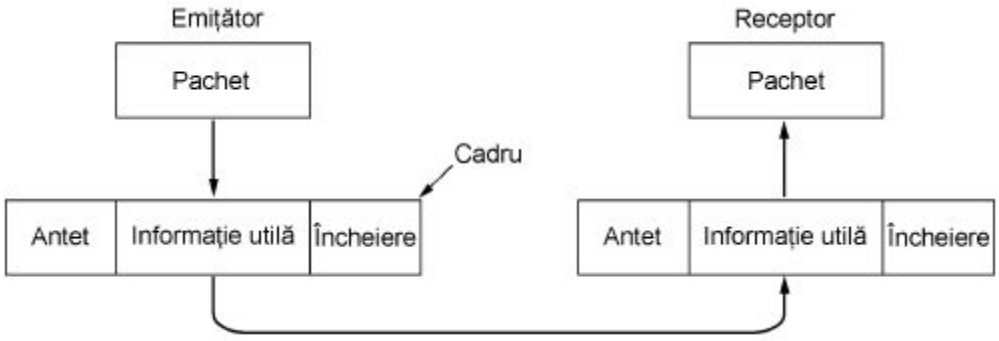
2. Nivelul legatura de date

2.a. Caracteristici ale proiectarii nivelului legaturii de date

Nivelul legatura de date are un numar de functii specifice pe care trebuie sa le indeplineasca. Aceste functii includ:

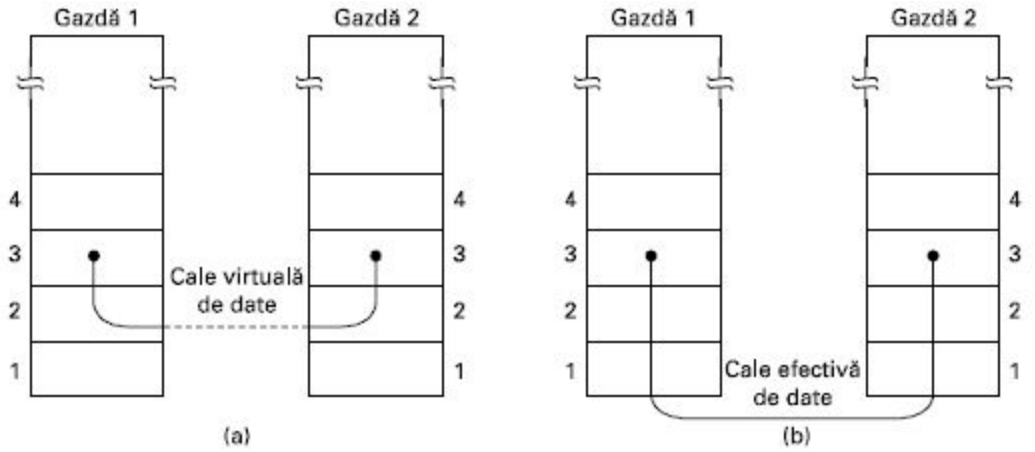
1. Furnizarea unei interfete bine-definite catre nivelul retea
2. Tratarea erorilor de transmisie
3. Reglarea fluxului cadrelor in asa fel, incat receptorii lenti sa nu fie inundati de catre emitatori rapizi

Pentru a indeplini aceste scopuri, nivelul legatura de date primeste pachete de la nivelul retea, pe care le incapsuleaza in **cadre** in vederea transmiterii. Fiecare **cadru** contine un antet, un camp de informatie utila pentru pachet si incheiere, dupa cum se vede in figura de mai jos. Gestionarea cadrelor reprezinta esenta a ceea ce face nivelul legatura de date.



Servicii oferite nivelului retea

Functia nivelului legatura de date este sa ofere servicii nivelului retea. Principalul serviciu este 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. Funcia nivelului legatura de date este sa transmita bitii spre masina destinatie, pentru ca acolo sa fie livrati nivelului retea, asa cum se arata in figura (a) de mai jos. Transmisia efectiva urmeaza calea din figura (b).



