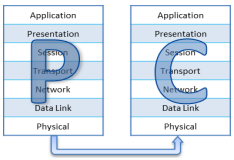
**Universitatea Politehnica București**
**Facultatea de Automatică și Calculatoare**




Computer Science & Engineering Department

# Protocoale de comunicații
# Tema1
# Legătura de date

Responsabili: Costin Raiciu(*costin.raiciu@cs.pub.ro*), Liviu Ioan(*liviu.ioan@cti.pub.ro*)
Termen de predare: 31 martie 2013, ora 23:50
Ultima modificare: 19 martie 2013, ora 19:50

Se cere să se implementeze un protocol cu fereastră glisantă folosit pentru a transmite un fișier.
Fișierul va fi citit de către transmițător(*send.c*) și trimis către receptor(*recv.c*).
Protocolul implementat trebuie să folosească eficient legătura de date - legătura de date trebuie menținută plină.
Implementarea va porni de la scheletul de cod pentru tema curentă.
Transmițătorul primește următoarele argumente:

- numărul cerinței rezolvate(0/1/2/3/4)
- numele fișierului care va fi transmis(*fileX*)
- viteza de transmisie
- timpul de propagare

Argumentul primit de receptor este:

- numărul cerinței rezolvate(0/1/2/3/4) - identic cu cel primit de transmițător

Receptorul va salva datele primite în fișierul recv_*fileX*.
Notă: o temă care implementează multiple cerințele va avea, la rulare, comportamentul indicat de primul parametru primit de transmițător și receptor.
Structura argumentelor pentru transmițător și receptor este obligatorie.
Reamintim caracteristicile legăturii de date simulate:

1. Viteza este constantă și timpul de propagare este constant(acești parametri sunt fixați; nu se modifică pe timpul execuției programului).
   Viteza de transmisie are valori între între 5 și 20 Mb/s.
   Timpul de propagare ia valori între 10 și 1000 ms.
2. Dacă un cadru $C_x$ este trimis înaintea unui cadru $C_y$, atunci receptorul va primi, mai întâi, $C_x$ și apoi $C_y$(pentru cazul în care cele două cadre nu se pierd).
3. Legătura de date poate pierde maxim 10% din cadrele trimise. Confirmările(trimise de către receptor) nu se pierd, nu se corup și sunt transmise instantaneu.
4. Legătura de date poate să corupă:
   - maxim 10% din cadrele trimise pentru cerința 3
   - 100% din cadre pentru cerința 4(*BONUS*)

Un cadru corupt conține *un singur octet* invalid (atât pentru cerința 3, cât și pentru cerința 4) în câmpul *payload* al structurii *msg*.

Parametrii pentru legătura de date se setează din scriptul *run_experiment.sh*. Așa cum s-a precizat, valorile valide sunt:

- SPEED = 5 ... 20
- DELAY = 10 ... 1000
- LOSS = 0 ... 10
- CORRUPT = 0 ... 10 / 100

**Cerințe**

0. Să se implementeze un protocol cu fereastră glisantă care utilizează eficient legătura de date pentru a transmite un fișier.
   Legătura de date nu pierde și nu corupe cadre.
   *Punctaj: 10 p*

1. Să se implementeze un protocol de tipul *Go-Back-N*.
   Legătura de date nu corupe datele, însă poate pierde cadre. Se pierd maxim 10% din cadre.
   *Punctaj: 10 p*

2. Să se implementeze un protocol cu retransmitere selectivă(*Selective Repeat*).
   Legătura de date nu corupe datele, însă poate pierde cadre. Se pierd maxim 10% din cadre.
   *Punctaj: 40 p*

3. Să se implementeze un mecanism de detecție pentru cadrele eronate.
   Legătura de date poate pierde/corupe cadrele.
   Se pierd maxim 10% din cadre. Se corup maxim 10% din cadre.
   *Punctaj: 20p*

4. *BONUS* Să se implementeze un cod corector - pentru a evita retransmisiile.
   Legătura de date corupe toate cadrele(CORRUPT=100).
   Punctajul obținut va fi în funcție de eficacitatea codului corector.
   Cu cât informația redundantă este mai redusă, cu atât punctajul acordat va fi mai mare.
   *Punctaj: 25p*

   Se acordă *20 p* pentru *README*, care va conține explicațiile aferente fiecărei cerințe rezolvate. Punctajul acordat fiecărei cerințe rezolvate depinde de eficiența implementării. Concret, se va urmări respectarea protocolului indicat de cerința respectivă. "Eficiența" nu se referă la utilizarea unor structuri de date speciale sau la optimizări la nivel de limbaj de programare.

**API Simulator**

- *int send_message(msg* m)*
  - parametru: mesajul care va fi trimis
  - rezultat: numărul de octeți transferați(în caz de succes) sau *-1* în caz de eroare
- *int recv_message(msg* m)*
  - parametru: adresa la care se memorează datele primite
  - rezultat: numărul de octeți recepționați(în caz de succes) sau *-1* în caz de eroare
  - apelul este *blocant* - se așteaptă până la primirea unui mesaj
- *int recv_message_timeout(msg *m, int timeout)*
  - parametri:
    - *timeout* - timpul scurs până la ieșirea din apel; măsurat în milisecunde
    - *m* - adresa la care se memorează mesajul primit
  - rezultat:
    - *-1* dacă nu s-a primit un mesaj în timpul *timeout* sau în caz de eroare
    - numărul de octeți al mesajului recepționat(dacă se primește un mesaj)
  - apelul se blochează până la recepția primului cadru sau până ce expiră *timeout* ms

**Detalii de implementare**

Pentru transmiterea mesajelor, se va folosi structura *msg* din lib.h.
Nu este permis să fie modificată structura *msg*.
Valorea utilizată pentru *timeout* trebuie să fie: $max(1s, 2 * DELAY)$.
Nu aveți nevoie să utilizați alte mecanisme pentru timeout în afara celui pus la dispoziție(*recv_message_timeout*).
Rezolvarea temei va porni de la scheletul de cod pentru tema 1.
Pentru lucrul cu fișiere, veți utiliza funcțiile *open*, *read*, *write*, și *close*. În acest sens, puteți consulta rezolvarea laboratorului 2 sau paginile de manual din Linux(*man 2 open*).
Dimensiunea fișierului transmis este între 100KB(kilobytes) și 10MB(megabytes).
Se recomandă folosirea utilitarului *dd* pentru a genera fișiere de test.
Exemplu: *dd if=/dev/urandom of=test_file bs=512k count=4*

**Arhiva temei**

Denumire arhivă: Nume_Prenume_Grupa_Tema1.zip
Conținut arhivă:

- send.c
- recv.c
- lib.h (dacă s-au adus modificări)
- alte surse adăugate de voi
- README (format txt)

Nu este necesar să includeți dosarul *link_emulator*.