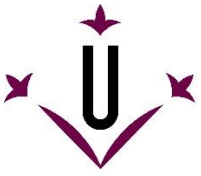
Pràctica 1.
PROGRAMACIÓ D'APLICACIONS DE XARXA

Júlia Nogales López

Xarxes
Grau en Enginyeria Informàtica
2023/2024



Universitat de Lleida
Escola Politècnica Superior

Índex

Índex de figures

1. Introducció

Per la realització d'aquesta pràctica s'ha implementat un sistema de comunicació basat en el model client-servidor, on el client actua com a controlador dels sensors i el servidor adquireix les funcionalitats d'un equip central per gestionar i controlar les peticions de tots els clients.

El client ha estat programat en Python 3.9 i el servidor en Ansi C (C99 + la llibreria pthread). Les funcionalitats implementades tant en el **client** com en el **servidor** han estat la subscripció, el manteniment de la comunicació i la introducció de comandes.

L'estructura del client i del servidor, juntament amb l'estructura del codi amb els seus mètodes implementats, està representada en els següents diagrames de blocs, respectivament. En negre estan representats els diferents estats pels que es troba el client/controlador durant l'execució del programa, en blau els mètodes que s'executen mentre es troba en l'estat de la mateixa alçada, i en taronja les execucions paral · leles amb un *thread*.

