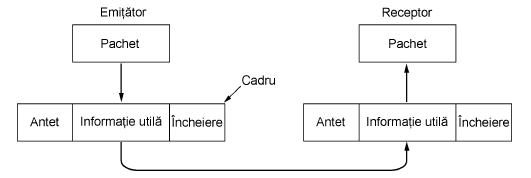
2. Nivelul legatura de date

2.a. Caracteristici ale proiectarii nivelului legaturii de date

Functii:

1. Furnizarea unei **interfete** bine-definite catre nivelul retea
2. **Tratarea erorilor** de transmisie
3. **Gestionare frame**: reglarea fluxului cadrelor in asa fel, incat receptorii lenti sa nu fie inundati de catre emitatori rapizi

Niv. leg. de date primeste pachete de la nivelul retea, pe care le incapsuleaza in **frame-uri** in vederea transmiterii. Fiecare **frame** contine un **antet**, un camp de **informatie utila** pentru pachet si **incheiere**.



# Servicii oferite nivelului retea

* **transferul datelor de la nivelul retea al masinii sursa la nivelul retea al masinii destinatie**. La nivelul retea al masinii sursa exista o entitate, sa-i spunem proces, care trimite biti catre nivelul legatura de date, pentru a fi transmisi la destinatie. Functia nivelului legatura de date este sa transmita bitii spre masina destinatie, pentru ca acolo sa fie livrati nivelului retea.
* Alte servicii:

1. Serviciu neconfirmat fara conexiune
2. Serviciu confirmat fara conexiune
3. Serviciu confirmat orientat-conexiune

# Incadrarea

Nivelul fizic trimite catre nivelul leg. de un flux de biti. Nu se garanteaza ca acest flux de biti nu contine erori. Numarul de biti receptionati poate fi mai mic, egal cu, sau mai mare decat numarul de biti transmisi si pot avea valori diferite. Este la latitudinea nivelului legatura de date sa detecteze si, daca este necesar, sa corecteze erorile.

2.b. Detectarea si corectarea erorilor

1. pe langa fiecare bloc de date trimis sa se includa suficienta informatie redundanta astfel incat receptorul sa poata deduce care a fost caracterul trimis (**coduri corectoare de erori**)
2. sa se includa suficienta redundanta pentru a permite receptorului sa constate ca a aparut o eroare, dar nu care este eroarea, si sa ceara o retransmisie (**coduri detectoare de erori**)

# Coduri corectoare de erori

## Metoda Hamming

Bitii cuvantului de cod ( = biti de date si de control intr-un singur frame) sunt numerotati consecutiv, incepand cu bitul 1 de la marginea din stanga:

* Biti de control: puteri ai lui 2
* Date: ceilalti biti

Un bit de control contribuie la un bit de data daca pozitia bitului de control face parte din descompunerea bitului de date ca o suma.

Pentru fiecare bit de control, paritatea grupului bitilor de date la care contribuie trebuie sa corespunda cu valoarea sa.

Algoritm: cuvantul de cod soseste. Initializam un contor la 0. Examinam fiecare bit de control. Daca nu are paritatea corecta, adaugam pozitia asocita cu el la contor. La final, bitul invalid are pozitia egala cu valoarea contorului daca aceasta este diferit de 0.

Codurile Hamming pot corecta numai erori singulare, insa exista un artificiu care poate fi utilizat pentru a permite codurilor Hamming sa corecteze erorile in rafala.

## Codul polinomial (Cyclic redundancy code “**CRC**”)

Asociem sirului de biti de lungime k un polinom cu coeficienti 0 si 1 de gradul k-1 – bitul cel mai din stanga corespunde puterii k-1. Emitatorul si receptorul se pun de accord asupra unul polinom generator G (care trebuie sa aiba primul si ultimul bit egal cu 1).

**Emitarorul**: asociaza mesajul (sirul de biti) cu un polinom M. Impare M (la care se adauga grad(G) – 1 biti de 0 la sfarsit) la G utilizand XOR si obtine polinomul R.

Ex: M = 1101110 si G = 101.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 1 | 1 | 1 |  | 0 | 0 | 0 |  |
| 1 | 0 | 1 | 1 |  |  |  |  |  |  | XOR |
| 0 | 1 | 1 | 0 | 1 | 1 |  | 0 | 0 | 0 |  |
|  | 1 | 0 | 1 | 1 |  |  |  |  |  | XOR |
| 0 | 0 | 1 | 1 | 0 | 1 |  | 0 | 0 | 0 |  |
|  |  | 1 | 0 | 1 | 1 |  |  |  |  | XOR |
| 0 | 0 | 0 | 1 | 1 | 0 |  | 0 | 0 | 0 |  |
|  |  |  | 1 | 0 | 1 |  | 1 |  |  | XOR |
| 0 | 0 | 0 | 0 | 1 | 1 |  | 1 | 0 | 0 |  |
|  |  |  |  | 1 | 0 |  | 1 | 1 |  | XOR |
| 0 | 0 | 0 | 0 | 0 | 1 |  | 0 | 1 | 0 |  |
|  |  |  |  |  | 1 |  | 0 | 1 | 1 | XOR |
| 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 1 |  |

**Receptorul**: primeste M si R. Folosind algoritmul de mai sus, verifica daca M (la care se adauga R la sfarsit) impartit la G (cu XOR) da restul 0. Daca restul nu este 0 => eroare de transmisie.

