

Primera pràctica – XARXES

Aplicació de missatgeria instantània amb protocol TCP/IP

Noms: Xavier Roca Nadal i Marc Grau Genoher

DNIs: - i 45171971V

Adreça Xavier:

Adreça Marc: [u1939685@campus.udg.edu](mailto:u1939685@campus.udg.edu)

Xarxes – Enginyeria Informàtica

Grup: P.Lab-4 (Divendres 1:00 PM a 2:00 PM)

Nom del professor: Pere Vilà

Enllaç repositori: https://github.com/MARCGRAUUdG/MI\_XARXES

# Requeriments mínims i millores

Llista dels requeriments mínims i millores que heu fet, i breument com heu fet les millores en el codi.

Per començar vam fer una versió estable de Client Servidor aplicant ja una capa TCP, ja que vam veure l’esquelet i ho vam trobar prou senzill com per fer la primera setmana i vam suposar que tindriem algún problema per tant avançar feina seria el millor.

Un cop vam aconseguir una versió estable (Branca P1 de github) vam passar a fer peer to peer.

A aquest pas ens vam encallar molt i vam estar pràcticament tota la duració de la pràctica buscant els diferents problemes que teniem. Un dels que ens va portar més temps és que quan creavem el socket de servidor, li passavem un port 0 per tant ens en feia un automàticament, però llavors no podien saber els seus ports i no podiem arribar a connectar.

Després d’una tutoria vam aconseguir passar el problema i la resta van ser més fàcils de trobar, sobretot van ser errors de C, com ara punters o bé el fet que els printf sense un “\n” al final no es mostren quan toquen.

Vam aconseguir una versió estable del p2p utilitzant només el protocol TCP i aquesta està a la branca P2 del repositori.

Per últim vam fer a la vegada la interfície MI, que no ens va suposar cap problema i la codificació i descodificació dels missatges.

Aquest últim pas no ens va ser gaire difícil ja que teniem els conceptes bastant clars i tot i estar encallats amb el p2p, ja haviem anat pensant com ho fariem.

La versió estable d’aquest últim pas, i per tant la versió estable final sense millores es troba a la branca del repositori MI.

Millores: (XAVI)

# Manual d’usuari

Feu una explicació de com un usuari ha de fer servir l’aplicació

Per començar hem de tenir git baixat a l’ordinador, ens dirigim al directori on volem guardar l’aplicació i escrivim la comanda seguent a un terminal:

$ git clone https://github.com/MARCGRAUUdG/MI\_XARXES.git

(Captura carpeta clonada)

Un cop tinguem la carpeta, entrem a dins i obrim un terminal allà. Repetim el procés a una altra màquina connectada a la mateixa xarxa.

Un cop tinguem els terminals oberts, per a cada un haurem d’executar:

$ gcc -o p2p “Llista de tots els fitxers separats per un espai”

Això ens generarà un fitxer executable anomenat p2p. Per tal d’executar-lo fem:

$ ./p2p

(Captura executable)

Un cop haguem executat l’aplicació als dos terminals, cadascú introduirà el seu nickname i el port on es situarà.

Llavors el que vulgui inciar la converça entrarà l’IP del company i seguidament el seu port. El company estarà esperant al missatge d’escriure la IP sense tocar res.

Si no hi ha cap problema de xarxa, ens connectarem amb el company i ens apareixerà un missatge informatiu així com el nostre nickname i el del nostre company.

(Captura connectat correctament)

A partir d’aquí podrem mantenir una conversa normal i corrent. Els missatges que enviem apareixerant sense res més a la pantalla i els del nostre company estaràn marcats amb el seu nickname al davant.

(Captura test missatges)

Per finalitzar la conversació ho farem entrant el caràcter ‘#’ al principi de tot, independentment de si està sol o acompanyat. En cas de voler ficar un hastag al principi de tot, es pot utilitzar amb un espai al davant si no volem finalitzar la conversa.

Un cop finalitzada veurem un missatge de desconnexió i el nostre company també. També veurem un missatge que ens preguntarà tan a nosaltres com al nostre company si desitjem obrir una altra conversa, introduirem ‘S’ si ho desitgem o bé ‘N’ si volem finalitzar l’aplicació.

(Captura de finalització)

# Disseny de l’aplicació

## L’arquitectura en capes

Feu un dibuix que mostri les capes (també incloeu-hi la interfície aplicació-usuari), que indiqui quines d’aquestes capes es troben dins del S.O. i quines fora, i en aquest darrer cas, que indiqui en quins fitxers estan implementades

## Diagrama funcional

Feu un dibuix que descrigui l’estructura de l’aplicació a “alt nivell”, que mostri els principals “blocs funcionals”, units per “fletxes” que indiquin la seqüència temporal que segueixen. P.e., podríeu agafar el vostre programa principal main(), fer que cada crida a la interfície de la capa d’aplicació fos un “bloc funcional”, i representar en quin ordre s’executen, el flux que se segueix, etc., és a dir, fer una mena de representació gràfica del main().