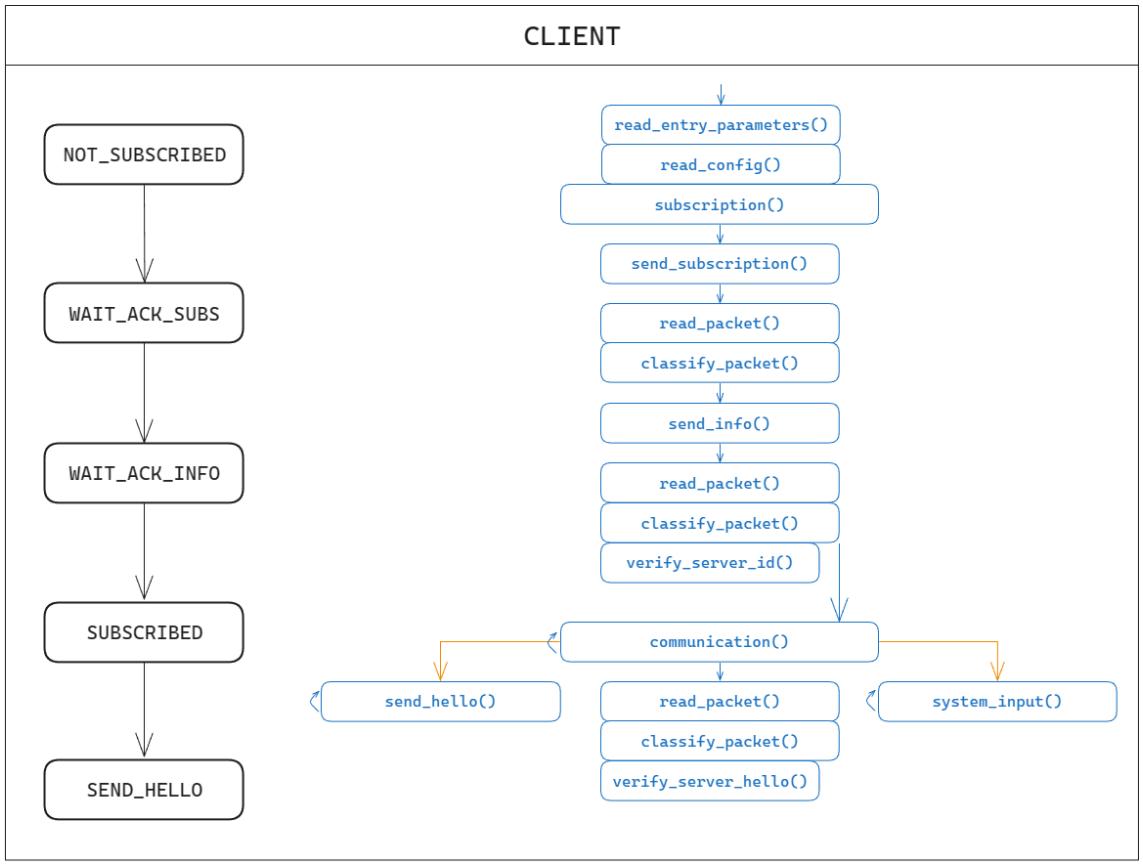


Figura 1. Diagrama de l'estructura del client

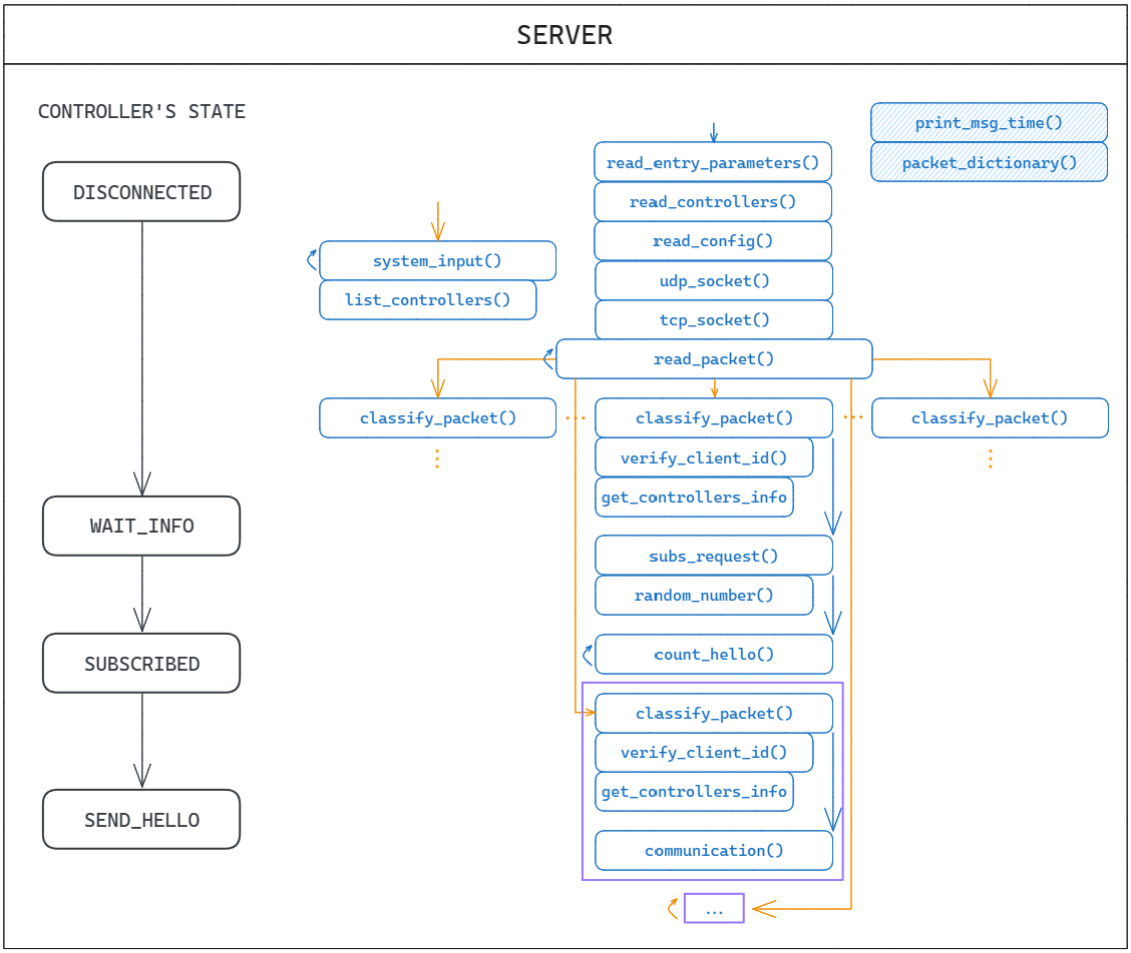


Figura 2. Diagrama de l'estructura del servidor

2.Implementació Client

Inicialment, el client executa la funció *read_config()*, la qual llegeix el fitxer on està emmagatzemada la configuració d'aquest, per defecte sota el nom de "client.cfg" (tot i que es pot passar com a paràmetre en executar el programa mitjançant la comanda `-c <nom_fitxer>`), i que conté la informació necessària per establir connexió amb el servidor. També permet entrar la comanda debug (`-d`), que dona informació addicional sobre la comunicació entre client-servidor i l'estat del client.

