Xarxes: Pràctica 1, Programació d'aplicacions de xarxa

César Fernàndez, Enric Guitart, Carles Mateu

Febrer 2020

Finalitat de la pràctica:

- Aprendre a programar aplicacions de xarxes
- Entendre el model client-servidor
- Aprendre a programar i dissenyar un protocol de comunicacions

Introducció

En els processos industrials cada cop és més freqüent que el maquinari proporcioni múltiples dades sobre el procés de fabricació. Aquesta informació s'emprava inicialment per la supervisió i control del propi maquinari. Combinant les dades de múltiples fonts es va passar a poder proporcionar dades de traçabilitat dels productes. I actualment la «Indústria 4.0» proposa la transformació dels processos productius mitjançant la digitalització de les fabriques.

En aquesta pràctica es desenvoluparà un protocol de comunicacions per un sistema de lectura i emmagatzematge de dades de fabricació emprant el model client-servidor. Els clients seran els dispositius encarregats de realitzar les mesures en els diferents punts de fabricació i el servidor serà un equip informàtic destinat a rebre les dades dels clients i emmagatzemar-les per el seu tractament.

En les següents seccions es descriuran les característiques i funcions de cadascun dels actors del sistema.

> **Informació**
>
> **Traçabilitat**: Conjunt de mesures, accions i procediments que permeten enregistrar i identificar cada producte des del seu origen fins el seu destí. Permet seguir el "rastre"d'un producte en totes les etapes de producció, transformació i distribució.

Client

Com s'ha dit abans, un client serà qualsevol dispositiu capaç de proporcionar informació dels processos de fabricació. En aquests dispositius s'instal·larà el software client que ha de permetre la comunicació amb el servidor emprant un protocol de comunicacions. Per la pràctica, el software client s'executarà en el PC simulant un dispositiu de mesures.

Genèricament un client ha de realitzar quatre tasques:

- Registrar-se en el servidor
- Mantenir comunicació periòdica amb el servidor
- Enviar les dades al servidor
- Esperar connexions TCP del servidor per rebre o enviar informació

> **Informació**
>
> **Indústria 4.0**: També conegut com la "Quarta Revolució Industrial"o Ïndustria Intel·ligent"es pot definir com el conjunt de tècniques que automatitzen la producció i aporten informació en temps real. La majoria de les tècniques emprades pertanyen al mon de les Comunicacions, la Intel·ligència Artificial i la Robòtica : *Cloud Computing*, *Big Data*, *Data Mining*, *Data Analytics*, *Internet of Things* , *Machine Learning*, *Deep Learning* , *Collaborative Robot*

Donat que el client se simularà en un PC, caldrà una cinquena tasca que consistirà en l'espera de comandes per la consola. Amb aquestes comandes se simularan les lectures de dades, els seus enviaments cap al servidor i algunes operacions de control.

En cada client caldrà un arxiu de configuració amb dades d'identificació del dispositiu i les dades necessàries per la comunicació inicial amb el servidor. Per defecte aquest arxiu s'anomenarà `client.cfg` i estarà emmagatzemat en el mateix directori que el software client. Per poder simular múltiples clients en un un sol PC s'ha de

poder especificar el nom de l'arxiu de configuració del client en endegar el software client emprant l'opció *-c* (`-c <nom_arxiu>`). L'arxiu contindrà un paràmetre de configuració per línia i en cada línia s'especificarà el nom del paràmetre i el seu valor separats per el signe d'igualtat (=).

Els paràmetres que ha de contenir l'arxiu són:

- *Id*: Identificador del dispositiu (alfanumèric de 12 dígits)
- *Elements*: Elements del dispositiu. S'empra un codi alfanumèric de 7 dígits per identificar cada element i els diferents elements del dispositiu se separen per punt i coma (;). El codi té el format XXX-Y-Z, on:
  - XXX: Magnitud de l'element (TEM, PRE, HUM, LUM, LEC, PRS, AUD, VID, PIC, TXT, TIC, TIM, STA,..)
  - Y: Ordinal per diferenciar magnituds iguals
  - Z: Tipus d'element: I → entrada (actuador), O → sortida (sensor).

  Per cada element caldrà definir una variable que emmagatzemi el seu valor. Cada element pot emmagatzemar un valor alfanumèric d'una mida màxima de 15 bytes. Un dispositiu pot tenir com a màxim 5 elements.
- *Local-TCP*: Port TCP (**T***ransport* **C***ontrol* **P***rotocol*) local per la transferència de dades amb el servidor.
- *Server*: Nom o adreça IP (**I***nternet* **P***rotocol*) del servidor.
- *Server-UDP*: Port UDP (**U***ser* **D***atagram* **P***rotocol*) del servidor.

Registrar-se en el servidor

La primera tasca que ha de dur a terme un client del sistema és registrar-se en el servidor i fins que no finalitzi satisfactòriament el client no efectuarà cap altra tasca.

Per la comunicació d'aquesta tasca s'emprarà el protocol UDP amb la PDU (**P***rotocol* **D***ata* **U***nit*) que es mostra en la figura 1.

