PC-d

10.0.2.5        ser
10.0.2.4        cli

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
20.12.23-xarxes@X1:~/Desktop/shared$ ./cli
Entra l'URI [STOP per aturar el client]: pueb://ser/NOEXISTEIX
Tot bé @IP: 10.0.2.5

No s'ha trobat el fitxer.Vols realitzar una altra petició? [y/n]
```

Figura 1: Contingut del fitxer /etc/hosts seguit d'una prova de l'execució.
Es pot veure com es resol la IP de ser a 10.0.2.5

2.2 Ús de "doc id" a la interfície aplicació-usuari del C

L'objectiu d'aquest requisit és entrar tot el contingut necessari per la connexió del client amb un sol string, de la mateixa forma que podríem entrar la URL per a connectar-nos a una pàgina web mitjançant un navegador.

El codi consisteix en la lectura d'un array de caràcters per *stdin* seguit d'una funció que trosseja i retorna cadascuna de les parts necessàries (sempre i quan es compleixi el format **pueb://servidor[:port]/directoris/fitxer.ext**). El port és opcional perquè s'ha establert com a conveni que el port per defecte és el 7999 (de la mateixa forma que per HTTP és el 80). Si es vol canviar, només cal introduir-lo precedit per ":". La funció que trosseja i retorna les diferents parts de l'URI s'anomena **desferURI**(...) i es troba al fitxer p3-cliUEB.c.

A la Figura 2 (a continuació) es pot veure com entrant pueb://ser/NOEXISTEIX s'obtenen les següents dades:

- El protocol és el pueb, l'específic de la pràctica. Si se n'entra un altre, el programa avisa a l'usuari.
- Es pot entrar tant una adreça IP com un hostname que el DNS resoldrà.
- Si no s'entra el port s'assumeix que és el 7999, altrament s'utilitza el port especificat.
- Després de la "/" del hostname, comença el directori d'execució.

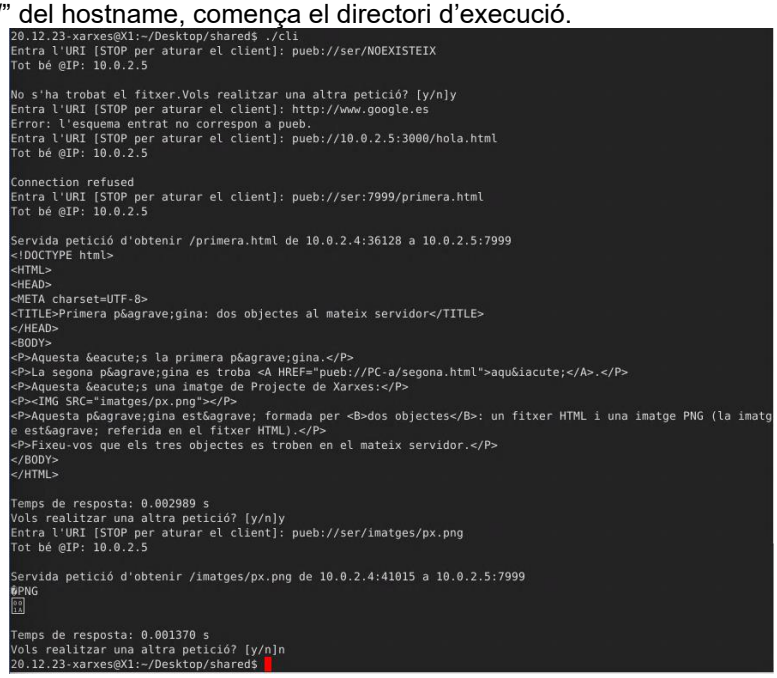

Figura 2: Exemples d'execució on es veu la feina feta pel "doc id".

