Detecția și corecția erorilor

Moraru Andrei-Lucian

Clăbescu Cătălin

grupa 443A

Cuprins

Capitolul 1 – Noțiuni introductive despre informație și erori (Moraru Andrei-Lucian).....3

Capitolul 2 – Coduri polinomiale. Codul Reed-Solomon (Moraru Andrei-Lucian).....9

Capitolul 3 – Utilizarea CRC în rețelele de calculatoare (Clăbescu Cătălin).....15

Capitolul 4 – Concluzii (Clăbescu Cătălin).....20

Bibliografie 22

Capitolul 1 – Noțiuni introductive despre informație și erori

În rețelele de calculatoare, ca dealtfel în orice dispozitiv care transmite informație, apariția erorilor este inevitabilă. Deși unele medii de transmisie precum fibra optică sunt construite astfel încât apariția erorilor să fie minimă, acest lucru nu este o regulă. Conceptele de detecție și corecție a erorilor se bazează amândouă pe un concept simplu, anume acela de a adăuga o redundanță în transmisie. Singura diferență este în cantitatea de redundanță, care este mai mică în cazul detecției (unde putem doar să spunem că s-a produs o eroare și să cerem retransmiterea datelor) și mai mare în cazul corecției, unde utilizatorul poate să deducă felul în care trebuiau să arate datele transmise. Asupra mecanismelor de detecție și corecție vom reveni un pic mai târziu în lucrare. Pentru început, vom da câteva definiții importante despre teoria informației, care ne vor ajuta să înțelegem mai bine anumite concepte utilizate în operațiile de detecție și corecție.

Teoria informației răspunde la două întrebări fundamentale în telecomunicații. Prima dintre ele se referă la cantitatea maximă de informație ce poate fi transmisă printr-un canal de comunicație. În anii 40, comunitatea științifică a crezut că mărind cantitatea de informație transmisă printr-un canal, crește și probabilitatea eronării ei. Shannon a surprins însă lumea științifică, arătând că transmisia poate fi făcută corect, cu condiția ca rata de transmisie să nu depășească o mărime numită *capacitatea canalului*. Această mărime se poate calcula din caracteristicile zgomotului existent în canal.

A doua problemă se referă la capacitatea maximă de compresie a datelor. Shannon a arătat că datele reprezentând procese aleatoare ca muzica sau vorbirea nu pot fi compresate sub o anumită valoare limită pe care a numit-o *entropie*, un termen folosit deja în termodinamică. Apoi a arătat că dacă entropia este mai mică decât capacitatea canalului, atunci transmisia datelor se poate face fără erori.

În continuare, vom prezenta câteva definiții relevante legate de informație. În primul rând, *informația* este o mărime măsurabilă care se referă la un experiment care poate avea mai multe rezultate. Informația obținută prin obținerea unui rezultat anume este definită ca fiind:

$$i = \log_a \frac{1}{p}$$

unde prin P am notat probabilitatea acelui rezultat. Dacă un experiment poate avea N rezultate posibile, atunci informația devine:

$$i = \log_a N$$

Un *semnal* este o manifestare fizică, capabilă să se propage printr-un mediu dat. În mod restrâns, noțiunea se referă numai la acele manifestări fizice care transmit informație. Sunt excluse acele semnale care dăunează transimisiei, denumite perturbații.

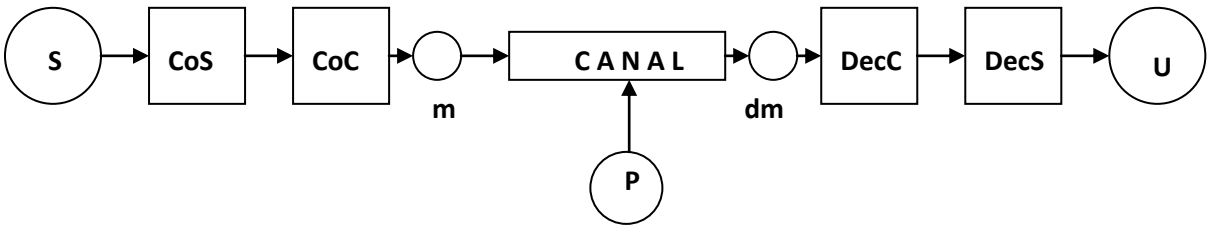
O *perturbație* este un semnal care modifică în mod necontrolat semnalul aleator util care transmie informația.

Un *mesaj* este un semnal care corespunde unei realizări particulare printr-un ansamblu de idei, imagini sau date. El poate îmbrăca o haină fizică diferită.

O *sursă* este un mecanism prin care dintr-o mulțime de mesaje posibile se alege în mod imprevizibil un mesaj particular destinat a fi transmis unui corespondent.

Un *corespondent*, denumit și utilizator, este destinația finală la care trebuie să ajungă mesajul.

Canalul de transmisie reprezintă totalitatea mijloacelor de transmisie destinate transportului mesajului de la sursă la destinație. Prin canal se înțelege atât mediul cât și aparatura prin care se face transmisia. Schema unui canal de transmisie este următoarea:



S - sursă de mesaje

CoS - Codor de sursă (compresia datelor)

CoC - Codor de canal (protecție contra perturbațiilor)

M - modulator

CANAL - Canal de comunicație

P - Perturbații

DecC - Decodor de canal

DecS - Decodor de sursă

U - Utilizator

În ceea ce privește unitatea de măsură a informației, ea se obține din baza de logaritmare. Cea mai des folosită este bit-ul, care corespunde bazei de logaritmare 2. Pe lânga ea mai avem dit-ul și nit-ul, care corespund bazelor de logaritmare 10, respectiv e. Aceste unități de măsură nu sunt independente între ele, existând o dependență numerică între cele 3. În cele ce urmează în această lucrare, vom folosi bit-ul ca unitate de măsură a informației.