2.1. Subscripció

Un cop tractats els paràmetres i la configuració inicial, el client, que es troba en estat NOT_SUBSCRIBED, inicia la comunicació amb el servidor enviant-li una petició de subscripció (SUBS_REQ) emprant el protocol UDP. Aquesta petició inclou la seva adreça MAC i un nombre aleatori inicialitzat a zeros. Un cop el client fa la petició de registre es manté en espera (WAIT_ACK_SUBS) per la resposta

del servidor, que pot ser d'acceptació (SUBS_ACK), de rebuig (SUBS_REJ) o d'error (SUBS_NACK).

El client també pot enviar-li paquets de negació al servidor si durant el procés comprova que les dades d'identificació d'aquest són errònies mitjançant la funció *verify_server_id()*.

Si les dades són correctes i està autoritzat, el client és acceptat, envia un altre paquet (SUBS_INFO) per verificar les dades (on ha d'adjuntar el número aleatori prèviament generat pel servidor) mitjançant *send_info()* i actualitza el seu estat a WAIT_ACK_INFO. Finalment, si és verificat i acceptat pel servidor mitjançant el paquet INFO_ACK, el client passa a l'estat SUBSCRIBED i es completa la fase de subscripció.

Si durant aquest procés la resposta del servidor és negativa, automàticament retornarà a l'estat NOT_SUBSCRIBED i reiniciarà el procés de subscripció. En canvi, si no rep resposta del servidor reiniciarà el procés de subscripció fins que es completin 'o' intents de subscripció. Superats 'o' processos sense resposta, el client finalitzarà l'execució.

En la figura següent es mostra l'esquema del procés de subscripció del client amb els respectius canvis d'estat segons els paquets rebuts i enviats al servidor, marcat en vermell les situacions alternatives al procediment correcte/esperat:

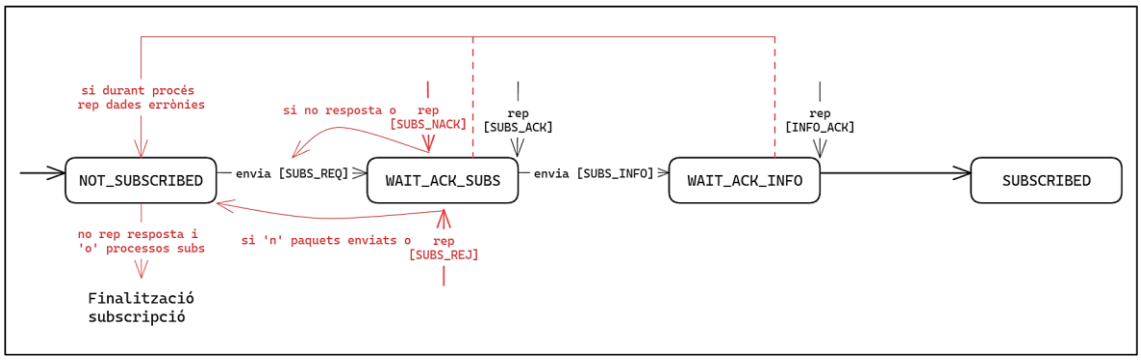


Figura 3. Diagrama d'estats del procés de subscripció del client

2.2. Manteniment de la comunicació

Quan el client ja ha completat la fase de subscripció, aquest passa a mantenir comunicació periòdica amb el servidor. Per fer-ho, envia cada "v" segons un paquet de manteniment de la comunicació (HELLO) mitjançant un thread, mentre que en el programa principal espera a que el servidor respongui constantment amb el mateix tipus de paquet per verificar que no s'ha perdut la connexió.

Per rebre el primer paquet s'inicia un *timeout* del temps indicat (v * r) on, en cas de no rebre el paquet en el temps desitjat, el programa "fingeix" que ha rebut un paquet desconegut (*NotHello*) per passar a l'estat NOT_SUBSCRIBED i reiniciar el procés de subscripció.

Si efectivament aconsegueix rebre el primer paquet HELLO del servidor, el client actualitza el seu estat a SEND_HELLO, inicialitza un altre thread per escoltar comandes entrades per consola i es manté així fins que hi hagi alguna negativa per part de qualsevol dels dos missatgers. Si qualsevol deixa de transmetre aquest paquet, ja sigui perquè envia el paquet de rebuig (HELLO_REJ) o perquè el servidor deixa de transmetre durant 's' paquets (*strike_hello_miss*), es cancel · larà el procés de comunicació, el client passarà a l'estat NOT_SUBSCRIBED i altre cop iniciarà el procés de subscripció.

