PROTOCOALE DE COMUNICAȚIE : LABORATOR 3

Implementarea detectarii si corectarii erorilor

Responsabil: Alecsandru PĂTRAȘCU

Cuprins

Cerinte laborator

In cadrul laboratorului curent, legatura de date va corupe informatia transmisa. Se cere sa se implementeze mecanisme prin care datele corupte sa e detectate/corectate.

Astfel, transmitatorul construieste pachetele si le trimite catre receptor. Receptorul primeste pachetele, insa datele din *payload* nu coincid cu cele trimise de transmitator.

Se garanteaza ca fiecare pachet primit de receptor contine *cel mult 1 bit eronat* in cadrul zonei de payload. De asemenea, numarul de octeti transferati de legatura de date nu este afectat (nu se pierd date).

Detaliere laborator

Atunci cand legatura de date corupe informatiile transmise, exista urmatoarele mecanisme pentru a gestiona erorile aparute:

1. Detectia erorilor
   - Transmitatorul adauga informatii de redundanta in pachetele trimise, astfel incat receptorul sa poata identica un pachet care nu este valid.
   - Pentru problema din laborator, in care fiecare pachet contine cel mult 1 bit eronat, se poate folosi o vericare de *paritate*.
   - Concret, informatia de redundanta adaugata reprezinta suma modulo 2 a tuturor bitilor din *payload*-ul pachetului.

- La receptie, se calculeaza suma bitilor modulo 2 din payload. Daca rezultatul calculului este identic cu valoarea precizata de transmitator, atunci pachetul este valid. Altfel, pachetul este eronat si transferul a esuat.

2. Corectarea erorilor
   - Transmitatorul adauga informatii de redundanta in pachetele trimise, astfel incat receptorul sa poata corecta un pachet care nu este valid.
   - Pentru a corecta bitul gresit, se va utiliza un cod Hamming (explicatii mai jos).
   - In acest fel, transferul se poate efectua cu succes.

Coduri Hamming

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

Transmitatorul adauga o serie de biti de paritate care sunt responsabili pentru anumite pozitii din cadrul zonei de *payload*.

Notatii:

- n - numarul de biti de date efective
- k - numarul de biti de paritate necesari

Intrebare: dorim sa trimitem n biti de date. Care este numarul necesar de biti de paritate k?

Pentru codurile Hamming, dorim ca numarul format din alaturarea bitilor de paritate (*sindromul*) sa ne indice pozitia eronata din mesajul transmis.

Lungimea mesajului transmis este de n + k biti.

De asemenea, dorim ca acest numar sa ne indice si daca mesajul este valid (in cazul in care nu exista o pozitie eronata).

Asadar, sindromul este un numar intre 0 si n + k. Daca sindromul este egal cu 0, atunci nu exista erori. O valoare a sindromului intre 1 si n + k reprezinta pozitia eronata din cadrul mesajului.

Concluzie: numarul necesar de biti de paritate rezulta din conditia $2^k \geq n + k + 1$.

Algoritm

Bitii de paritate sunt inserat, in cadrul mesajului pe pozitiile de forma $2^i, i \in 0, 1, ..., k-1$.

Bitul de paritate de pe pozitia 2^i raspunde de acele pozitii care au in reprezentarea binara valoarea 1 pe pozitia i.

i	2^i	Pozitii 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 doreste transmiterea unui mesaj de 4 biti.

Mesajul contine 4 biti, deci n = 4.

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

In total, vor transmisi 7 biti:

- 4 biti de date (D_0, D_1, D_2 si D_3)

- 3 biti de verificare (P_0, P_1 si P_2)

Mesajul are forma:

Pozitie	1	2	3	4	5	6	7
Continut	P_0	P_1	D_0	P_2	D_1	D_2	D_3

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 receptie:

Se recalculeaza valorile pentru cei 3 biti de paritate. *Atentie, bitii de paritate pot fii si ei eronati, deci trebuie luati in 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: $P_2P_1P_0$

Daca sindromul este egal cu 0 ($P_0 = P_1 = P_2 = 0$), atunci nu exista erori. Altfel, sindromul ne va indica pozitia care trebuie negata pentru a corecta mesajul.

Detalii de implementare

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

Pasi rezolvare

1. Se adauga campurile necesare unui mesaj - pentru detectia erorilor.
2. Se determina numarul de pachete eronate primite de receptor.
3. *BONUS:* Se implementeaza corectarea erorilor din mesaje (folosind un cod Hamming), astfel incat transmisia sa aiba loc cu succes.

Nota: nu trebuie sa fie modicata structura definita in *lib.h* - se vor utiliza doar campurile *len* si *payload*.

Software disponibil

1. Simulator legatura de date - executabilul *link* - generat in urma comenzii *make*
2. Schelet de cod pentru transmitator si receptor
3. API simulator:
   - *int send message(msg* m)*
     - parametru: mesajul care va trimis
     - rezultat: numarul de octeti transferati(in caz de succes) sau -1 in caz de eroare
   - *int recv message(msg* m)*
     - parametru: adresa la care se memoreaza datele primite
     - numarul de octeti receptionati(in caz de succes) sau -1 in caz de eroare
4. Compilare schelet de cod: *make*