Teorema eșantionării este una din cele mai importante teoreme din teoria informației. Ea spune că pentru transmisia unui semnal analogic nu este necesară transmisia sa pe întreaga durată a axei timpului, ci este suficient să se transmită eșantioane ale acestui semnal la intrevale de timp mai mici sau egale cu $\frac{1}{2*w}$ unde cu w am notat cea mai mare frecvență din spectrul semnalului.

Prin *codare* înțelegem stabilirea unei corespondențe între două limbaje, această corespondență asigurând trecerea de la un limbaj A la un limbaj B. *Decodarea* este operația inversă.

Entropia unei surse este informația medie pe simbol. Medierea se face ponderat, iar formula de calcul este următoarea :

$$H(x) = -\sum_{i=1}^{n} p(x_i) * \log_2 \mathrm{p}(\mathrm{x_i})$$

Valoarea maximă a acestei entropii este:

$$H_{max}(x) = \log_2 n$$

unde n e numărul de simboluri ce pot fi produse de o sursă.

Redundanța unei surse este:

$$R = H_{max}(x) - H(x)$$

iar unitatea de măsură pentru aceste 3 mărimi este bit/simbol.

Detecția și corecția erorilor sunt 2 etape foarte importante în transmiterea informației. În teorie există și așa numitele canale fără zgomot, în care aceste 2 etape nu se regăsesc întrucât nu există factori perturbatori care să ducă la apariția erorilor. Cum însă în practică astfel de surse nu există, apare posibilitatea de eroare în transmisie.

În cazul canalelor cu perturbații, prin codare nu urmărim reducerea sau anularea redundanței unei surse, ci introducem o redundanță suplimentară controlată cu scopul de a detecta și/sau corecta erorile introduse de zgomot în canal. Prin canal se transmit simboluri binare iar pe lângă aceste simboluri purtătoare de informație (numite simboluri utile sau informaționale) se mai transmit alte simboluri redundante calculate prin anumite reguli.

Regulile de calcul sunt cunoscute și la recepție și prin verificarea lor se stabilește dacă s-au introdus erori și se și corectează daca sistemul permite acest lucru. Corecția înseamnă găsirea poziției simbolurilor care s-au transformat din 0 în 1 sau din 1 în 0 și schimbarea simbolului recepționat cu valoarea opusă.

Utilizarea redundanței suplimentare duce și la utilizarea doar parțială capacității canalului. De asemenea trebuie menționat că în cazul utilizării corecției erorilor este necesară o redundanță mărită, ceea ce duce la scăderea și mai mult a informației transmise.

Detecția erorilor presupune introducerea în lanțul de transmisie a unui canal de reacție. Acest canal are rolul de a cere retransmiterea semnalul în cazul în care s-au detectat erori.

Corespondența dintre mesajul sursei și simbolurile de 1 și 0 poartă numele de codare. Codurile sunt de 2 tipuri :

1. coduri bloc: grup, ciclice (folosesc polinoame și sunt utilizate foarte mult în detecția și corecția erorilor)
2. convoluționale

Cuvântul de cod reprezintă vectorul format din simbolurile informaționale și cele de control (aparținând redundanței suplimentare).

Tipurile de erori care apar la transmisiunile de date sunt:

- erorile simple, în cazul cărora este alterată valoarea unui singur simbol;
- erorile duble, pentru care se modifică valorile a 2 simboluri;
- erorile triple, caz în care sunt transformate 3 simboluri;
- pachete de erori, care reprezintă succesiuni de simboluri în care primul și ultimul sunt eronate, iar celelalte simboluri pot fi eronate sau nu.

Mecanismul de detecție și corecție a erorilor – constă în trei etape. În prima din ele se realizează detecția, celelalte două fiind rezervate corecției. Pentru detecție se aplică un operator asupra vectorului $v_i^{'} = v_i + \varepsilon$, unde prin ε am definit vectorul eroare, în urma căreia se calculează un vector corector numit sindrom și notat z_i. Valorile lui confirmă sau infirmă transmisia corectă a lui v_i. Dacă dorim să avem și corecție, trebuie aplicat un alt operator pe vectorii corectori în urma căruia să rezulte vectorul transmis v_i. Etapa de detecție este identică pentru toate sistemele, însă cea de corecție este diferită de la un cod la altul.

Pentru detecția erorilor sunt folosite, pe lângă codurile menționate mai sus, alte 2 metode:

- Paritatea unidimensională.
- Paritatea bidimensională.

In cazul *parității unidimensionale* se adaugă la mesaj un bit de paritate, care face ca numărul de biți de 1 din mesaj să fie par sau impar.

Dacă un singur bit se alterează în timpul transmisiei acesta va schimba paritatea de la pară la impară sau invers. Emițătorul generează bitul de paritate prin sumarea modulo 2, adică prin aplicarea unui "sau exclusiv" tuturor biților din mesaj. După aceea bitul de paritate sau complementul lui se adaugă la mesaj. Receptorul poate verifica mesajul prin sumarea modulo 2 a biților și apoi comparând rezultatul cu bitul de paritate. O operație echivalentă este sumarea modulo 2 atât a biților din mesaj cât și a bitului de paritate și apoi se verifică dacă rezultatul este 0(aceasta in cazul unei paritati pare).