En el següent diagrama estan representades totes les etapes que es podrien dur a terme durant el procés de la comunicació periòdica entre client i servidor des del punt de vista del client:

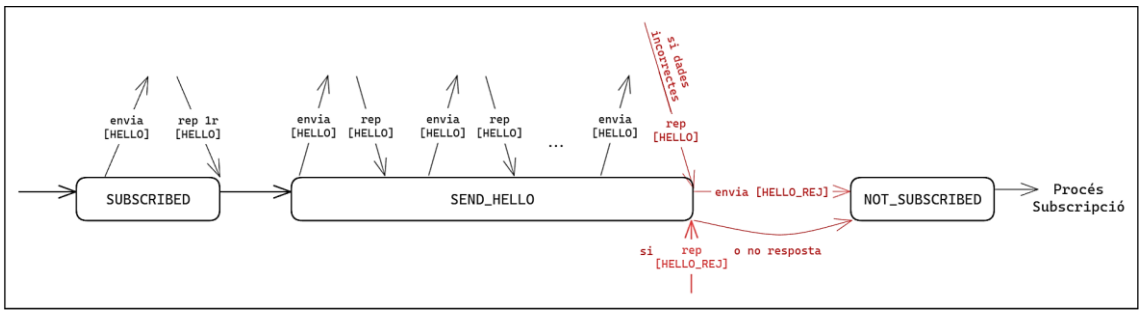


Figura 4. Diagrama d'estats del manteniment de la comunicació del client

2.3. Introducció de comandes

Tot i que ja s'ha comentat breument en l'inici de l'apartat "Implementació Client", s'ha gestionat la introducció de comandes per consola, tant en el moment d'executar el programa (`-d` i `-c <nom_fitxer>`) com durant tot el programa. Per poder "escoltar" qualsevol entrada que es pugui dur a terme durant el programa, s'ha creat un thread que enregistra tots els inputs per consola (*system_input()*) i, segons la comanda entrada (`stat`, `set` o `quit`), mostra les dades del controlador, les modifica o tanca el programa, respectivament.

3.Implementació Servidor

Pel que respecta a la implementació del servidor, la diferència principal amb el client, sense tenir en compte el llenguatge de programació, ha estat la utilització de threads per poder gestionar més d'un client que intenta comunicar-se, és a dir, obrir un procés nou per cada petició entrada d'un controlador diferent (o d'un nou paquet HELLO).

Igual que en el client, inicialment el servidor també ha de llegir els arxius de configuració propis ("server.cfg") i dels controladors registrats ("controllers.dat") que té per defecte o els que s'introdueixen per consola al executar el programa amb la comanda `-c <nom_fitxer_config>` i `-u <nom_fitxer_controladors>`, respectivament. A més, si es desitja activar el mode debug per rebre més informació de l'execució del programa es pot utilitzar la comanda `-d` per visualitzar-ho.

Un cop inicialitzat el programa es creen els sockets UDP i TCP (tot i que el protocol TCP no s'ha implementat) per, posteriorment, poder tenir un canal de comunicació amb els clients.

3.1. Subscripció

El procés de subscripció és individual per cada client i s'inicia només quan un controlador envia una petició de subscripció al servidor (SUBS_REQ). El servidor rep totes les peticions, ja que des de que s'inicia la seva execució, *read_packet()* ja es troba en un bucle infinit (`while(true)`) per esperar peticions dels clients.

Quan el servidor rep un paquet crea un nou thread que s'utilitzarà per gestionar tot l'enviament de paquets amb aquest client fins arribar a la fase de comunicació periòdica, que es crearà un nou procés per cada paquet UDP rebut. En rebre el primer paquet del client, SUBS_REQ, comprova que el controlador estigui registrat i en estat DISCONNECTED. Un cop verificat que totes les dades són correctes, crea un nou port UDP (que és el que s'utilitzarà en endavant per la comunicació amb aquest controlador), genera un número aleatori per verificar la identitat del controlador en els futurs paquets rebuts, li envia el paquet d'acceptació i actualitza l'estat del controlador a WAIT_INFO.

Igual que en el cas del client, el servidor espera rebre el paquet de resposta (SUBS_INFO) en un màxim de temps determinat ("s" cops el temps inicial d'enviament). Si no es rep aquest paquet, enviarà un SUBS_REJ i finalitzarà el procés de subscripció d'aquell controlador, actualitzant el seu estat a DISCONNECTED. Si el rep dins del temps estipulat, li retornarà l'acceptació del paquet (INFO_ACK), canviarà l'estat a SUBSCRIBED i es donarà per acabada la fase de subscripció.

