Enginyeria de Sistemes TIC

Xarxes de Comunicació

Pràctica 3 - Protocol d'enllaç / Transferència fiable

Francisco del Águila López

Octubre 2021

Escola Politècnica Superior d'Enginyeria de Manresa
Universitat Politècnica de Catalunya

1 Objectiu

L'objectiu d'aquesta pràctica és definir un mòdul en C que implementi un protocol fiable per sobre del nivell d'accés al medi. Es considera un intercanvi de dades entre dos dispositius en una xarxa d'àrea local basada en comunicació infraroja amb morse.

2 Punt de partida

El punt de partida serà la llibreria libpbn.a juntament amb el mòdul (lan.h, lan.c) implementat a la pràctica anterior i el mòdul Error_morse.

Cal recordar que la llibreria libpbn.a disposa dels mòduls:

- Block_Ether: És qui implementa la capa física. Aquest mòdul es farà servir essencialment pel mòdul Lan. En aquesta pràctica, la interacció amb el mòdul Ether serà nula, ja que ho farà el mòdul Lan, respectant estrictament una estructura per capes.
- Serial: Permet enviar i rebre dades pel port sèrie implementant un driver basat en interrupcions i cues.
- Block_serial: Permet enviar i rebre blocs de bytes pel port sèrie. Utilitza el mòdul Serial.

- Timer: Aquest mòdul permet realitzar accions temporitzades gràcies a la instauració de funcions de *callback* temporitzades.
- Error_Morse: Permet la detecció d'errors associats a missatges formats per blocs de bytes.

3 Protocol fiable

Per simplificar la pràctica, s'implementarà el protocol de parada i espera. D'aquesta manera, no es podrà transmetre el nou missatge fins que no s'hagi reconegut l'anterior. La implementació del protocol s'ha de basar en els autòmats vists a les classes de teoria.

Una consideració a tenir en compte per implementar un protocol fiable és que es fa necessari el concepte d'establiment de connexió. Amb l'establiment de connexió s'aconsegueix que tant el transmissor com el receptor sincronitzin els identificadors de les trames, és a dir, el número de trama que s'enviarà amb el número de trama que s'espera rebre. Per altra banda, quan un dispositiu està intercanviant informació amb un altra, l'establiment de connexió fa que només s'estigui "atenent" les trames del dispositiu amb qui hi ha comunicació, fent que les trames rebudes d'altres dispositius siguin ignorades. Per implementar aquest establiment de connexió, el mòdul Frame (també anomenada capa Frame) oferirà a la seva API les funcions frame_connect() i frame_disconnect().

3.1 Callback de final de transmissió

El mòdul Ether disposa a la seva API d'un callback que es pot instal·lar per indicar el final de transmissió quan s'ha cridat a la funció *ether_block_put()*. Això pot indicar quan el canal estarà ocupat degut a la pròpia transmissió i al mateix moment (en l'execució del callback) indica quan queda el canal alliberat ja que la transmissió ha finalitzat.

Aquest callback s'ha traslladat a la capa Lan i per tant queda disponible també a la capa Frame.

L'indicador de final de transmissió pot ser molt útil per activar temporitzadors a partir del moment en que segur que el canal no està ocupat per la nostra pròpia transmissió. La gestió d'aquest esdeveniment per part de l'autòmat de transmissió és molt útil per disposar d'un estat en el que s'indica que la transmissió d'un missatge ja s'ha iniciat, però encara no s'ha finalitzat. Un cop rebut aquest esdeveniment (final de transmissió) llavors es poden activar els temporitzadors necessaris per a la gestió de la comunicació fiable. Sense aquest estat de transició, la gestió dels temporitzadors associats a l'enviament de missatges es fa molt dispersa ja que per una banda els missatges poden tenir durades molt diferents i per altra, la capa Lan (accés al medi) també pot afegir més latència a la transmissió.