Această tehnică detectează erorile simple(modificarea valorii unui singur bit). De fapt detectează erorile în orice număr impar de biți, dar nu realizează și detectia unui număr de erori pare.

Paritatea bidimensională este cunoscută sub numele de paritate longitudinală/verticală. Informația este dispusă pe linii formând o matrice. Acest tip de paritate folosește atât câte un bit de paritate pentru fiecare linie a matricei dar și un cuvânt de paritate reprezentat de ultima linie a matricei, în care fiecare bit reprezintă paritatea coloanei corespunzătoare. Bitul de paritate al fiecarei linii se găsește pe ultima coloană din matrice, care poartă denumirea de VRC(Vertical Redundancy Check), în timp ce biții de paritate pentru fiecare coloană formează ultima linie a matricei, numită LRC(Longitudinal Redundancy Check) sau cuvânt de paritate.

Acestea ar fi noțiunile introductive cele mai importante. În cele ce urmează vom prezenta cele mai importante coduri de tip polinomial, cu accent pe codul Reed-Solomon, aplicațiile acestora, precum și cu o prezentare teoretică și aplicată a Controlului Redundant Ciclic (CRC).

Capitolul 2 – Coduri polinomiale. Codul Reed-Solomon

Codurile polinomiale sunt un exemplu de coduri ciclice care tratează șirurile de biți de 1 și 0 în relație cu coeficienții unui polinom, după cum se va arăta puțin mai jos. Codurile ciclice constituie un caz particular al codurilor grup. Denumirea de ”ciclic” provine de la faptul că orice permutare ciclică a unui cuvânt de cod valid produce tot un cuvânt de cod valid. Permutarea cilică a unui cuvânt se obține luând primul element și punându-l la sfârșitul cuvântului.

Formalizând, fie cuvântul de cod $v_{n-1}v_{n-2}v_{n-3} \ldots v_3v_2v_1v_0$. Un cuvânt obținut printr-o permutare ciclică din acesta este $v_{n-2}v_{n-3}v_{n-4} \ldots v_2v_1v_0v_{n-1}$. După mai mulți pași, prin pemutări consecutive se poate obține cuvântul : $v_0v_{n-1}v_{n-2} \ldots v_3v_2v_1$.

De exemplu dacă 10111 este un cuvânt de cod (conform unui cod cilcic), atunci și 01111 este un cuvânt de cod, de altfel ca și 11110, 11101, 11011.

Lungimea cuvântului de cod se notează cu n. Dintre cele n simboluri k sunt de informație, iar $m = n - k$ de control. Dacă în cazul codurilor grup reprezentarea era sub formă de vectori, în cazul codurilor ciclice reprezentarea este sub formă de polinom.

Polinomul de informație $i(x)$ are gradul $k - 1$ și este de forma:

$$i(x) = i_{k-1} * x^{k-1} + i_{k-2} * x^{k-2} + \ldots + i_1 * x + i_0$$

unde $[i_{k-1}, i_{k-2}, \ldots i_0]$ sunt simbolurile de informație.

Cuvântul de cod este reprezentat printr-un polinom de grad $n - 1$:

$$v(x) = v_{k-1} * x^{n-1} + v_{k-2} * x^{n-2} + \ldots + v_1 * x + v_0$$

Similar cu matricea generatoare, G, de la coduri grup, aici se utilizează polinomul generator, notat $g(x)$. Gradul acestuia este m:

$$g(x) = g_m * x^m + g_{m-1} * x^{m-1} + \ldots + g_1 * x + g_0$$

Pentru a fi un polinom generator al unui cod funcțional, $g(x)$ trebuie să fie polinom primitiv. Într-o simplificare a definiției în care se ține cont și de faptul că coeficienții polinomului pot fi doar 0 sau 1, un polinom este primitiv dacă este:

• *ireductibil*: acest lucru presupune că nu există nici un polinom de grad nenul și mai mic decât *m* care să fie divizor al lui $g(x)$

• dacă α este o rădăcină a lui $g(x)$ ($g(\alpha) = 0$) atunci cel mai mic întreg *T* pentru care $\alpha^i = \alpha^{i+T}$ este $T = 2^m - 1 = 2^{grad(g)} - 1$.

De reținut este că polinoamele primitive sunt listate (cunoscute), iar de importanță practică este faptul că coeficienții de grad maxim (g_m) și minim (g_0) sunt nenuli: $g_m = 1$, $g_0 = 1$.

Echivalent matricei de control de la coduri grup, în cazul codurilor ciclice, se definește un polinom de control, $h(x)$. Acesta are gradul $k - 1$ și se află în relația următoare cu polinomul generator:

$$g(x)h(x) = x^n + 1$$

După cum am spus, codurile polinomiale leagă un șir de biți de 1 și 0 de coeficienții unui polinom. Se știe că pentru un număr reprezentat în binar, adică folosind biți de 1 și 0, utilizează puterile lui 2 pentru a-i reprezenta valoarea numerică. Pentru un cuvânt de cod binar, biții de 1 semnifică puterile respective ale lui x care vor apărea în reprezentarea polinomului. Pentru a ilustra acest concept, să dăm un exemplu. Fie cuvântul de cod 110101. Acesta are biți de 1 pe pozițiile corespunzătoare puterilor 5, 4, 2 și 0 ale lui 2, prin urmare polinomul asociat acestui cod este:

$$p(x) = x^5 + x^4 + x^2 + 1$$

Aritmetica polinomială este de tip modulo 2. Nu există transport la adunare şi nici împrumut la scădere (adunările şi scăderile sunt identice cu SAU EXCLUSIV) iar scăderea este realizată modulo 2.