2.3 El servidor concurrent

L'objectiu del servidor concurrent és poder rebre i contestar vàries connexions simultànies de clients diferents. A la pràctica anterior el servidor només podia mantenir oberta una connexió en un moment donat amb un client, la resta s'havien d'esperar a que el primer acabés. Ara el nombre de connexions simultànies queda limitat pel paràmetre *#maxconTCP* que es troba al fitxer *p3-serUEB.cfg*.

Per a implementar el servidor concurrent s'ha utilitzat la crida *select(...)* i una llista dels sockets dels quals s'espera escoltar algun missatge. En funció de si el socket pel qual es rep un missatge és *stdin* (0), el socket d'escolta de noves connexions o el socket d'una connexió en específic s'executa un codi específic:

- Si rebem text per teclat, comprovem si és STOP i en cas que ho sigui s'atura l'execució del servidor (millora opcional).
- Si rebem alguna cosa pel socket d'escolta, acceptem la connexió nova (sempre i quan el nombre de connexions actuals sigui inferior a *#maxconTCP*) i l'afegim a la llista del select.
- Si rebem alguna cosa per algun socket de connexió, responem com fèiem a la pràctica anterior.

Aquesta funcionalitat s'ha implementat amb les crides T_HaArribatAlgunaCosaPerLLegir(p3-tTCP.c) i UEBs_HaArribatAlgunaCosaPerLlegir (p3-aUEBs.c).

Per estudiar-ho s'ha fet la següent prova de funcionament:

- En el C, les connexions TCP s'ha allargat "artificialment" 20 segons.
- En una estació E1 s'ha engegat un S; en una estació E2 s'han engegat 3 Cs, C1, C2 i C3, en diferents terminals; llavors els 3 Cs han fet peticions al S (p.e., obtenir fitxers que existeixin al S) més o menys alhora.

Llavors:

- C1: Sí obté el fitxer amb un temps de resposta petit.
- C2: Sí obté el fitxer amb un temps de resposta petit.