3.2 Unitat de Dades de Protocol

La unitat de dades de protocol (PDU) en aquest cas rep el nom de trama. Un dels aspectes més rellevants en la especificació d'un protocol és la definició de com han de ser aquestes trames. En el cas de la pràctica, es pot classificar els tipus de trames en:

1. *Trames de dades:* Són les que transporten la informació corresponent als missatges que es volen transmetre. Tenen un camp específic on viatgen aquestes dades.
2. *Trames de control:* No contenen cap tipus d'informació. Només es fan servir per gestionar el correcte funcionament del protocol. En el cas de la pràctica són les trames encarregades de realitzar les confirmacions de la rebuda de les dades. També en un futur poden encarregar-se de realitzar altres funcions com l'establiment de la connexió o tancament de la connexió.

3.2.1 Trames de dades

Estan formades pels següents camps:

Tipus És el camp que determina que aquesta trama és una trama de informació. Aquest camp és de la mida de 1 byte. Per tal de poder distingir-la de la resta de trames, el valor d'aquest byte serà "I".

Numeració de trama És el camp corresponent a identificar les trames. En el cas del protocol parada / espera, només caldria un únic bit per identificar les trames, però el canal físic de comunicació està basat en caràcters morse, per tant es reservarà un caràcter (Byte) per identificar les trames. Els caràcters que es faran servir per identificar les trames són "0" i "1".

Camp de dades És un camp de mida variable múltiple de Byte, que contindrà les dades que s'han de transportar. En general, contindrà la unitat de dades de protocol de la capa superior, però en el cas de la pràctica, la capa superior directament serà la capa d'aplicació, per tant contindrà els missatges que es volen transmetre.

3.2.2 Trames de control de reconeixement

Són trames de mida molt petita. Es fan servir per indicar la confirmació o reconeixement de les trames de dades enviades. Per simplificar i aprofitar la numeració de trames només es defineixen trames de confirmació. Per indicar una NO confirmació es farà servir el número de trama que no correspon.

En aquest cas, els camps són:

Tipus És un camp de 1 byte. Serveix per identificar el tipus de trama que tenim. Quedarà indicat amb el valor "R".

Numeració de trama Té mida de 1 caràcter (byte). Es recorda que en cas de voler enviar una NO confirmació s'enviarà una confirmació de la trama amb la identificació que no correspon (Si la trama de dades rebuda té identificador "0" i es vol enviar una NO confirmació, s'enviarà una trama de control de reconeixement amb identificador "1").

Com a conclusió, aquestes trames tindran mida fixe de 2 caràcters sempre.

3.2.3 Definició de la unitat de dades de protocol en C

Per tal de treballar amb aquest tipus de trames, es definirà el següent tipus de dades:

```
typedef struct {
    morse_c_t tipus;
    morse_c_t num;
    morse_c_t payload[MAX_MSS_FRAME];
} frame_i_pdu_t;
```

```
typedef struct {
    morse_c_t tipus;
    morse_c_t num;
    morse_c_t payload[1];
} frame_r_pdu_t;
```

Considereu aplicar l'atribut "volatile" allà on creieu necessari.

Justifiqueu el perquè frame_r_pdu_t té un payload de mida 1.

3.3 Implementació del protocol (primera aproximació)

Per tal de fer la implementació del protocol de manera progressiva, considerarem en primer cas que entre els dos nodes que es comuniquen, un d'ells serà només transmissor i l'altre serà només receptor. El node transmissor s'encarregarà només d'enviar trames de dades cap al receptor. Per tant, les úniques trames que pot rebre seran només les de reconeixement. Per altre banda, el node receptor només podrà rebre trames de dades i les úniques trames que pot transmetre seran les de reconeixement. Aquest esquema inicial facilita la implementació ja que en el node transmissor només s'ha d'implementar l'autòmat de transmissió i en el node de recepció només l'autòmat de recepció. El sistema final ha d'acabar fusionant els dos autòmats al mateix mòdul.

En aquest primer moment la transferència d'informació va en un sol sentit. Això no vol dir que un dispositiu només sigui transmissor i l'altre receptor, sinó que els dos dispositius seran transmissors i receptors però que els missatges a nivell d'aplicació (les dades que es volen transmetre) només aniran en un sol sentit.