În ceea ce privește practica, polinomul folosit în IEEE 802 este:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

Codul Reed-Solomon

Codurile Reed-Solomon sunt tot un exemplu de coduri ciclice, cu deosebirea că nu sutn coduri binare, ele nefolosind câmpul binar format din simbolurile 0 și 1. Pentru aceste coduri se utilizează un câmp finit de ordin superior numit câmp Galois. Astfel, cuvintele de cod de tip RS nu sunt succesiuni de biți, ci succesiuni de caractere. Deși aceste caractere pot fi

reprezentate cu ajutorul câmpului binar, din punct de vedere al codării și decodării RS sunt indivizibile.

Cuvintele de cod RS au aceeași structură ca și codurile ciclice iar relația de codare este aceeași cu cea folosită în codurile ciclice:

$v(x) = i(x)*x^m + \text{rest}\,(i(x)*x^m/g(x))$

Prin această relație de codare se obține polinomul atașat cuvântului de cod, polinom ai cărui coeficienți sunt tocmai caracterele ce alcătuiesc cuvântul de cod. Relaţia de codare indică, de asemenea, că v(x) este un multiplu al lui g(x).

Codul RS, având parametrii n, k şi m, este capabil să corecteze un număr e_c de caractere eronate, unde: $2*e_c = m = n - k$. La decodare, spre deosebire de codurile ciclice, într-un cuvânt de cod RS recepţionat, în vederea corecţiei, este necesară atât localizarea erorii, cât şi stabilirea valorii ei.

Conceptul original al codării RS se referea la codarea mesajelor de k simboluri, considerând aceste simboluri drept coeficienții unui polinom p(x) cu grad maxim k-1 pe un câmp finit de ordin N. Evaluarea acestui polinom se face folosind un număr n, mai mare decât k, de puncte de intrare. Prin eșantionarea unui polinom de grad k-1 cu un număr de puncte mai mare decât k duce la un sistem supradeterminat și permite refacerea codului la recepție prin utilizarea interpolării Lagrange dacă ni se dau oricare k din cele n puncte de eșantionare. Secvența de puncte este creată de un generator al multiplilor valorilor din câmpul finit și include și valoarea 0, permițând orice valoare a lui n (care poate ajunge și până la N).

În practică, codurile RS nu sunt văzute ca coduri ciclice, ci sunt considerate coduri BCH. În acest caz, simbolurile codate sunt considerate drept coeficienții unui polinom de ieșire s(x) construit prin multiplicarea polinomului mesaj p(x) de grad k-1 cu un polinom generator g(x) de gard t=N-k-1. Emițătorul transmite cei N-1 coeficienți ai lui s(x), iar receptorul utilizează împărțirea polinomială prin g(x) a polinomului recepționat pentru a determina dacă s-au produs erori. Dacă restul acestei împărțiri este diferit de 0, atunci au fost detectate erori. Dacă notăm cu r(x) acest rest, atunci receptorul va evalua acest rest bazându-se pe g(x) și va putea construi un sistem care determină care coeficienți au fost transmiși eronat. Dacă ecuațiile acestui sistem pot fi rezolvate, receptorul știe cum să modifice r(x) în scopul obținerii polinomului s(x) corect.

La prima vedere, aceste 2 definiții par complet diferite. În primul caz cuvintele de cod reprezintă valorile unui polinom pe când în al doilea caz sunt coeficienții unui polinom. În plus, codurile polinomiale din primul caz trebuie să aibă grad mic, pe când în al doilea caz aceste polinoame trebuie să aibă anumite rădăcini. Echivalența celor două definiții este demonstrată folosind transformarea Fourier discretă, care stabilește o legătură dual între coeficienții unui polinom și valorile unui polinom.

Proprietățile codului RS

Codul Reed-Solomon este un cod bloc liniar de lungime n pe un câmp finit F cu dimensiune k și distanța Hamming minimă n-k+1. Codul Reed-Solomon este optimal în sensul că distanța minimă are valaorea maximă posibilă pentru un cod liniar de dimensiune (n, k).

Abilitățile de corecție a acestui cod sunt determinate de distanța minimă sau de mărimea n-k, care reprezintă redundanța din bloc. Dacă pozițiile simbolurilor eronate nu sunt cunoscute, atunci un cod RS poate corecta până la (n-k)/2 simboluri eronate. Altfel spus, el poate corecta un număr de erori egal cu jumătate din numărul de simboluri redundante adăugate blocului. Uneori, pozițiile eronate sunt cunoscute dinainte, aceste poziții purtând denumirea, în limba engleză, de *erasures*. Un cod RS poate corecta de două ori mai multe erasures-uri decât erori iar mai general vorbind, orice combinație de erasures și erori poate fi corectată atât timp cât se respectă condiția $2*E+S \leq n-k$, unde cu E am notat numărul de erori iar cu S am notat numărul de erasures din bloc.

