PROTOCOALE DE COMUNICAȚIE : LABORATOR 3

Implementarea detectării și corectării erorilor

Responsabil: Alecsandru PĂTRAȘCU

Cuprins

Cerințe laborator

În cadrul laboratorului curent, legătura de date va corupe informația transmisă. Se cere să se implementeze mecanisme prin care datele corupte să fie detectate/corectate.

Astfel, transmițătorul construiește pachetele și le trimite către receptor. Receptorul primește pachetele, însă datele din *payload* nu coincid neapărat cu cele trimise de transmițător.

Se garantează că fiecare pachet primit de receptor conține **cel mult 1 bit eronat** în cadrul zonei de *payload*. De asemenea, numărul de octeți transferați de legătura de date nu este afectat (nu se pierd date).

Detaliere laborator

Atunci când legătura de date corupe informațiile transmise, există următoarele mecanisme pentru a gestiona erorile apărute:

1. Detecția erorilor
   - Transmițătorul adaugă informații de redundanță în pachetele trimise, astfel încât receptorul să poată identifica un pachet care nu este valid.
   - Pentru problema din laborator, în care fiecare pachet conține cel mult 1 bit eronat, se poate folosi o verificare de *paritate*.
   - Concret, informația de redundanță adăugată reprezintă suma modulo 2 a tuturor biților din *payload*-ul pachetului.

- La recepție, se calculează suma biților modulo 2 din *payload*. Dacă rezultatul calculului este identic cu valoarea precizată de transmițător, atunci pachetul este valid. Altfel, pachetul este eronat și transferul a eșuat.

2. Corectarea erorilor
   - Transmițătorul adaugă informații de redundanță în pachetele trimise, astfel încât receptorul să poată corecta un pachet care nu este valid.
   - Pentru a corecta bitul greșit, se va utiliza un cod Hamming (explicații mai jos).
   - În acest fel, transferul se poate efectua cu succes.

Coduri Hamming

Pentru a recupera un pachet eronat, se poate utiliza un cod Hamming.

Transmițătorul adaugă o serie de biți de paritate care sunt responsabili pentru anumite poziții din cadrul zonei de *payload*.

Notații:

- n - numărul de biți de date efective
- k - numărul de biți de paritate necesari.

Dorim să transmitem n biți de date. Care este numărul necesar k de biți de paritate?

Pentru codurile Hamming, dorim ca numărul format din alăturarea biților de paritate (**sindromul**) să ne indice poziția eronată din mesajul transmis.

Lungimea mesajului transmis este de $n + k$ biți.

De asemenea, dorim ca acest numar să ne indice și dacă mesajul este valid (în cazul în care nu există o poziție eronată).

Așadar, sindromul este un număr între 0 și $n + k$. Dacă sindromul este egal cu 0, atunci nu există erori. O valoare a sindromului între 1 și $n + k$ reprezintă poziția eronată din cadrul mesajului.

În concluzie, numărul necesar de biți de paritate rezultă din condiția $2^k \geq n + k + 1$.

Algoritm

Biții de paritate sunt inserați, în cadrul mesajului, pe pozițiile de forma $2^i, i \in 0, 1, ..., k - 1$.

Bitul de paritate de pe poziția 2^i răspunde de acele poziții care au în reprezentarea binară valoarea 1 pe poziția i.

i	2^i	Poziții verificate
0	1	1, 3, 5, 7, 9, 11, 13, 15, 17
1	2	2, 3, 6, 7, 10, 11, 14, 15, 18
2	4	4, 5, 6, 7, 12, 13, 14, 15, 20
3	8	8, 9, 10, 11, 12, 13, 14, 15, 24

Exemplu: se dorește transmiterea unui mesaj de 4 biți, deci $n = 4$.

Din condiția $2^k \geq n + k + 1$, avem $k = 3$.

In total, vor fi transmiși 7 biți:

- 4 biți de date (D_0, D_1, D_2 si D_3)
- 3 biți de verificare (P_0, P_1 si P_2).

Poziție	1	2	3	4	5	6	7
Conținut	P_0	P_1	D_0	P_2	D_1	D_2	D_3

Mesajul are forma:

La transmisie:

$P_0 = D_0 + D_1 + D_3$

$P_1 = D_0 + D_2 + D_3$

$P_2 = D_1 + D_2 + D_3$

La recepție:

Se recalculează valorile pentru cei 3 biți de paritate. **Atentie, biții de paritate pot fi și ei eronați, deci trebuie luați în calcul!**

$P_0 = P_0 + D_0 + D_1 + D_3$

$P_1 = P_1 + D_0 + D_2 + D_3$

$P_2 = P_2 + D_1 + D_2 + D_3$

Sumele calculate reprezinta sume *modulo 2*.

Sindromul este $P_2P_1P_0$

Dacă sindromul este egal cu 0 ($P_0 = P_1 = P_2 = 0$), atunci nu există erori. Altfel, sindromul ne va indica poziția care trebuie negată pentru a corecta mesajul.

Detalii de implementare

1. Implementarea va porni de la scheletul de cod atașat laboratorului.
2. O modalitate facila de a determina suma *modulo 2* este folosirea operatorului *XOR*.

Pași de rezolvare

1. Se adaugă câmpurile necesare unui mesaj pentru detecția erorilor.
2. Se determină numărul de pachete eronate primite de receptor.
3. **BONUS:** Se implementează corectarea erorilor din mesaje (folosind un cod Hamming), astfel încât transmisia să aibă loc cu succes.

Notă: nu trebuie să fie modicată structura definită în *link_emulator/lib.h* - se vor utiliza doar câmpurile *len* și *payload*.

Software disponibil

1. Simulator legatură de date - executabilul *link* generat în urma comenzii *make*
2. Schelet de cod pentru transmițător și receptor
3. API simulator:
   - *int send message(msg* m)*
     - parametru: mesajul care va fi trimis
     - rezultat: numărul de octeți transferați (în caz de succes) sau -1 in caz de eroare
   - *int recv message(msg* m)*
     - parametru: adresa la care se memorează datele primite
     - numărul de octeți recepționați (în caz de succes) sau -1 în caz de eroare
4. Compilare schelet de cod: *make*.