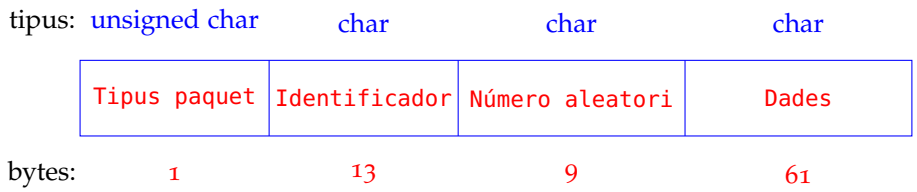

Figura 1: Format PDU UDP

Per la fase de registre s'han definit els tipus de paquets de la taula 1.

Valor	Mnemònic	Significat
0x00	REG_REQ	Petició de subscripció
0x01	REG_INFO	Paquet addicional de subscripció
0x02	REG_ACK	Acceptació de paquet de subscripció
0x03	INFO_ACK	Acceptació del paquet addicional de subscripció
0x04	REG_NACK	Error en paquet de petició de subscripció
0x05	INFO_NACK	Error en paquet addicional de subscripció
0x06	REG_REJ	Rebuig de subscripció

Taula 1: Tipus de paquet fase de registre

En la taula 2 s'indiquen els estats en els que pot estar un component del sistema (servidor o client) durant el procés de registre i el manteniment de comunicació periòdica (protocol UDP).

Valor	Mnemònic	Significat
0xa0	DISCONNECTED	Dispositiu desconnectat
0xa1	NOT_REGISTERED	Dispositiu connectat i no registrat
0xa2	WAIT_ACK_REG	Espera confirmació primer paquet registre
0xa3	WAIT_INFO	Servidor esperant paquet REG_INFO
0xa4	WAIT_ACK_INFO	Espera confirmació segon paquet registre
0xa5	REGISTERED	Dispositiu registrat, sense enviar ALIVE
0xa6	SEND_ALIVE	Dispositiu enviant paquets de ALIVE

Taula 2: Estats d'un client

INFORMACIÓ

Els estats d'un client o servidor són els que s'empraran per construir el diagrama d'estats del protocol.

Procediment de registre

- El client partirà de l'estat `NOT_REGISTERED`, haurà d'enviar una petició de registre [`REG_REQ`] i esperar resposta del servidor.
- En enviar el primer paquet [`REG_REQ`] el client passarà a estat `WAIT_ACK_REG`.
- Si no rep resposta del servidor en t segons tornarà a enviar la petició de registre.
- Per evitar saturar la xarxa, el temps t entre paquets s'anirà variant:
  - L'interval d'enviament dels primers p paquets és t.
  - Per cada paquet posterior a p l'interval d'enviament s'incrementarà en t segons fins arribar a $q * t$ segons a partir dels quals l'interval d'enviament es mantindrà constant en aquest valor.
  - Si després d'enviar n paquets no s'ha completat el procés de registre, s'espera u segons i s'inicia un nou procés de registre.
  - Si passats o processos de registre no s'ha finalitzat satisfactòriament el procés de registre, el client finalitzarà indicant que no s'ha pogut contactar amb el servidor.
  - Per les proves del protocol s'han establert els següents valors dels temporitzadors i llindars:

$$t= 1,\ u= 2,\ n= 7,\ o= 3,\ p= 3,\ q= 3$$

El procediment de registres sense resposta del servidor es mostra en la figura 2.

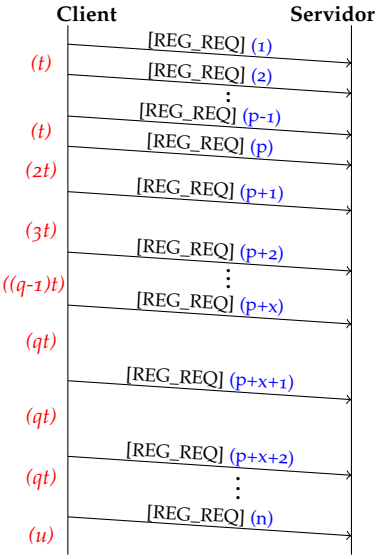

Figura 2: Temporitzacions fase registre

Els valors d'una PDU tipus [`REG_REQ`] han de ser:

- `Identificador`: Id del dispositiu (de l'arxiu configuració)
- `Número aleatori`: Zeros ("oooooooo")
- `Dades`: Buit ("")

En rebre una resposta del servidor, el client haurà d'avaluar el paquet rebut, l'estat en que es troba i actuar en conseqüència. El comportament del client segons el paquet rebut és el següent:

- [REG_ACK]: Sol es pot rebre en l'estat WAIT_ACK_REG. En rebre aquest paquet el client haurà d'emmagatzemar les dades d'identificació del servidor (Identificador, Número aleatori i adreça IP) per poder comprovar que els successius paquets del servidor contenen aquestes dades. Seguidament s'enviarà un paquet [REG_INFO], al port UDP indicat en el camp de dades del paquet [REG_ACK] rebut. El contingut del paquet [REG_INFO] serà:
  - Identificador: Id del dispositiu (de l'arxiu configuració)
  - Número aleatori: Valor enviat en el primer paquet del servidor (dades identificació servidor)
  - Dades: El port TCP del client per rebre connexions del servidor i la llista d'elements del dispositiu (de l'arxiu de configuració). El port TCP se separarà de la llista d'elements per una coma [,] (els paràmetres separats per punt i coma [;]).