În practică, pentru codurile RS se folosește un câmp finit cu 2^m elemente. În acest caz fiecare simbol poate fi reprezentat ca o valoare pe un număr m de biți. Emițătorul transmite datele sub forma unor blocuri codate iar numărul de elemente dintr-un bloc este $n=2^m-1$. Un caz particular și des utilizat este cel pentru care m=8. În ceea ce privește dimensiunea k, aceasta reprezintă un parametru care dă numărul de simboluri de date dintr-un bloc. De obicei k se alege 223, astfel că într-un bloc de date vom avea 223 simboluri de date și 32 de simboluri de paritate, toate reprezentate pe 8 biți pentru cazul când m=8. Un astfel de cod este capabil să corecteze până la 16 erori de simbol per bloc.

Aceste proprietăți ale codurilor RS le fac să se plieze excelent pe aplicațiile în care erorile pot apărea sub forma unor burst-uri. Acest lucru se întâmplă deoarece pentru cod nu are importanță câți biți sunt eronați. Dacă mai mulți biți ai unui simbol sunt eronați, acest lucru va fi contabilizat drept o singură eroare. De menționat este că dacă avem de-a face cu șiruri de date pentru care erorile sunt aleatoare sau nu conțin grupuri de mai mulți biți, atunci codurile Reed-Solomon sunt o alegere proastă în comparație cu un cod binar.

Codul Reed-Solomon, ca și un cod convoluțional, este transparent. Acest lucru înseamnă că chiar dacă simbolurile canalului au fost inversate la un moment dat în timpul transmisier, decodoarele continuă să funcționeze. Rezultatul va fi reprezentat de datele originale, dar inversate. Pe de altă parte, această transparență dispare în cazul în care codul este scurtat. Biții lipsă trebuie completați cu valori de 0 sau 1 în funcție de tipul datelor din punct de vedere al complementării lor. Acest lucru trebuie stabilit înainte de a începe decodarea.

Aplicații și utilizări ale codurilor Reed-Solomon

Codurile Reed-Solomon au ajuns să aibă o utilizare foarte variată de-a lungul timpului, de la comunicații în spațiu la electronice și electrocasnice. Ca și exemple, ele sunt intens folosite la realizarea CD-

urilor, DVD-urilor, Blu-Ray-urilor, în comunicații de date precum DSL și WiMAX, în sisteme de broadcast precum DVB și ATSC și în aplicații de calculatoare cum ar fi sistemele RAID 6.

În continuare vom prezenta câteva aplicații mai deosebite ale acestor coduri.

Prima dintre ele este cea de stocare a datelor. Codurile RS sunt utilizate aici pentru a corecta erorile de burst asociate defectelor media. Codarea RS este una din părțile cheie de realizare a compact disc-urilor. A fost pentru prima dată când s-a folosit o tehnică de codare cu corecție a erorilor în produsele de larg consum iar DVD-urile și DAT-urile folosesc același principiu. În cazul CD-ului, 2 straturi de coduri RS separate de o suprafață de întrețesere convoluțională cu 28 de direcții formează o schemă numita Cross-Interleaved Reed-Solomon Coding (CIRC). Primul element al acestei scheme este un cod RS interior de dimensiune mică (32, 28), care este obținut prin micșorarea unui cod de dimensiune (255,251) cu simboluri pe 8 biți. Acest cod poate corecta până la 2 bytes de erori pentru un bloc de 32 de bytes dimensiune. Pe lângă aceasta, marchează drept erasures toate blocurile care au mai mult de 2 bytes de erori în ele. Blocurile decodate de 28 de bytes, cu indicatorii de erasures, sunt desfăcute apoi de dispozitivul de deîntrețesere în blocuri diferite ale codului exterior de dimensiune (28,24). Datorită deîntrețeserii, un bloc de dimensiune 28 bytes al codului interior se transformă în 28 de blocuri de dimensiune 1 byte, unul fiecare din cele 28 de blocuri ale codului exterior, fapt ce permite foarte ușor corecția acestor blocuri. Rezultatul final este acela că un CIRC poate corecta burst-uri de dimensiune de până la 4000 de biți.

O a doua aplicație este în codurile de bare bidimensionale (PDF-417, MaxiCode, Datamatrix, QR Code, Aztec Code). Aceste coduri de bare folosesc coduri RS pentru a permite citirea corectă în cazul în care o porțiune din cod este stricată. Când scanner-ul nu poate recunoaște un simbol, îl tratează ca pe un erasure.

O a treia aplicație este în transmisia de date. În acest caz se folosesc variante specializate de coduri RS, cum ar fi Cauchy-RS și Vandermonde-RS, pentru a rezolva problema transmiterii datelor prin canale cu erasure-uri. Procesul de codare folosește coduri RS de dimensiune (N, K), de unde rezultă N cuvinte de cod a câte N simboluri ce stochează K simboluri de date. Aceste cuvinte de cod sunt mai apoi transmise pe canal. Orice combinație de K cuvinte de cod recepționate este suficientă pentru a reconstrui toate cele N cuvinte de cod. Rata codului este setată la ½ în cazul în care probabilitatea de apariție a unui erasure poate fi modelată și se observă ca fiind mai mică. În concluzie, N este de obicei de forma 2K, ceea ce înseamnă că este nevoie ca măcar jumătate din toate cunvintele de cod transmise trebuie recepționate pentru a putea reconstrui toate cuvintele de cod trimise.