Degut a que la capa Frame quedarà seccionada en transmissor i receptor, la seva implementació en forma de mòdul també. Per tant, crearem el mòdul frame_tx encarregat d'implementar l'autòmat de transmissió i el mòdul frame_rx encarregat d'implementar l'autòmat de recepció.

3.3.1 Dispositiu transmissor

Aquest dispositiu haurà d'implementar la màquina d'estats transmissora vista a classe en el mòdul frame_tx.

En aquest protocol es fan successives retransmissions cada vegada que hi ha un problema amb la trama de dades enviada. El número de transmissions successives serà il·limitat per facilitar així la implementació de l'autòmat. En aquest cas, qui s'encarregarà de determinar que la comunicació és inviable serà el mateix usuari. Per altra banda, les úniques trames que rebrà el transmissor seran trames de control per reconèixer les trames de dades prèviament enviades.

El valor de *timeout* del temporitzador el fixarem de manera que no es produeixin *timeouts* abans de temps.

capa Aplicació

Des del punt de vista de l'aplicació final (capa superior), la interacció amb l'usuari a través del port sèrie, facilitarà només l'enviament de la informació, ja que no haurà cap recepció de trames d'informació. També facilitarà l'establiment de connexió, establint amb quin node es manté la comunicació. Tanmateix, si s'habilita un mode de debug, es pot aprofitar el port sèrie pels missatges de debug.

Això significa que l'aplicació associada al transmissor només farà servir les funcions *frame_can_put()*, *frame_block_put()* i *frame_connect()* de la API. El nom que se li donarà a aquesta aplicació transmissora serà *app_tx*.

Des de l'aplicació l'usuari ha de poder escollir entre enviar trames d'informació (amb qui té la connexió establerta) o bé establir una nova connexió.

Establiment_de_connexió: Això s'implementarà de manera que per establir una nova connexió, s'enviarà el patró de text: "$>$ND[enter]". És a dir, el caràcter '$>$' determina amb qui es vol establir una nova connexió. A continuació s'escriu l'adreça de destí del node amb que s'estableix la connexió i per últim la tecla [enter] acaba la comanda.

Enviament_de_informació: En aquest cas, s'escriurà directament el text que es vol enviar seguit de la tecla [enter] per indicar el final.

3.3.2 Dispositiu receptor

Aquest dispositiu haurà d'implementar en el mòdul frame_rx la màquina d'estats receptora vista a classe.

Aquest mòdul, a part de lliurar el missatge correctament rebut a la capa d'aplicació, haurà d'enviar el corresponent reconeixement al transmissor. Aquestes trames de reconeixement són unes trames molt curtes amb un contingut molt concret.

En aquesta primera versió el mecanisme d'establiment de connexió es vol minimitzar al màxim. Per tant, des del punt de vista del receptor, quedarà determinat amb qui s'estableix la connexió ja que la capa d'aplicació li especificarà. Això vol dir, que cada vegada que es vulgui establir una nova connexió amb un altre dispositiu, l'usuari des de l'aplicació especificarà amb qui quedarà establerta la connexió com a receptor.

capa Aplicació

Des del punt de vista de la capa d'aplicació, només podrà rebre missatges. Per tant, simplement haurà de fer servir el callback de la recepció de missatges i la funció de recepció *frame_block_get()*. Per tant, l'aplicació receptora no farà res que no sigui anar mostrant els missatges rebuts. Igualment, si s'habilita un mode de debug, el port sèrie anirà mostrant aquests missatges.

Per simplificar la tasca d'establiment de connexió, l'usuari especificarà amb qui s'estableix la connexió, i per tant, de qui s'espera rebre missatges de la següent manera:

Establiment_de_connexió: S'enviarà el patró de text: "<ND[enter]". És a dir, el caràcter '<' determina de qui s'espera rebre els missatges d'informació (establir la connexió). Per tant, serà el destinatari (ND) de les trames de reconeixement. La tecla [enter] acaba la comanda.

El nom que si li donarà a l'aplicació receptora serà *app_rx*.