  Un cop enviat el paquet [REG_INFO] el client passarà a l'estat WAIT_ACK_INFO i esperarà rebre la confirmació, paquet [INFO_ACK], durant $2t$ segons. Si passat aquest temps no s'ha rebut el paquet de confirmació el client passarà a l'estat NOT_REGISTEREDi s'iniciarà un nou procés de subscripció.
- [REG_NACK]: El client passarà a l'estat NOT_REGISTERED i iniciarà l'enviament de paquets[REG_REQ] sense iniciar un nou procés de subscripció.
- [REG_REJ]: El client passarà a l'estat NOT_REGISTERED i s'iniciarà un nou procés de subscripció.
- [INFO_ACK]: Sol es pot rebre en l'estat WAIT_ACK_INFO i les dades d'identificació del servidor han de ser correctes. El client passarà a l'estat REGISTERED, considerarà finalitzada satisfactòriament la fase de registre i passarà a la següent tasca. El camp de dades d'aquest paquet ha de contenir el port TCP pel qual el servidor espera rebre dades del client.
- [INFO_NACK]: Sol es pot rebre en l'estat WAIT_ACK_INFO i les dades d'identificació del servidor han de ser correctes. El camp de dades ha de contenir el motiu per el que no s'ha acceptat el paquet [REG_INFO]. En rebre el paquet, el client passarà a l'estat NOT_REGISTERED i iniciarà l'enviament de paquets[REG_REQ] sense iniciar un nou procés de subscripció.

  En la figura 3 es mostra el diagrama de temps d'un procés de subscripció finalitzat correctament.

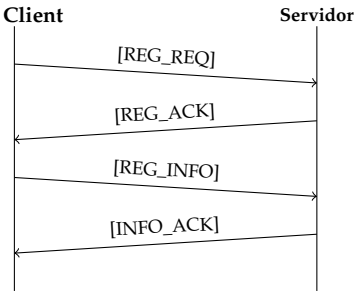

Figura 3: Procés de registre correcte

La recepció d'un paquet diferent a l'esperat en un estat especific o bé amb les dades d'identificació del servidor incorrectes comportarà el pas del client a l'estat NOT_REGISTERED i l'inici d'un nou procés de subscripció.

Mantenir comunicació periòdica amb el servidor

Per la comunicació periòdica amb el servidor s'emprarà el protocol UDP; la mateixa PDU i el mateix port del servidor que en la fase de subscripció. Per aquesta fase s'han definit 2 tipus de paquets que s'especifiquen en la taula 3.

Valor	Mnemònic	Significat
0x10	ALIVE	Enviament de ALIVE
0x11	ALIVE_REJ	Rebuig de ALIVE

Taula 3: Tipus de paquet en la comunicació periòdica amb el servidor

Procediment

Finalitzada la fase de subscripció, estat `REGISTERED`, el client enviarà al servidor un paquet [`ALIVE`] cada v segons. Per la prova del protocol s'establirà v=2. El contingut d'aquest paquet ha de ser:

- `Identificador`: Id del dispositiu (de l'arxiu configuració)
- `Número aleatori`: Valor rebut en el paquet [`REG_ACK`] pel servidor (dades identificació servidor)
- `Dades`: Buit ("")

Per constatar que hi ha comunicació amb el servidor, el client esperarà rebre com a resposta un paquet [`ALIVE`] que contingui el seu identificador en el camp de dades. El diagrama de temps d'aquest procediment es mostra en la figura 4. Per cada paquet rebut haurà de comprovar la identitat del servidor (Identificador, Número aleatori i adreça IP) i l'identificador rebut en el camp de dades. Si hi ha discrepància entre les dades d'identificació, tant del servidor com del client, es considerarà que hi ha un problema de suplantació d'identitat. En aquest cas el client passarà a l'estat `NOT_REGISTERED` i s'iniciarà un nou procés de subscripció.

En rebre el primer paquet [`ALIVE`] correcte del servidor el client passarà a l'estat `SEND_ALIVE` i es considerarà superada la fase de registre. Per la recepció d'aquest paquet s'estableix un temps màxim d'espera de r vegades l'interval d'enviament de [`ALIVE`] (per les proves del protocol r=2). Si no es rep aquest primer [`ALIVE`] en el temps estipulat el client passarà a l'estat `NOT_REGISTERED` i s'iniciarà un nou procés de subscripció.

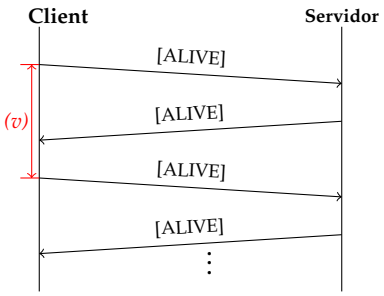

Figura 4: Manteniment de comunicació periòdic

Quan el client passa a l'estat `SEND_ALIVE` obrirà el port TCP per la recepció de connexions del servidor (*Local-TCP* de l'arxiu de configuració) i el mantindrà obert mentre es mantingui en aquest estat. En aquest estat també es podran introduir comandes per la consola del sistema.