A patra aplicație și poate cea mai interesantă este în comunicațiile satelitare. Codurile RS au fost folosite pentru a coda fotografiile digitale trimise de către satelitul Voyager. În acest caz au fost introduse codurile Reed-Solomon concatenate cu coduri convoluționale, o practică ce s-a răspândit de atunci foarte mult în comunicațiile satelitare. Decodoarele Viterbi tind să cauzeze erori în cazul burst-urilor de durată scurtă. Corecția acesot burst-uri este realizată foarte simplu folosind coduri Reed-Solomon simplificate sau

scurtate. Versiunile moderne ale concatenărilor de tip Reed-Solomon/coduri convoluționale decodate prin intermediul decodoarelor Viterbi au fost folosite în misiunile sateliților Mars Pathfinder, Galileo, Mars Exploration Rover și Cassini, unde funcționează la un nivel mai mic cu 1-1.5 dB decât limita maximă impusă de capacitatea Shannon. Aceste coduri concatenate au început însă să fie înlocuite de coduri turbo mult mai puternice unde datele transmise nu trebuie decodate imediat.

Capitolul 3 - Utilizarea CRC în rețelele de calculatoare

Primele încercări de comunicație între diferite calculatoare au avut la bază conectarea directă a acestora folosind o linie serială, pe care se efectua apoi schimbul de date. Deoarece linia era nesigură, s-au dezvoltat tehnici care să suplinească acest neajuns. Astfel fiecare entitate de date trimisă avea un câmp de control cu informații despre corectitudinea celorlalte câmpuri.

De asemenea, orice comunicație avea stabilite niște reguli care să fie aplicate în cazuri deosebite. Astfel au apărut o serie de protocoale de comunicație serială, care foloseau diferite coduri. Informațiile schimbate între diferite sisteme sunt codificate, pentru a realiza adaptarea statistică a sursei la canalul de comunicație. Canalul de comunicație poate fi afectat sau nu de perturbații.

Un canal neperturbat ar corespunde unei criptări sau compresii a informației, pe când un canal perturbat transmisiei acesteia. Sistemele actuale care înglobează calculatoare și echipamente automate necesită o transmisie cât mai corectă a informației ; pentru aceasta semnalul trimis este prelucrat înainte de a fi emis.

Prelucrarea semnalului se realizează foarte ușor în cazul transmisiei de semnale discrete, prin codificare. Receptorul va decodifica semnalul primit și va încerca să estimeze dacă au apărut erori.

CRC (Control Redundant Ciclic) este o metodă matematică folosită pentru a verifica integritatea datelor. Este o formă de sumă de control, ce se bazează pe teoria polinoamelor de lungime maximă. Chiar dacă metoda CRC este mai sigură decât metoda bazată pe o simplă sumă de control, nu oferă o adevarată securitate criptografică. CRC este o tehnică folosită pentru detecția erorilor de transmisie. Pentru detecția și corectarea erorilor, există un registru special în care se stochează suma de control a datelor transferate. Aceasta se compară cu suma de control calculată și se elimină astfel posibilele erori. În acest caz, tehnica CRC este folosită doar pentru a asigura integritatea datelor la transferurile pe magistrală, nu și pentru a îmbunătăți integritatea datelor stocate pe discurile hard. Din punct de vedere al tipului de cod, codurile CRC sunt tot coduri ciclice, deci utilizează polinoame. Polinomul acceptat ca standard pentru CRC este cel dat exemplu si mai sus, mai exact standardul IEEE 802.

În ceea ce privește calculul acestui CRC, avem două variante. Prima din ele este *varianta software*. Pentru aceasta există mai multe modalități de implementare, însă trebuie ținut cont de mai mulți factori, dintre care cei mai importanți ar fi:

- Ipotezele de la care se pornește calculul
- Dimensiunea polinomului generator
- Dimensiunea și structura mesajului pentru care se calculează CRC-ul
- Timpul în care se generează CRC-ul
- Dimensiunea spațiului de memorie alocat
- Performanțele procesorului de calcul

Metodele de implementare pot fi clasificate, după viteză, în două categorii:

- Algoritmi de viteză redusă
- Algoritmi de mare viteză

În continuare, pentru implementarea propriu-zisă a unui algoritm de calcul CRC, trebuie să implementăm regulile împărțirii în binar, deoarece CRC-ul folosește această aritmetică. Din nefericire nu putem utiliza instrucțiunea de împărțire disponibilă la un calculator deoarece împărțirea CRC nu este identică cu cea normală, un alt motiv fiind și dimensiunea mesajului, care poate ajunge până la câțiva MB iar procesoarele actuale au registre de dimensiune mult mai mică. Astfel, pentru implementare vom avea nevoie de un registru de deplasare, a cărui dimensiune trebuie să fie egală cu gradul polinomului generator și care să conțină biții aferenți mesajului, prelucrarea urmând a se realiza bit cu bit.

O altă metodă de implementare necesită există unui tabel care să conțină biții deplasați ai polinomului CRC. În vederea micșorării timpului de execuție s-a trecut la procesarea în paralel a biților în grupuri de câte 4b, 8b, 16b sau 32b. Prelucrarea pe nibluri, adică pe 4b, este însă de obicei evitată pe calculatoare, din simplul fapt că acestea operează cu octeți, astfel că pentru mărirea vitezei de lucru se folosesc de obicei prelucrări pe octeți sau multipli de octeți.