# La interfície aplicació – usuari

Llista de funcions utilitzades per construir la interfície aplicació-usuari, és a dir, i) funcions relatives a la interacció entre l’usuari i l’aplicació (via teclat, pantalla, etc.) i ii) funcions de la interfície de la capa d’aplicació. Quant al primer grup de funcions, feu-ne aquí una breu descripció, si són d’una llibreria, dieu de quina, etc.; quant al segon grup de funcions, no en feu aquí la seva descripció sinó en la subsecció 6.1.

## Funcions relatives a la interacció entre l’usuari i l’aplicació

Degut a la existència de la interfície MI i TCP, pràcticament no vam utilitzar cap altra funció, a no ser que ens estalvies repetició de codi. A més considerem que el main, tot i ser gran, és clar i fàcil d’entendre gràcies a la seva estructura i la presència de comentaris per indicar el que pot semblar confus.

## Funcions de la interfície de la capa d’aplicació

**void obtenirIPGlobal(char \*ipGlobal)**

(XAVI)

|  |
| --- |
| **int MI\_IniciaEscPetiRemConv(char \*IPLocal, int portTCPloc);** |
|  |

|  |
| --- |
| **int MI\_HaArribatPetiConv(int SckEscMI);** |
|  |

|  |
| --- |
| **int MI\_DemanaConv(const char \*IPrem, int portTCPrem, char \*IPloc, int \*portTCPloc, const char \*NicLoc, char \*NicRem);** |
|  |

|  |
| --- |
| **int MI\_AcceptaConv(int SckEscMI, char \*IPrem, int \*portTCPrem, char \*IPloc, int \*portTCPloc, const char \*NicLoc, char \*NicRem);** |
|  |

|  |
| --- |
| **int MI\_HaArribatLinia(int SckConvMI);** |
|  |

|  |
| --- |
| **int MI\_EnviaLinia(int SckConvMI, const char \*Linia);** |
|  |

|  |
| --- |
| **int MI\_RepLinia(int SckConvMI, char \*Linia);** |
|  |

|  |
| --- |
| **int MI\_AcabaConv(int SckConvMI);** |
|  |

**int MI\_AcabaEscPetiRemConv(int SckEscMI);**

# Les capes d’aplicació i transport – Serveis

Per cada capa, feu una breu llista de les seves tasques

## Serveis de la capa d’aplicació de MI

Mitjançant els serveis de la capa TCP, la interfície MI actua amb un programa principal P2P per tal de proporcionar una comunicació entre dos terminals, siguin del mateix dispositiu o bé des de dos de diferents.

Hem de tenir en compte que aquests dos dispositius sempre han d’estar a la mateixa xarxa.

## Serveis de la capa de transport TCP

La capa TCP és l’encarregada d’efectuar les funcions més bàsiques de l’aplicació i a la vegada les de més “Baix nivell”. Aquesta faria la creació de sockets, la connexió entre els dos i l’enviament i rebuda de paquets d’un a l’altre.

Com es fa ja seria feina de la capa MI o el programa principal.

# Les capes d’aplicació i transport – Interfície

Per cada capa, feu una breu llista de les seves funcions (són les funcions “externes”) acompanyades per una descripció.

## Interfície de la capa d’aplicació MI

|  |
| --- |
| **int MI\_IniciaEscPetiRemConv(char \*IPLocal, int portTCPloc);** |
| Crea un socket servidor o espera a rebre alguna cosa. |
|  |

|  |
| --- |
| **int MI\_HaArribatPetiConv(int SckEscMI);** |
| Crea una llista de sockets. A la primera posició es guarda un 0 que és el canal d’entrada de teclat i a la posició 1 el socket del company per poder escoltar-lo. Seguidament crida la funció T\_HaArribatAlgunaCosa per escoltar o el teclat o el socket del company. |
|  |

|  |
| --- |
| **int MI\_DemanaConv(const char \*IPrem, int portTCPrem, char \*IPloc, int \*portTCPloc, const char \*NicLoc, char \*NicRem);**  Es crida quan un terminal vol demanar una connexió a un altre. Aquesta crea un socket client, demana la connexió al terminal, troba l’ adreça locals i per últim envia el nickname local i rep el remot. |
|  |

|  |
| --- |
| **int MI\_AcceptaConv(int SckEscMI, char \*Iprem, int \*portTCPrem, char \*Iploc, int \*portTCPloc, const char \*NicLoc, char \*NicRem);**  Quan una terminal remot demana connexió, si la volem començar es crida aquesta funció. Aquesta accepta la connexió demanada i com la anterior troba l’adreça local, rep el nickname remot i envia el local. |
|  |