- C3: No obté el fitxer. Per pantalla surt: S'ha produït un error amb la interfície de sockets. Que el servidor està ocupat?

A la Figura 3 hi ha una captura de pantalla que mostra la prova de funcionament dels 3 Cs i el S concurrent.

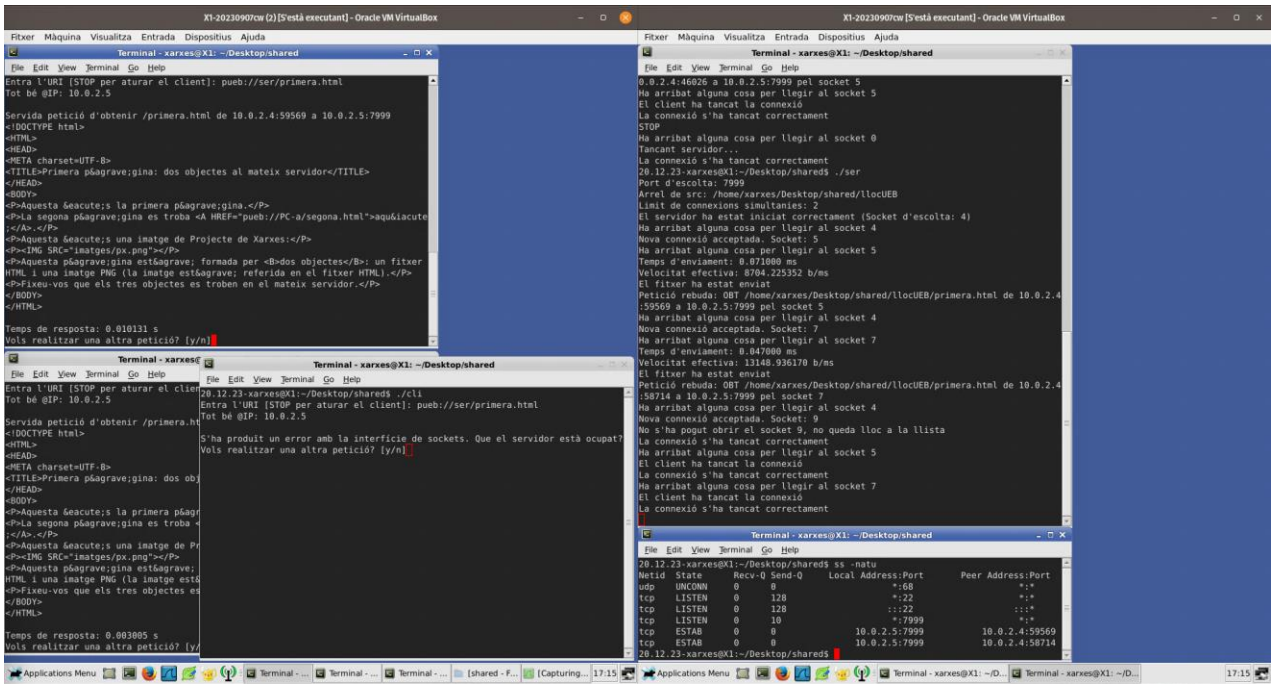

Figura 3: Prova de funcionament dels 3 Cs i el S concurrent.

Amb *Wireshark* hem fet una captura de paquets que conté les connexions TCP simultànies entre els Cs i el S (veure fitxer adjunt de captura de paquets). A la captura s'observa el següent:

- Podem veure com el client (10.0.2.4) inicia 3 connexions noves, pels ports 59569, 58714 i 38454.
- El servidor (10.0.2.5) accepta les 3 connexions.
- A les dues primeres els envia el fitxer sol·licitat (primera.html), mentre que a la tercera li tanca la connexió i li envia un paquet marcat com a RST (reset), que significa que tanca aquella connexió i no acceptarà més trànsit d'aquesta.
- Finalment, al cap de 20 segons es tanquen les connexions restants tal i com s'espera.

En el S hem fet servir la comanda de xarxa "ss" per trobar les adreces dels *sockets* dels Cs i del S. A la Figura 3 hi ha una captura de pantalla que mostra l'execució de "ss" en el S mentre les connexions TCP estan "vives".

3 Les millores

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

- MS5. Acabar l'execució del S "suaument", sense ^C (CTRL+C)
- MC4. Peticions d'obtenir una "carpeta"

3.1 Acabar l'execució del S "suaument"

L'objectiu d'aquesta millora consisteix en que el servidor llegeixi per teclat, i davant l'input "STOP", aquest tanqui tots els descriptors de fitxers i/o sockets oberts i finalitzi la seva execució.

Per dur a terme aquesta millora, s'ha afegit a la llista de sockets a escoltar el canal 0 (stdin), i s'ha donat un tractament especial quan la funció UEBs_HaArribatAlgunaCosaPerLlegir() retorna 0, com es pot veure a la següent figura.