Pe lângă implementarea software, există și varianta *implementării hardware*, care se realizează prin intermediul unui sistem cu microcontroller. Un astfel de sistem prezintă o serie de avantaje:

- Permite modificarea cu ușurință a metodei de calcul
- Permite o introducere și prelucrare simplă a datelor înainte de a fi supuse calculului
- Permite comunicația cu diferite periferice
- Permite comunicația cu calculatorul prin intermediul porturilor cum ar fi cel serial

Pentru a obține timpi foarte mici de execuție se folosește o implementare hardware ce utilizează bistabili în varianta paralelă.

După cum am menționat mai sus, CRC-ul reprezintă o formă de sumă de control. Suma de control la nivel de bloc reprezintă un mecanism de detecție a erorilor de transmisie. Modul în care lucrează este următorul:

- Mai întâi, datele se împart în blocuri ce se vor însuma, iar suma obținută este trunchiată, inversată și adăugată la sfârșit
- La recepție, blocurile primite (care include și acea sumă la sfârșit), se adună pe măsură ce sunt recepționate
- În cazul în care am obținut o sumă diferită de 0, înseamnă că datele nu au fost transmise corect iar secvența trebuie retransmisă. Corecția erorii nu este posibilă
- Pentru cazul în care folosim metoda CRC, suma de control este calculată prin împărțire aritmetică, secvența de biți fiind împărțită cu un număr special ales
- Împărțirea se face modulo 2, adică folosind operatorul logic SAU-EXCLUSIV (XOR), restul împărțirii reprezentând semnătura care va fi adăugată la sfârșit, după biții utili
- Divizorul se obține folosind algoritmii codurilor Hamming
- La recepție se recalculează restul împărțirii iar dacă acesta nu coincide cu restul primit, atunci secvența a fost transmisă cu erori

Performantele astfel obținute sunt foarte bune, un CRC ce generează un rest de 16 biți putând detecta:

- Toate erorile în burst (rafală) de maxim 16 biți
- Toate numerele impare de biți din eroare
- Aproape orice eroare, indiferent de lungimea ei

Metodele hardware oferă o ușurință mărită de calcul pentru CRC și folosesc registre de deplasare împreună cu porți logice de tip XOR.

Controlul fluxului de date

Indiferent că vorbim de metode software sau hardware, acest control este necesar pentru a se evita erorile de overflow, care apar de obicei când receptorul nu reușește să prelucreze datele când acestea vin cu o viteză prea mare. Alterarea acestor date poate fi detectată și cu ajutorul sumelor de control, care în Internet sunt calculate prin împărțirea

datelor în blocuri de câte 16 biți. Programul care calculează aceste sume tratează aceste blocuri ca pe niște numere întregi. Valorile în binar ale acestor blocuri sunt însemnate, suma de control urmând a fi transmisă împreună cu datele aferente blocurilor. Sumele sunt calculate la emițător și recalculate la receptor iar dacă nu sunt egale, atunci înseamnă că secvența nu s-a transmis corect.

Cum CRC-urile necesită de obicei niște calcule mai complexe, se preferă implementarea hardware a acestora.

În cazul rețelelor de calculatoare, suma de control este un cod de dispersie pe 32 de biți a datelor și reprezintă un număr care rezultă în urma unui calcul matematic ce este efectuat folosind datele din pachetul de la calculatorul sursă. Pachetele reprezintă unitatea de bază a comunicațiilor în rețea. Atunci când pachetul ajunge la destinație, calculul se reface iar dacă rezultatele sunt identice, înseamnă că pachetul a rămas intact, în caz contrar înseamnă că datele au fost alterate pe parcursul transmisiei din cauza zgomotului de pe cablu. Acest lucru trebuie semnalizat calculatorului sursă pentru ca acesta să retransmită datele.

Algoritmul de calcul al CRC

După cum am mai spus, codurile polinomiale se bazează pe tratarea șirurilor de biți ca reprezentări de polinoame, folosind coeficienți de 0 și 1. Pentru calculul CRC se va folosi aritmetica modulo 2, cea bazată pe adunări fără transport și scăderi fără împrumut. Pe lângă acestea vom folosi și un polinom generator G(x) cu MSB-ul și LSB-ul egale cu 1. În continuare dorim să adăugam un număr de biți la sfârșitul unui cadru, descris de un polinom M(x) ales astfel încât M(x) îl divide pe G(x). Algoritmul pentru calculul sumei de control are următorii pași:

1. Notând cu r gradul lui G(x), vom adăuga r biți de 0 la capătul cel mai puțin semnificativ al polinomului M(x), obținând un nou polinom notat M’(x)
2. Se împarte șirul de biți corespunzători lui G(x) într-un șir de biți corespunzător lui M’(x), prin împărțire modulo 2
3. Se scade restul (de dimensiune d≤r biți) din șirul de biți corespunzător lui M’(x) prin scădere modulo 2, rezultatul acestei scăderi fiind cadrul cu suma de control ce va fi transmis și care este divizibil modulo 2 cu G(x).

Pentru a ilustra acest concept, vom lua un exemplu numeric. Fie cadrul 1101011011 și codul pentru generator 10011. Polinomul G(x) va fi $x^4 + x + 1$, deci r=grad(G(x))=4, prin

urmare vom concatena 4 biți de 0 la capătul nesemnificativ al cadrului. Mesajul devine 11010110110000. Împărțirea acestui mesaj la polinomul generator G(x) va produce următoarele rezultate:

- Cadru transmis: 11010110111110
- Rest: 1110

De mentionat, în final, ar fi că un astfel de cod ca cel din exemplu este capabil să detecteze erori în rafală (burst) de dimensiune de până la r biți.