|  |
| --- |
| **int MI\_HaArribatLinia(int SckConvMI);** |
| Molt similar a la funció **MI\_HaArribatPetiConv**, però aquesta espera a rebre un missatge i no un nickname, sigui pel canal del socket remot o bé pel canal d’entrada de teclat. |
|  |

|  |
| --- |
| **int MI\_EnviaLinia(int SckConvMI, const char \*Linia);** |
| A través de la funció **TCP\_Envia**, envia un missatge passat per paràmetres al socket passat per paràmetres també. Retorna la mida del missatge. |
|  |

|  |
| --- |
| **int MI\_RepLinia(int SckConvMI, char \*Linia);** |
| A través de la funció **TCP\_Rep**, rep un missatge enviat per la terminal remota i retorna la mida del missatge. El missatge es passa a la interfície MI per paràmetres. |
|  |

|  |
| --- |
| **int MI\_AcabaConv(int SckConvMI);** |
| Tenca el sockets passats per paràmetre des de la interfície MI i finalitza la connexió. |
|  |

**int MI\_AcabaEscPetiRemConv(int SckEscMI);**

Tenca els sockets i acaba amb l’escolta de peticions remotes.

## Interfície de la capa de transport TCP

|  |
| --- |
| **int TCP\_CreaSockClient(const char \*IPloc, int portTCPloc);** |
|  |

|  |
| --- |
| **int TCP\_CreaSockServidor(const char \*IPloc, int portTCPloc);** |
|  |

|  |
| --- |
| **int TCP\_DemanaConnexio(int Sck, const char \*IPrem, int portTCPrem);** |
|  |

|  |
| --- |
| **int TCP\_AcceptaConnexio(int Sck, char \*IPrem, int \*portTCPrem);** |
|  |

|  |
| --- |
| **int TCP\_Envia(int Sck, const char \*SeqBytes, int LongSeqBytes);** |
|  |

|  |
| --- |
| **int TCP\_Rep(int Sck, char \*SeqBytes, int LongSeqBytes);** |
|  |

|  |
| --- |
| **int TCP\_TancaSock(int Sck);** |
|  |

|  |
| --- |
| **int TCP\_TrobaAdrSockLoc(int Sck, char \*IPloc, int \*portTCPloc);** |
|  |

|  |
| --- |
| **int TCP\_TrobaAdrSockRem(int Sck, char \*IPrem, int \*portTCPrem);** |
|  |

|  |
| --- |
| **int T\_HaArribatAlgunaCosa(const int \*LlistaSck, int LongLlistaSck);** |
|  |

**char\* T\_MostraError(void);**

# Les capes d’aplicació i transport – Protocols

Per cada capa, descriviu el seu protocol, és a dir, el nom, el significat, el format i la seqüència temporal dels seus missatges. Indiqueu també la relació entre els missatges i la interfície de la capa. Quant a les seqüències temporals, feu-les en un cas concret, el descrit a la planificació (secció 7, 5a setmana), de l’estudi dels protocols d’aplicació i transport de l’aplicació de MI, amb l’analitzador de protocols Wireshark i la comanda de xarxa netstat. Feu servir la “plantilla” PowerPoint que es troba al Moodle de l’assignatura per dibuixar-les, i indiqueu-hi també la relació entre els missatges i la interfície de la capa.

## Protocol de la capa d’aplicació MI (PMI)

Nom, significat, format i seqüència temporal dels missatges PMI (en l’estudi anterior), i la relació entre els missatges PMI i la interfície de la capa MI.

## Protocol de la capa de transport TCP

Nom, significat, format i seqüència temporal dels missatges TCP (en el mateix estudi anterior), i la relació entre els missatges TCP i la interfície de la capa TCP. Dieu també com lliga l’anterior seqüència de missatges PMI amb la seqüència de missatges TCP.

## Estudi amb netstat

Amb netstat mostreu les adreces dels socketsdels 2 processos (en el mateix estudi anterior) en diferents instants (a l’inici sense conversa, en conversa, s’acaba la conversa, s’acaba l’execució, esperant prou temps), i compareu-les amb les que mostra l’aplicació per pantalla, i les observades en els paquets capturats amb el Wireshark.

# Problemes i sugeriments

Dificultats que han aparegut durant la realització d’aquesta pràctica (i si s’han solucionat i com). Suggeriments als professors a fi de millorar aquesta pràctica.

# Treball en parella i dedicació

Expliqueu com heu treballat de manera conjunta per fer la pràctica. Dieu les hores de classe (és a dir, 6h) i les hores fora de classe que heu necessitat(hores individuals per persona, és a dir, no sumeu les de la parella d’estudiants).