Si el client deixar de rebre s [`ALIVE`] consecutius, considerarà que s'ha perdut la comunicació amb el servidor, passarà a l'estat `NOT_REGISTERED` i s'iniciarà un nou procés de subscripció. Per les proves del protocol s'ha establert s=3.

Els paquets [`ALIVE`] sol es poden rebre en els estats: `REGISTERED` (per el primer paquet) i `SEND_ALIVE`.

Per últim si el client rep un paquet [`ALIVE_REJ`] passarà a l'estat `NOT_REGISTERED` i s'iniciarà un nou procés de subscripció.

Enviar les dades al servidor

Tal com s'ha comentat en la introducció, la lectura i enviament de dades se simularà mitjançant comandes en la consola del sistema. Per aquestes funcions s'han definit 3 comandes:

- `stat`: Mostrar en la consola del sistema els elements del dispositiu i el seu valor actual

- `set`: Establir un nou valor a un element (simular una lectura de dades). En aquesta comanda caldrà especificar l'identificador de l'element i el nou valor separats per espais:

  `set` <*identificador_element*> <*nou_valor*>

- `send`: Enviar el valor d'un element al servidor. En aquest cas sol caldrà especificar l'identificador de l'element que es vol enviar al servidor:

  `send` <*identificador_element*>

Les dues primeres comandes son «locals» i no impliquen cap comunicació amb el servidor, la tercera ha d'establir una connexió amb el servidor per enviar les dades de l'element especificat.

Addicionalment a les tres comandes relacionades amb l'enviament de dades al servidor, en la consola del client s'ha de poder introduir la comanda `quit`, comanda que tanca tots els canals de comunicació oberts i finalitza l'execució del client. Aquesta comanda, igual que les tres anteriors, sol es podrà introduir quan el client es trobi en l'estat `SEND_ALIVE`.

L'enviament de dades s'ha de dur a terme amb el protocol TCP i en el port del servidor especificat en el segon paquet d'acceptació de dades de registre([`INFO_ACK`]). Per aquesta comunicació s'ha definit la PDU que es mostra en la figura 5.

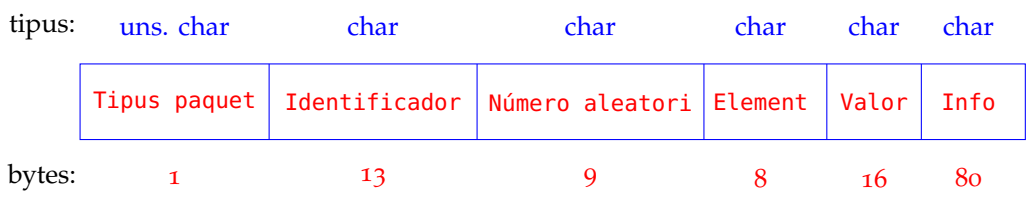

Figura 5: Format PDU TCP

Els tipus de paquet per la transferència de dades amb el servidor s'especifiquen en la taula 4.

Valor	Mnemònic	Significat
0x20	SEND_DATA	Enviament de dades des del dispositiu
0x21	SET_DATA	Enviament de dades des del servidor
0x22	GET_DATA	Petició de dades des del servidor
0x23	DATA_ACK	Acceptació d'un paquet de dades
0x24	DATA_NACK	Error en un paquet de dades
0x25	DATA_REJ	Rebuig d'un paquet de dades

Taula 4: Tipus de paquet transferència de dades amb el servidor

Procediment

La comanda `send` <*identificador_element*> endegarà el procés d'enviament de dades cap al servidor. El client establirà una comunicació TCP amb el servidor (port especificat en el paquet [`INFO_ACK`]) i, un cop establerta la comunicació, enviarà un paquet [`SEND_DATA`] amb els següents valors en la PDU:

- `Identificador`: Id del dispositiu (de l'arxiu configuració)

- `Número aleatori`: Valor rebut en el paquet [REG_ACK] per el servidor (dades identificació servidor)
- `Element`: Identificador de l'element especificat en la comanda `send`
- `Valor`: Valor associat a l'element
- `Info`: Data i hora de la mesura amb el format: *yyyy-mm-dd;hh:mm:ss*

Seguidament esperarà resposta del servidor durant m segons (m=3 per les proves del protocol). El diagrama de temps de l'enviament de dades al servidor és el mostrat en la figura 6.

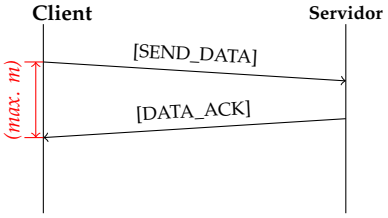

Figura 6: Enviament de dades al servidor

Del servidor es poden rebre les següents respostes (tipus paquet):

- [DATA_ACK]: Enviament d'informació acceptada, s'han emmagatzemat les dades en el servidor. El camp `Info` d'aquest paquet ha de contenir l'identificador del dispositiu,
- [DATA_NACK]: El servidor no ha pogut emmagatzemar les dades enviades o bé s'han rebut les dades errònies. Les dades no han estat acceptades.
- [DATA_REJ]: La informació enviada al servidor ha estat rebutjada.