2.c. Protocoale elementare pentru legatura de date

Frame-ul contine: kind, seq, ack şi info, dintre care primele trei conţin informaţii de control, iar ultimul poate conţine datele efective care trebuie transferate.

Câmpul kind (tip) spune dacă există sau nu date în cadru. Câmpurile seq şi ack sunt utilizate pentru numere de secvenţă şi, respectiv, confirmări (acknowledgements). Câmpul info al unui cadru de date conţine un singur pachet de date; câmpul info al unui cadru de control nu este utilizat.

# Un protocol simplex fără restricţii

* Datele sunt transmise intr-o singura directie
* Emitatorul si receptorul sunt mereu pe starea ready.
* Timp de prelucrare ignorat.
* Nu se produc erori (nu se pierd frame-uri/frameurile nu au pierderi de biti)
* Memorie de stocare infita

Sender()

{

        forever

        {

                from\_host(buffer); //preia pachet de la niv. retea

                S.info = buffer; //construire frame de iesire; util. doar campul info

                send\_frame(S); //trimitere

        }

}

Receiver()

{

        forever

        {

                wait(event); //asteapta sa se intample ceva

                get\_frame(R);

                to\_host(R.info); //trimite catre niv. retea

        }

}

# Un protocol simplu Stop-and-Wait (pas-cu-pas)

* Datele sunt transmise intr-o singura directie
* Nu se produc erori (nu se pierd frame-uri/frameurile nu au pierderi de biti)
* La receptor, timpul de prelucrare este finit (aici nu se mai ignora)

Aceste asumtii implica faptul ca emitatorul nu poate trimite frame-uri mai repede decat receptorul poate procesa (adica sa nu trimita un nou frame pana cand cel current nu a fost trimis in nivelul retea/frame-ul nu a fost preluat din nivelul fizic ca nu cumva sa se faca overwrite).

Receptorul trimite un acknowledge frame catre emitator pentru a-l anunta ca a terminat de procesat ultimul frame si l-a trimis catre host, asa ca acum poate primi un frame nou. Emitatorul trebuie sa astepte acknowledge frame-ul din partea receptorului pentru a putea trimite un nou frame.

Sender()

{

        forever

        {

                from\_host(buffer);

                S.info = buffer;

                send\_frame(S);

                wait\_ack\_frame(event);

        }

}

Receiver()

{

        forever

        {

                wait\_frame(event);

                get\_frame(R);

                to\_host(R.info);

                send\_frame(S);

        }

}

2.d.1. PROTOCOALE CU FEREASTRĂ GLISANTĂ

* Date transmise in ambele directii
* Se transmit acknowledge frames si normal frames.
* Receptorul inspecteaza campul KIND pentru a vedea daca frame-ul transmis este acknowledge frame sau frame normal.

**Tehnica piggybacking**

Se primeste un frame in niv. leg. de date de la niv. fizic (alt host). Se asteapta pana cand host-ul trimite un frame catre niv. leg. de date. Se trimite un singur frame (cel primit de la host) utilizand campul ACK (care anunta ca niv. leg de date e gata sa primeasca un frame nou) catre destinatie (celalalt host).

**Problema**: nu stim cat va dura pana cand host-ul curent trimite un frame.

**Solutie**: timp de asteptare fixat. Daca este depasit, se trimite un frame ce contine doar ACK.

**Protocol cu fereastra glisanta**

* **Emitator**: mentine sending window (lista cu frame-uri trimise, dar fara confirmare): soseste un frame de la host. Ii atribuim cel mai mare SEQ, il inseram in lista, limita ferestrei creste cu 1. Cand soseste o confirmare, ACK-ul frame-ului primit da pozitia frame-ului care a fost confirmat. Fereastra face slide la pozitia ce urmeaza dupa aceasta .
* **Receptor**: mentine receiving window (lista cu frame-uri care pot fi trimise in sus): soseste un frame de la alt host. Inseram frame-ul in lista. Cel mai din stanga frame este trimis catre host. Se trimite un frame ACK catre emitator. Fereastra face slide catre urmatorul frame care trebuie primit.
* Ambele ferestre sunt ordonate

2.d.2. Protocoale de revenire cu n paşi (Go Back n)

Daca un frame eronat ajunge la receptor este eliminat. Totusi, niv. leg. de date este obligat sa livreze frame-uri catre niv retea in ordine, asa ca apare o eroare.

**Revenire cu n paşi (go back n)**

Daca receptorul detecteaza un frame eronat, semnaleaza emitatorul si elimina toate celelalte frame-uri din receiving window. Emitatorul, care acum poate ca trimite frame-ul N + X cand eroarea este primita, initializeaza retransmiterea frame-ului N si a tuturor frame-urilor ce urmatoare.