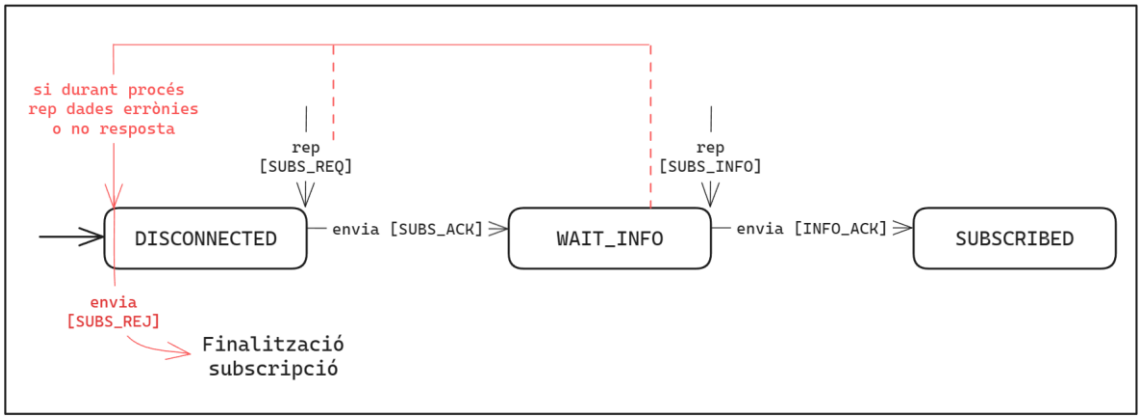


Figura 5. Diagrama d'estats del procés de subscripció del servidor

3.2. Manteniment de la comunicació

Al final de la funció *subs_request()*, després de canviar-li l'estat al controlador a SUBSCRIBED, es crida al mètode *count_hello()*, encarregat de comprovar que arriba el primer HELLO en el temps estipulat (v * r) i que arriben els següents paquets abans de 'x' "strikes", és a dir, 'x' "timeout" (v) completats sense rebre HELLO. Si no es compleix qualsevol de les dues situacions, s'actualitza l'estat del controlador a DISCONNECTED i es cancel · la el procés.

Per l'altra banda, per poder rebre i enviar els paquets HELLO al controlador corresponent s'ha implementat la funció *communication()*, que es crida des del procés principal del servidor (*read_packet()*) un cop comprovat que el paquet i les seves dades són correctes mitjançant la funció auxiliar *verify_client_id()*. Si no ho són, s'executa *send_hello_rej()* per enviar un paquet HELLO_REJ al client i suspendre el procés de comunicació.

Les úniques funcions de *communication()* son enviar el paquet HELLO al client corresponent i actualitzar el valor "hello" del controlador a 'true' per indicar a *count_hello()* que s'ha rebut un HELLO. Posteriorment, *count_hello()* (després de cada *sleep()*), comprova que el valor "hello" estigui a 'true' i l'actualitza a 'false'.

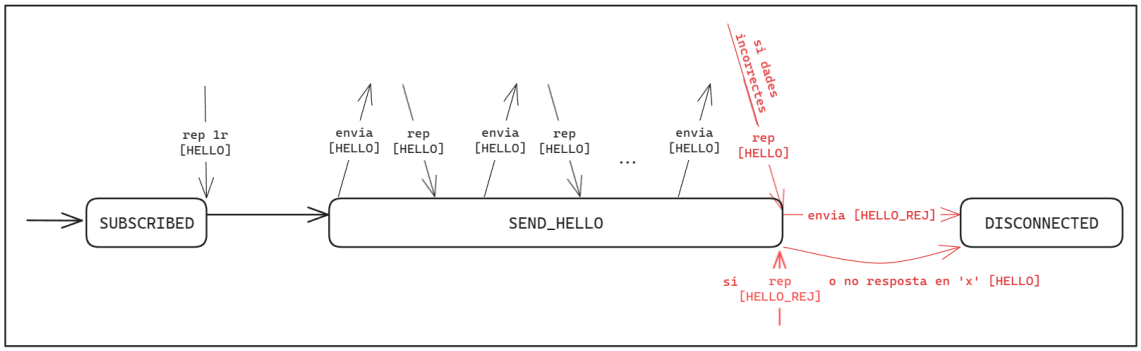


Figura 6. Diagrama d'estats del manteniment de la comunicació del servidor

3.3. Introducció de comandes

Pel cas del servidor, les comandes a implementar que s'introdueixen per consola durant l'execució del programa han estat `list` i `quit`, on "list" mostra la llista dels controladors i el seu estat actual i "quit" tanca el programa.