Si la resposta del servidor és [DATA_REJ] o bé hi ha discrepàncies amb les dades d'identificació, tant del servidor com del dispositiu, el client passarà a l'estat `NOT_SUBSCRIBED` i s'iniciarà un nou procés de subscripció.

La recepció d'un paquet de resposta [DATA_NACK], la manca de resposta del servidor o la discrepància entre les dades de l'element del dispositiu enviades i rebudes (Nom o valor), es considerarà que les dades no han estat acceptades. En aquest cas caldria reenviar-les fins rebre confirmació correcta de la recepció. En la pràctica no caldrà implementar aquest reenviament, únicament caldrà informar per consola d'aquesta situació.

Després de rebre una resposta o bé després de la temporització per manca de resposta, el client tancarà la comunicació TCP amb el servidor.

Esperar connexions TCP del servidor

Un cop finalitzada amb èxit la fase de fase de registre i rebuda la confirmació al primer paquet [ALIVE] (client en estat `SEND_ALIVE`), el client obrirà el port TCP especificat en l'arxiu de configuració amb el paràmetre *Local-TCP* per rebre informació o peticions d'informació des del servidor. Per aquestes peticions s'empra la mateixa PDU (figura 5) i tipus de paquets (taula 4) que en l'enviament de dades des del client cap al servidor.

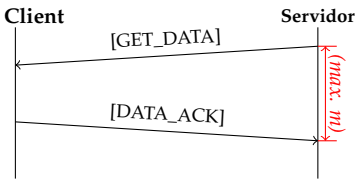

Figura 7: Peticio de control del servidor [GET]

Del servidor es poden rebre dos tipus de paquets TCP:

- [SET_DATA]: Establir el valor d'un element d'entrada (actuador). Caldrà comprovar que l'element pertany al dispositiu i que es tracta d'un dispositiu d'entrada. Si tot és correcte s'assignarà el valor rebut a l'element (variable associada) i el client respondrà amb un paquet [DATA_ACK] que contindrà les seves dades d'identificació (Id dispositiu i Número aleatori) i els valors de l'element rebuts (Nom i Valor). El camp `Info` contindrà l'identificador del dispositiu.
- [GET_DATA]: Obtenir el valor d'un element. S'haurà de comprovar que l'element pertany al dispositiu i en cas afirmatiu es respondrà amb un paquet [DATA_ACK]

que contindrà les seves dades d'identificació (Id dispositiu i Número aleatori) el nom de l'element i el valor de l'element (variable associada). El camp `Info`, igual que en el cas anterior, contindrà l'identificador del dispositiu.

Si en el paquet rebut del servidor hi ha discrepàncies en les dades d'identificació del servidor o el dispositiu, s'enviarà un paquet [`DATA_REJ`] amb les dades d'identificació del client (Id dispositiu i Número aleatori), les dades rebudes de l'element (Nom i Valor) i en el camp `Info` el motiu del rebuig. El client passarà a l'estat `NOT_SUBSCRIBED` i s'iniciarà un nou procés de subscripció.

Cas d'haver discrepàncies amb la identificació de l'element (tipus o nom), s'enviarà un paquet [`DATA_NACK`] amb les dades d'identificació del client (Id dispositiu i Número aleatori), les dades rebudes de l'element (Nom i Valor) i en el camp `Info` el motiu del error.

Després d'haver enviat el paquet de resposta al servidor, o si no s'ha rebut cap paquet del servidor en el temps estipulat (el mateix que en el cas d'enviament de dades al servidor: *m*), es finalitzarà la comunicació amb el servidor. Les figures 7 i 8 mostren els diagrames de temps per la recepció de comandes correctes del servidor.

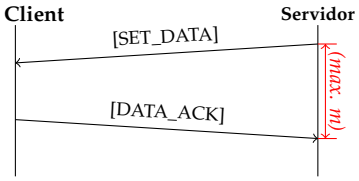

Figura 8: Petició de control del servidor [SET]

Servidor

Per la seva part, el servidor genèricament ha d'efectuar quatre tasques:

- Atendre les peticions de registre dels clients.
- Gestionar el manteniment de comunicació amb els equips registrats en el sistema.
- Esperar connexions TCP dels clients i tractar la informació que enviïn.
- Enviar informació i peticions d'informació als clients mitjançant TCP.

El servidor haurà d'atendre múltiples clients a l'hora (servidor concurrent).

Igual que el client, el servidor ha de llegir d'un arxiu el seus paràmetres de configuració. Aquest arxiu ha de tenir la mateixa estructura que l'esmentada per el client i per defecte s'anomenarà: `server.cfg`. Igual que en el client aquest nom es podrà variar emprant l'opció *-c* (`-c`<*nom_arxiu*>).

Els paràmetres que ha de contenir aquest arxiu seran:

- *Id*: Identificador del servidor, codi alfanumèric de 12 dígits.
- *UDP-port*: Port UDP per el que atendrà les peticions de registre i el manteniment de la comunicació.
- *TCP-port*: Port TCP per que el atendrà l'enviament de dades dels clients.