Aplicație

În continuare, vom prezenta o aplicație ce calculează CRC pentru un bloc de date, realizată în limbajul C++.

```
char mxor(char a, char b)
{
    if((a='0')&&(b='0'))
                return '0';
    if((a='1')&&(b='0'))
                return '1';
    if((a='0')&&(b='1'))
                return '1';
    if((a='1')&&(b='1'))
                return '0';
};
char *crc(char *cod) //functia scade succesiv polinomul x^3+x+1 din secventa care trebuie codata
{
     string str="1011";
     char *g=new char[4]; //declararea polinomului x^3+x+1;
     char* remainder = new char[16];
     int i,j;
     for(i=0;i<4;i++)
```

```
            *(g+i)=str[i];
    for(i=0;i<8;i++)
                *(remainder+i)=*(cod+i);
    for(i=8;i<16;i++)
                *(remainder+i)='0';
    for(i=0;i<8;i++)
            if(*(remainder+i)=='1')
            {
                    for(j=0;j<4;j++)
                            *(remainder+i+j)=mxor(*(remainder+i+j),*(g+j));
            };
    return remainder;
};
```

Capitolul 4 – Concluzii

Detecția și corecția erorilor au reprezentat mereu unele din cele mai importante provocări din domeniul transmisiunii informatiei, iar într-o eră în care comunicațiile s-au dezvoltat în atâtea forme, este lesne de înțeles de ce se dorește transmisiunea cât mai fidelă a informației.

Metodele de detecție și corecție sunt variate și se bazează pe o varietate de coduri, fie ele ciclice, grup sau convoluționale. Codurile grup și cele convolutionale folosesc în general o codare binară a informației, tratând simbolurile prin intermediul reprezentării lor cu biți de 0 și 1, în timp ce codurile ciclice asociază câte un polinom fiecărui astfel de cod. Codurile ciclice se folosesc cel mai des în calcularea sumei de control (checksum în engleză), metodă foarte des folosită pentru detecția erorilor care, deși nu poate realiza și corectia, este destul de ușor de implementat și de utilizat.

Codurile Reed-Solomon au reprezentat un pas înainte în acest domeniu. Chiar dacă alfabetul lor și spatiul pe care sunt definite diferă față de codurile ciclice (și grup, pentru că aceste coduri Reed-Solomon pot fi privite și ca coduri grup, existând o echivalență între relațiile lor de reprezentare ca și coduri ciclice, respectiv cele de reprezentare a lor ca și coduri grup, aceasta din urmă reprezentând de fapt varianta clasică în care sunt privite codurile Reed-Solomon), ele și-au găsit implementări în diverse domenii precum comunicațiile satelitare, stocarea datelor, rețele de calculatoare etc.

Metoda CRC (Control Redundant Ciclic sau Cyclic Redundancy Check în engleză) reprezintă o altă metodă extrem de des folosită în domeniul detecției erorilor. Ea se bazează pe calculul unei sume de control și pe compararea aritmetică a valorilor obținute cu niște valori ce ar trebui obținute în cazul unei transmisii corecte a datelor. Dacă aceste valori diferă între ele, atunci spunem că transmisia nu a fost realizată corect și anunțăm emițătorul să retransmită secvența în cauză. Această metodă are un avantaj enorm, reprezentat de posibilitatea detectării erorilor în rafală, adică cele întinse pe mai mulți biți, ceea ce o face foarte utilă în domenii precum comunicațiile, în special cele între calculatoare și cele pe Internet, unde datele circulă sub forma unor pachete și este foarte important să putem verifica erorile pe mai mulți biți mai degrabă decât bit cu bit, stocarea datelor etc.

Aceste două metode prezentate aici sunt doar două exemple dintr-o gamă largă de posibilități în acest domeniu, în care mereu se încearcă găsirea de modalități noi, mai rapide și

mai eficiente de detecție și corectie a erorilor, însă CRC și codurile Reed-Solomon vor rămâne mereu două puncte de reper foarte importante.

Bibliografie

1. http://www.scribd.com/doc/46827525/10/Codul-Reed-Solomon

2. Welch, L. R. (1997), *The Original View of Reed–Solomon Codes*, Lecture Notes

3. http://www.scribd.com/doc/4532428/Retele-de-calculatoare

4.http://etti.poly.ro/cursuri/anul%20IV/cd/Cap%203%20Protectia%20datelor%20impotriva%20erorilor.pdf

5. http://www.scritube.com/stiinta/informatica/Tipuri-de-erori-si-modalitatil31596.php

6. Andrew Tanenbaum - Computer Netwroks 5th edition

7. Radu Radescu - Arhitectura Sistemelor de Calcul

8. Hong, Jonathan; Vetterli, Martin (August 1995), "Simple Algorithms for BCH Decoding", *IEEE Transactions on Communications* **43** (8): 2324–2333

9. MacWilliams, F. J.; Sloane, N. J. A. (1977), *The Theory of Error-Correcting Codes*, New York, NY: North-Holland Publishing Company

10. Berlekamp, Elwyn R. (1984) [1968], *Algebraic Coding Theory*, Laguna Hills, CA: Aegean Park Press