3.4 Implementació del protocol (segona aproximació)

3.4.1 Dispositiu transceptor

Quan aquesta comunicació unidireccional funcioni correctament, l'objectiu final de la pràctica consisteix en fusionar els dos autòmats a dins del mateix mòdul. Això seria extraordinàriament simple si els dos autòmats fossin totalment independents, ja que implicaria cridar un o l'altre en funció de l'esdeveniment que passi. El problema apareix quan hi ha esdeveniments que poden estar associats als dos autòmats, com seria la rebuda d'un missatge. En aquest cas no se sap a priori si és un missatge de reconeixement (processat per l'autòmat de transmissió) o bé un missatge de dades (processat per l'autòmat de recepció). En aquest cas, després de reconèixer quin tipus de trama és, som capaços de llençar esdeveniments que només estiguin associats a un sol autòmat, desapareixent així el problema. Per tant, quan es rep un missatge genèric s'ha de pre-processar per distingir si és un missatge de dades o de reconeixement. Un cop feta la distinció, s'ha de llençar l'esdeveniment "rebuda missatge dades" (processat per recepció) o bé l'esdeveniment "rebuda missatge reconeixement" (processar per transmissió).

Anomena a aquesta aplicació *app_trans* i al mòdul Frame fusionat *frame_fus*.

4 Mòdul Frame

La versió final del mòdul Frame incorporarà simultàniament la transmissió i la recepció en cada dispositiu, a més d'una gestió de l'establiment de la connexió. A continuació es mostra com hauria de ser la API d'aquest mòdul.

4.1 API del mòdul

El mòdul Frame és on s'ha d'implementar el protocol fiable que s'ha definit en aquesta pràctica. Aquest mòdul farà servir els mòduls de les anteriors pràctiques. Aquest mòdul ha d'oferir funcions (servei) per permetre una comunicació fiable, per tant el fitxer de capçalera podria ser el següent

```
#ifndef FRAME_H
#define FRAME_H
#include <stdint.h>
#include <stdbool.h>
#include "lan.h"

#define MAX_MSS_FRAME (MAX_MSS_LAN - 2)
typedef morse_c_t message_frame_t[MAX_MSS_FRAME];

void frame_init(morse_c_t no);

void frame_connect(morse_c_t nd);
void frame_disconnect(void);

bool frame_can_put(void);
void frame_block_put(const missatge_frame_t m);

typedef void (*fc_t)(void); //Callback
void frame_block_get(message_frame_t m);
void on_frame_received(fc_t f);

#endif
```

Aquest mòdul ofereix funcions equivalents a les que ofereixen els mòduls Ether o Lan. Però en aquest cas ens assegurem que el servei que ofereix és fiable.

La funció *frame_init()* inicialitza el mòdul i per tant el protocol. Cal destacar que amb aquesta inicialització queda fixada la identificació del propi node.

La funció *frame_connect()* estableix la connexió amb un altre dispositiu de la xarxa.

Quan s'estableix, sincronitza la numeració de les trames tant en transmissió com en recepció. Per tant, deixarà els autòmats en l'estat inicial.

La funció *frame_disconnect()* allibera la connexió i per tant deshabilita els autòmats tant de transmissió com de recepció.

La funció *frame_can_put()* determina si es possible enviar un nou missatge. Aquesta funció donarà resultat false quan hi hagi algun missatge pendent de ser reconegut.

La funció *frame_block_put()* permet enviar un missatge, sempre que la funció frame_can_put() sigui "true". En aquest cas, un cop establerta la connexió, s'ha de tenir en compte que la comunicació ha sigut inicialitzada per mantenir una conversa entre 2 nodes. Per tant, no és necessari indicar qui és el destinatari del missatge.

La funció *frame_block_get()* és la que s'encarrega de fer la recepció del missatge rebut, deixant el seu contingut a la variable del tipus *message_frame_t* que es passa com a paràmetre. Actua de la mateixa manera que ho fan les funcions equivalents als mòduls Ether i Lan.

El lliurament de les dades rebudes es fa via Callback gràcies a la funció *on_frame_received()*.