El servidor ha de tenir emmagatzemats en un arxiu els identificadors dels dispositius (clients) que es poden connectar al sistema. Aquest arxiu es llegirà en iniciar-se el servidor i es mantindrà en memòria amb estructura de dades que li permeti controlar l'estat dels clients, els codis aleatoris, adreça IP, etc. El nom de l'arxiu és `bbdd_dev.dat` i contindrà un identificador de dispositiu en cada línia.

En iniciar-se el servidor haurà d'obrir el dos canals de comunicació, UDP i TCP, i esperar les comunicacions dels clients. Per cada client, i cada protocol, haurà de destinar un procés que l'atengui. Un cop oberts els canals de comunicació restarà a l'espera de comandes per la consola del sistema.

Es descriuen a continuació cadascuna de les quatre tasques que ha d'efectuar el servidor:

Atendre peticions de registre

En cada petició de registre el servidor haurà de comprovar que qui l'efectua és un dispositiu autoritzat en el sistema i que li envia les dades correctes (Número aleatori tot a zeros i camp de dades buit). Si hi ha discrepància amb aquestes dades, el servidor respondrà amb un paquet [REG_REJ] i el client es passarà a l'estat DISCONNECTED finalitzant el procés de subscripció. El contingut d'aquest paquet ha de ser:

- Identificador: Identificador del servidor (de l'arxiu de configuració)
- Número aleatori: Tot el contingut a valor zero (vuit zeros)
- Dades: Motiu del rebuig

Si totes les dades d'identificació del client són correctes i aquest es troba en l'estat DISCONNECTED, el servidor generarà un nombre aleatori per la comunicació, obrirà un nou port UDP per continuar el procés de subscripció amb el client i li enviarà un paquet [REG_ACK] amb el contingut:

- Identificador: Identificador del servidor (de l'arxiu de configuració)
- Número aleatori: Valor del nombre aleatori generat
- Dades: Nou port UDP obert per continuar amb la subscripció

Seguidament el client es passarà a l'estat WAIT_INFO i esperarà rebre un paquet [REG_INFO], per el nou port UDP obert, durant s vegades el temps inicial d'enviaments de peticions de subscripció (per les proves del protocol s=2). Si no es rep el paquet en el temps esperat es passarà el client a l'estat DISCONNECTED i finalitzarà el procés de subscripció en el servidor.

La recepció d'un paquet [REG_INFO] correcte, tant en les dades d'identificació del client com en les dades addicionals, comportarà l'enviament d'un paquet [INFO_ACK], el canvi de l'estat del client a REGISTERED, l'emmagatzemament de les dades rebudes i la finalització del procés de subscripció. El contingut del paquet [INFO_ACK] ha de ser:

- Identificador: Identificador del servidor (de l'arxiu de configuració)
- Número aleatori: Valor del nombre aleatori generat
- Dades: Port TCP del servidor per l'enviament d'informació dels dispositius

Si el paquet [REG_INFO] és incorrecte s'enviarà un paquet [INFO_NACK], el client es canviarà a l'estat DISCONNECTED i finalitzarà el procés de registre en el servidor. Els valors dels camps del paquet [INFO_NACK] seran:

- Identificador: Identificador del servidor (de l'arxiu de configuració)
- Número aleatori: Valor del nombre aleatori generat
- Dades: Motiu per el que no s'ha acceptat el paquet [REG_INFO]

La recepció d'un paquet no adequat a l'estat en el que es troba un client provocarà el canvi d'estat del client a DISCONNECTED i la finalització, en el servidor, del procés de registre.

Gestionar el manteniment de comunicació periòdica

La gestió de la comunicació periòdica amb els clients té dues parts:

1. Respondre als paquets [ALIVE]: El servidor respondrà a cada paquet [ALIVE] correcte (dades d'identificació i estat del client) amb un altre paquet [ALIVE] amb el contingut:

   - `Identificador`: Identificador del servidor (de l'arxiu de configuració)
   - `Número aleatori`: Valor del nombre aleatori generat
   - `Dades`: Identificador del dispositiu al que s'envia el paquet

   Amb aquesta operació el client podrà constatar que el servidor es troba operatiu. En rebre el primer paquet [ALIVE] d'un client que està en l'estat REGISTERED es passarà el client a l'estat SEND_ALIVE. Aquest primer paquet s'ha de rebre abans de w segons (per les proves del protocol s'ha establert w=3) des de que el client ha passat a l'estat REGISTERED. De no ser així el client es passarà a l'estat DISCONNECTED. Si el paquet [ALIVE] rebut és incorrecte s'enviarà un paquet [ALIVE_REJ] i es passarà el client a l'estat DISCONNECTED.

2. Comprovar que els clients registrats estan operatius: S'haurà de comprovar que tots els clients que han superat la fase de registre mantinguin la comunicació periòdica. En aquest sentit, si es deixen de rebre x paquets [ALIVE] consecutius d'un client (x=3 per les proves del protocol), el client es passarà a l'estat DISCONNECTED.