```
else if (socket_aux == 0) { // Apagar servidor

    int input_l = read(0,buffer,1000);
    buffer[input_l-1] = '\0';

    if (strcmp(buffer, "STOP") == 0) {
        sprintf(buffer, "Tancant servidor...\n");
        escriure(buffer);
        break;
    }
    else {
        char auxi[200];
        strcpy(auxi,buffer);
        sprintf(buffer, "Comanda %s desconeguda. Intenta \"STOP\" per parar\n", auxi);
        escriure(buffer);
    }

}
```

Figura 4: Segment de codi de tractament d'inputs per stdin al servidor.

Com es pot veure a la Figura 4, es llegeixo allò que hagi arribat i si coincideix amb l'string "STOP", mostra un missatge de finalització i surt del bucle principal. Si no s'ha entrat "STOP", ho diu i segueix amb el bucle principal.

Un cop finalitzat el bucle principal, es procedeix a tancar tots els sockets i descriptors de fitxers oberts (excepte stdin):

```
// Tanca connexions
for (int i = 0; i < longLlistaSck; i++) {
    if (llistaSck[i] != 0 && llistaSck[i] != -1) {
        UEBs_TancaConnexio(llistaSck[i], text_res);
        escriure(text_res);
    }
}

free(llistaSck);

close(fd); //Fitxer log

return 0;
```

Figura 5: Finalització del programa, alliberant espai i tancant els sockets i descriptors de fitxers

La comprovació del funcionament ha estat senzilla: s'ha entrat una cadena de caràcters qualsevol, i el servidor ha mostrat el missatge d'error esperat, demanant que s'entri "STOP". S'ha entrat "STOP" i el programa ha finalitzat. Mitjançant la crida "ss -natu" es pot veure que no han quedat sockets oberts.

3.2 Peticions d'obtenir una "carpeta"

L'objectiu d'aquesta millora consisteix en distingir entre els paths finalitzats amb '/' i els que no hi acaben. Aquells que no hi acabin seran interpretats com fins ara, com a fitxers: si existeixen, s'enviaran al client, si no existeixen, se'l notificarà.

En canvi, per aquells paths finalitzats amb '/' s'entendran com a directoris, i el servidor enviarà al client el fitxer "index.html" d'aquell directori. En cas de no existir tal fitxer, s'enviarà al client el resultat de la comanda "ls -l" en aquell mateix directori.

```
if (NomFitx[strlen(NomFitx)-1] == '/') { // Si és un directori
    strcat(NomFitx,"index.html\0");
    descFitx = open(NomFitx, O_RDONLY);
    if (descFitx == -1) { // Si no existeix index.html
        char cmd[512];
        sprintf(cmd, "ls -l %s", auxStr);
        FILE *fp;

        if ((fp = popen(cmd, "r")) == NULL) { // Crida al sistema
            strcpy(TextRes, "Error obrint la pipe\n\0");
            return -4;
        }

        while (fgets(auxStr, 1024, fp) != NULL) { // Llegeix la crida per la pipe fp
            strcat(fitxer, auxStr);
            bytes_fitxer += strlen(auxStr);
        }

        fitxer[bytes_fitxer] = '\0';

        pclose(fp);
    }
    else { // Si existeix index.html
        bytes_fitxer = read(descFitx, fitxer, 10000);
        fitxer[bytes_fitxer] = '\0';
    }
}
else { // Si és un arxiu qualsevol
    descFitx = open(NomFitx, O_RDONLY);
    if (descFitx < 0 || !is_regular_file(NomFitx)) {
        ConstiEnvMis(SckCon, ERR, "No s'ha trobat el fitxer.\0", 26);
        strcpy(TextRes, "El fitxer no s'ha trobat\n\0");
        return 1;
    }
    bytes_fitxer = read(descFitx, fitxer, 10000);
    fitxer[bytes_fitxer] = '\0';
}
```

Figura 6: implementació de la millora complexa MC4

En la Figura 6 es poden veure les ramificacions condicionals de les diferents possibilitats anteriorment esmentades: la comprovació de l'últim caràcter, la execució de "ls -l" al path demanat i la cerca de index.html.

Per comprovar el funcionament de la millora s'han testejat totes les possibles combinacions dels condicionals, creant inclús un directori amb un arxiu "index.html" per veure si era enviat al client. Totes les possibilitats han estat cobertes i han estat executades donant el resultat esperat.

4 Problemes i suggeriments

Tota la pràctica s'ha desenvolupat segons els terminis preestablerts sense gaires contratemps ni problemàtiques externes a aquelles que el propi fet de programar pugui comportar (errors de sintaxis, o de noms).
La única cosa que no ha quedat totalment solucionada és que quan el servidor és ple i talla la connexió amb un nou client, el UEB del client retorna un -1 (error en la interfícies de sockets) en comptes d'un -3 (el servidor ha tancat la connexió). Hem plantejat diverses solucions però no n'hem sabut treure l'entrellat del tot.

5 Treball en parella i dedicació

Les hores de classe han estat totalment aprofitades, sortint de classe amb el requisit mínim treballat en la sessió totalment implementat.
A casa hem destinat una hora cadascú a implementar les dues millores i solució de bugs originats pels requisits mínims (treball simultani en trucada) i una hora més per escriure aquesta memòria, repartint també la càrrega de treball equitativament.

Bibliografia

[1] Lluís Fàbrega, *Pràctica 2: L'aplicació UEB amb sockets TCP/IP − Part I*, curs 2023-24, UdG, 2023.

[2] Lluís Fàbrega, *Pràctica 3: L'aplicació UEB amb sockets TCP/IP − Part II*, curs 2023-24, UdG, 2023.

[3] Lluís Fàbrega, *La interfície de sockets de C a UNIX*, curs 2023-24, UdG, 2023.