Tant la part transmissora com la part receptora necessiten una única trama cadascuna sense la necessitat de tenir cap tipus de buffer. En un protocol de parada / espera no es pot transmetre cap altre trama fins que aquesta no sigui reconeguda totalment. Quan es rep una trama, aquesta es lliurada directament a la capa superior, quedant així de nou la variable disponible per rebre la següent trama.

5 Capa d'aplicació

L'aplicació serà la més simple possible i consisteix en la transmissió de missatges de text entre els dos dispositius. Es a dir, s'implementarà un xat. Els dos dispositius Arduino estaran connectats pel canal Morse entre ells i és on es fa servir el protocol d'enllaç dissenyat. Cadascun dels Arduino estarà connectat a un PC a través del canal sèrie. El PC farà la funció simplement de terminal amb una aplicació de terminal com pot ser el *picocom*.

Cada bloc de dades rebut (cada missatge) es presentarà per pantalla del terminal com una línia nova. De la mateixa manera, cada missatge transmès també es presentarà per pantalla com una línia nova.

5.1 Implementació de la capa d'aplicació

5.1.1 Part transmissora morse

La part de l'aplicació transmissora morse consisteix en anar rebent pel port sèrie el que escriu l'usuari amb el teclat i anar guardant aquests caràcters en una variable de

tipus message_frame_t. Aquesta recepció s'anirà fent fins trobar el caràcter sentinella (del port sèrie) [enter] (polsat per l'usuari). En aquest moment l'aplicació sol·licitarà la transmissió pel canal morse fent servir les funcions adequades (*frame_can_put()* i *frame_block_put()*). Si la funció *frame_can_put()* retorna un "False", l'aplicació quedarà bloquejada en aquest punt fins que es retorni un "True". Paral·lelament a aquestes accions, també es retornarà pel port sèrie el bloc de dades que s'ha transmès. El format d'aquest missatge d'eco serà "local: missatge". Aquest missatge d'eco ha de formar una nova línia a pantalla, per tant s'ha d'enviar també els caràcters de formatació (retorn de carro i/o avançament de línia) per a que quedi correctament visible a pantalla.

5.1.2 Part receptora morse

Aquesta part és molt més simple que l'anterior. Consisteix en definir un callback que s'executarà en el moment que hagi un missatge disponible per part de la capa Frame. La feina que ha de fer aquest callback és rebre rebre el missatge pròpiament dit a través de la funció *frame_block_get()* deixant el contingut del missatge en una variable de tipus *message_frame_t*. Posteriorment, enviarà aquest missatge a través del port sèrie per a que pugui ser visualitzat per la pantalla del PC. El format d'aquest missatge que s'envia pel port sèrie serà "remot: missatge". Aquest missatge també ha de formar una nova línia a pantalla, per tant s'ha d'enviar juntament amb el missatge els caràcters de formatació (retorn de carro i/o avançament de línia) per a que quedi correctament visible a pantalla.

5.1.3 Establiment de connexió

Per indicat que es vol establir una connexió s'enviarà el patró de text: ">ND[enter]". És a dir, el caràcter '>' determina amb qui es vol establir una nova connexió. A continuació s'escriu l'adreça de destí del node amb que s'estableix la connexió i per últim la tecla [enter] acaba la comanda. Quan l'aplicació detecta aquest patró, s'ha de cridar a la funció *frame_connect()*.

Per indicar que es vol alliberar la connexió s'enviarà el patró "/[enter]". Quan l'aplicació detecta aquest patró, s'ha de cridar a la funció *frame_disconnect()*.

6 Treball pràctic

1. Dibuixa el graf corresponent al transmissor adaptant el graf de les transparències de classe amb els noms de les funcions i esdeveniments corresponents a aquesta pràctica.
2. Dibuixa el graf corresponent al receptor adaptant el graf de les transparències de classe amb els noms de les funcions i esdeveniments corresponents a aquesta pràctica.

3. Implementa les aplicacions i mòduls corresponents al punt 3.3. Determina quines funcions de la API general del punt 4.1 implementes a cada mòdul.
4. Implementa l'aplicació i el mòdul Frame fusionat del punt 3.4.
5. Afegeix al mòdul Frame la gestió de la connexió.