Esperar connexions TCP dels clients

Per cada connexió TCP que rebi el servidor haurà de crear un procés per atendre-la (servidor concurrent). En aquesta connexió sol pot rebre un tipus de paquet: [SEND_DATA]. El servidor haurà de comprovar que es tracta d'un dispositiu autoritzat en el sistema, que les dades d'identificació són correctes, que es troba en l'estat SEND_ALIVE i que l'element referenciat en el paquet pertany al dispositiu. Si tot és correcte, el servidor emmagatzemarà les dades a disc i respondrà al client amb un paquet [DATA_ACK] amb les seves dades d'identificació, les dades de l'element i valor rebudes del client i l'identificador del dispositiu en el camp `Info`. Un cop enviat el paquet [DATA_ACK] es finalitzarà la comunicació amb el client.

Les dades rebudes del client s'emmagatzemaran en el directori de treball del servidor en un arxiu històric de dades. El nom d'aquest arxiu serà l'identificador del dispositiu amb l'extensió `.data`. Per cada dada rebuda es generarà una línia de dades en l'arxiu i cada línia ha de contenir:

- La data de recepció (DDDD-MM-YY)
- L'hora de recepció (HH:MM:SS)
- El tipus de paquet que ha generat la dada
- El nom de l'element
- El valor de l'element

En cada línia els components se separaran per punt i coma (`;`)

Si el servidor no pot emmagatzemar les dades rebudes del client s'enviarà un paquet [DATA_NACK] amb les dades d'identificació del servidor, l'element i el valor rebut i en el

camp `info` el motiu de la fallida d'emmagatzemament. Aquest mateix paquet s'enviarà si hi ha errors en les dades rebudes (element i/o valor), indicant en el camp `info` el motiu de l'error.

Cas que el paquet [SEND_DATA] sigui erroni en la identificació del dispositiu, s'enviarà un paquet [DATA_REJ] amb les seves dades d'identificació i les dades de l'element i valor rebudes del client, el client es passarà a `DISCONNECTED` i es finalitzarà la comunicació amb el client.

Per les comunicacions TCP en el servidor s'ha establert la mateixa temporització per la recepció de paquets que en el client: *m*.

Enviar informació i peticions d'informació als clients

Des del servidor es podrà enviar informació als elements dels dispositius que ho permetin (actuadors) i també es podrà sol·licitar informació als elements dels dispositius (actuadors i sensors). Aquestes operacions es duran a terme mitjançant una connexió TCP al port especificat per el client en la fase de subscripció. La PDU per aquesta comunicació és la descrita en la figura 5 i els tipus de paquets els especificats en la taula 4.

Per l'enviament d'informació s'emprarà el paquet [SET_DATA] i per la sol·licitud d'informació el paquet [GET_DATA]. Ambdós paquets s'hauran d'omplir amb l'identificador del servidor, el nombre aleatori assignat al client amb el que es vol comunicar, el nom de l'element i el valor assignat en el cas de [SET_DATA] o bé valor buit ("") en el cas de [GET_DATA]. Tots dos paquets hauran de contenir en el camp `Info` l'identificador del dispositiu amb el que es vol comunicar.

Un cop enviat el paquet corresponent el servidor esperarà resposta del client durant *m* segons. L'acció del servidor dependrà del tipus de paquet rebut:

- [DATA_ACK]: El paquet contindrà les dades sol·licitades (o les dades enviades) i caldrà emmagatzemar-les tal com s'ha especificat en l'apartat anterior. Un cop emmagatzemades es tancarà la comunicació amb el client.
- [DATA_NACK]: Es finalitzarà la comunicació amb el client considerant l'operació fallida.
- [DATA_REJ]: El client passarà a l'estat `DISCONNECTED` i es finalitzarà la comunicació amb el client. L'operació també es considerarà fallida.
- Sense resposta: Es finalitzarà la comunicació amb el client considerant l'operació fallida.

En el supòsit que el servidor no pugui contactar amb el client per el port TCP es passarà el client a l'estat `DISCONNECTED` i es finalitzarà la comunicació.

Igual que en el client, la manca de resposta o bé la recepció d'un paquet [DATA_ACK], implicaria el reenviament de la informació en no haver-se efectuat correctament l'operació. En aquest cas tampoc caldrà implementar-ho en la pràctica i serà suficient informar per la consola de la fallida.

Les operacions d'enviament i petició d'informació sol es podran efectuar als dispositius en l'estat `SEND_ALIVE` i es duran a terme mitjançant les següents comandes en la consola del sistema:

- `set`: Establir un valor a un element d'un dispositiu (ha de tractar-se d'un element d'entrada, actuador). En aquesta comanda caldrà especificar l'identificador del dispositiu, l'identificador de l'element i el valor a establir separats per espais:

`set` <*identificador_dispositiu*> <*identificador_element*> <*nou_valor*>

- `get`: Obtenir el valor d'un element d'un dispositiu (en aquest cas tant pot ser un element d'entrada com de sortida). Per aquesta comanda sol caldrà especificar l'identificador del dispositiu i l'identificador de l'element del que es vol obtenir el valor:

  `send` <*identificador_dispositiu*> <*identificador_element*>

Juntament a les dues comandes de comunicació amb el client, en la consola del sistema del servidor s'han de permetre dues comandes més:

- `list`: Mostra en la consola del sistema una taula amb els dispositius autoritzats en el sistema. Per cada dispositiu s'han de mostrar les següents dades:
  - Identificador
  - Número aleatori
  - Adreça IP
  - Elements del dispositiu
- `quit`: Tanca tots els canals de comunicació i finalitza l'execució del servidor.

Totes les comandes estaran disponibles en la consola del sistema un cop el servidor estigui operatiu.

Implementació

La implementació de la pràctica s'haurà de fer en dos llenguatges:

- **Client**: Python (3.x [superior a 3.6])
  - Sol es poden emprar llibreries estàndard
  - De les llibreries de comunicacions s'han d'emprar `socket` i `select`
  - S'ha d'emprar *shebang* (`#!`) amb la versió de Python
- **Servidor**: Ansi C (C99).
  - Sol es poden emprar llibreries estàndard
  - S'emprarà l'eina `make` per compilar
  - S'ha de compilar amb `gcc` i les opcions `-ansi -pedantic -Wall`

Tant el client com el servidor han de permetre els següents paràmetres d'entrada:

-d : Fer *debug*. S'ha de presentar per consola un missatge per cada esdeveniment significatiu. Aquests missatges han de permetre fer un seguiment del funcionament de la implementació i del protocol.

-c <*arxiu*>: Arxiu de dades de configuració del software. Especifica el nom de l'arxiu d'on es llegiran les dades necessàries per la comunicació entre client i servidor. Si no s'especifica aquest paràmetre el nom per defecte serà especificat en la documentació.

El servidor ha de permetre un paràmetre addicional:

-u <*arxiu*>: Arxiu de dispositius autoritzats. Especifica el nom de l'arxiu que conté els identificadors dels dispositius autoritzats en el sistema. Per defecte el seu valor ha de ser: `bbdd_dev.dat`.

Independentment de l'opció de *debug* esmentada anteriorment, tant el client com el servidor han de mostrar per consola un missatge amb els canvis d'estat dels clients.

Per facilitar el desenvolupament de la pràctica, es proposa dividir-la en dues parts i cada part dividir-la en cinc fases progressives:

1. Implementació del client
   (a) Registre
   (b) Manteniment de la comunicació
   (c) Introducció de comandes bàsiques
   (d) Enviament de dades al servidor
   (e) Recepció de dades del servidor
2. Implementació del servidor
   (a) Registre
   (b) Manteniment de la comunicació
   (c) Introducció de comandes bàsiques
   (d) Recepció de dades dels clients
   (e) Enviament de dades als clients

Per poder implementar el client sense la dependència d'haver de construir un servidor, es proporcionarà un servidor bàsic per avaluar el desenvolupament del client. Igualment es proporcionarà un client bàsic per avaluar el desenvolupament del servidor. Tant el client com el servidor que es proporcionin permetran efectuar tests bàsics de funcionament de cadascuna de les parts.

Lliurament

Documentació

La pràctica és individual, cada alumne haurà de lliurar un únic arxiu amb les següents característiques:

- *Nom*: **XARXES-P1-***nom* (On *nom* són els dos cognoms de l'autor separats per guionet (-))
- *Format*: ZIP, TGZ o TBZ.
- *Contingut*:
  - Codi font degudament comentat pel seu seguiment.
  - Arxius necessaris pel correcte funcionament del codi (configuracions clients, servidor, etc).
  - Un informe en format PDF que inclogui:
    1. L'estructura, a nivell esquemàtic (diagrama de blocs), del client i del servidor.
    2. L'estratègia emprada per el manteniment de la comunicació, tant en el client com en el servidor.
    3. Un diagrama d'estats del protocol implementat sobre UDP (registre i manteniment de la comunicació).
    4. Totes aquelles consideracions que es considerin oportunes.

    Aquest document ha de complir les següents condicions:

    * Una bona estructuració.
    * Un format dels elements de text correctes.
    * Un contingut clar i una redacció correcta (no s'acceptaran errades ortogràfiques ni abreviatures o símbols per substituir paraules).
- S'ha de lliurar al campus virtual (apartat Activitats) i no s'acceptarà per cap altre mitjà.

> **IMPORTANT**
>
> **La documentació que no compleixi tots aquests requisits no serà avaluada.**

Termini

El termini per rebre la documentació relacionada amb aquesta pràctica finalitza el dia **20 d'abril de 2020**.

Avaluació

Per a que una pràctica pugui ser avaluada haurà de complir els següents requisits:

- El codi del client no ha de donar cap tipus d'error en el procés de compilació ni en la execució.
- El codi del servidor s'ha d'executar sense errors.
- S'executarà el servidor i dos clients en la mateixa màquina i, com a mínim, han d'arribar a la fase de manteniment de la comunicació (els clients enviant periòdicament ALIVE i el servidor responent a cadascun d'ells). El servidor ha de poder llistar els equips autoritzats en els sistema (comanda `list`) i el client visualitzar l'estat dels elements (comanda `stat`). Tant en el servidor com en el client s'ha de poder emparar (i funcionar) la comanda `quit`.
- Complir tots els requeriments de l'apartat *Documentació* de la secció **Lliurament**

La avaluació de la pràctica es realitzarà en un sistema Linux (Fedora fc31.x86_64).