Nivelul legatura de date poate fi proiectat sa ofere diferite servicii. Serviciile oferite pot varia de la sistem la sistem. Trei posibilitati de baza, oferite in mod curent, sunt:

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

Incadrarea

In vederea furnizarii unui serviciu nivelului retea, nivelul legatura de date trebuie sa utilizeze serviciul furnizat de catre nivelul fizic. Sarcina nivelului fizic este sa primeasca un flux de biti si sa incerce sa-l trimita la destinatie. 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

Exista 2 strategii de baza pentru tratarea erorilor. O modalitate este ca pe langa fiecare bloc de date trimis sa se includa suficienta informatie redundanta astfel incat receptorul sa poata deduce care a fost caracterul trimis. O alta solutie este 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. Prima strategie utilizeaza **coduri corectoare de erori**, iar cea de-a doua utilizeaza **coduri detectoare de erori**.

Coduri corectoare de erori

In cazul canalelor de comunicatie fara fir, este indicat sa adaugam destula informatie redundanta fiecarui bloc, in loc sa ne bazam pe retransmisie, care poate sa fie la randul sau afectata de erori.

In mod normal, un cadru contine *m* biti de date si *r* biti redundanti sau de control. Sa consideram lungimea totala *n* (adica *m+r*). O unitate formata din n biti, care contine date si biti de control, este numita frecvent **cuvand de cod** de n biti.

Date fiind doua cuvinte de cod, este posibil sa determinam cati biti corespunzatori difera. Pentru a determina cati biti difera, aplicam operatorul SAU EXCLUSIV intre cele doua cuvinte de cod si numaram bitii 1 din rezultat.

Exemplu:
10001001
10110001
00111000

Numarul de pozitii binare in care doua cuvinte de cod difera se numeste **distanta Hamming**. Daca doua cuvinte de cod sunt despartite de o distanta Hamming *d*, sunt necesare *d* erori de un singur bit pentru a-l converti pe unul in celalalt.

Proprietatile detectoare si corectoare de erori ale unui cod depinde de distanta sa Hamming. Pentru a **detecta** *d* erori, este nevoie de un cod cu distanta *d+1*. Atunci cand receptorul vede un cuvant de cod incorect, poate spune ca s-a produs o eroare de transmisie. Pentru a **corecta** *d* erori, este nevoie de un cod cu distanta *2d+1*.

Metoda Hamming

Bitii cuvantului de cod sunt numerotati consecutiv, incepand cu bitul 1 de la marginea din stanga. Bitii care sunt puteri ale lui 2 (1,2,4,8,16,etc) sunt biti de control. Restul sunt completati cu cei m biti de date. Fiecare bit de control forteaza ca paritatea unui grup de biti, inclusiv el insusi, sa fie para (sau impara).

Un bit poate fi inclus in mai multe calcule de paritate. Pentru a vedea la care biti de control contribuie bitul de date din pozitia k, rescriem k ca o suma de puteri ale lui 2. De exemplu, 11=1+2+8 si 29=1+4+8+16. Un bit este verificat de acei biti de control care apar in dezvoltarea sa.

Cand soseste un cuvand de cod, receptorul initializeaza un contor la 0, Acesta examineaza apoi fiecare bit *k* (*k*=1,2,4,8,...) de control pentru a vedea daca are paritatea corecta. Daca nu, adauga *k* la contor. Daca, dupa ce au fost examinati toti bitii de control, contorul este 0, cuvantul de cod este acceptat ca valid. Daca valoarea este nenula, ea reprezinta numarul bitului incorect. De exemplu, daca bitii de control 1,2 si 8 sunt eronati, atunci bitul inversat este 11, deoarece este singurul verificat de bitii 1,2 si 8.

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.

Coduri detectoare de erori

Uneori este mai eficient sa utilizam un cod detector de erori si sa retransmitem blocul in care s-au detectat erori. De exemplu pe canalele cu siguranta mare, cum ar fi fibra optica.

Codul polinomial (Cyclic redundancy code "**CRC**")

Codurile polinomiale sunt bazate pe tratarea șirurilor de biți ca reprezentări de polinoame cu coeficienți 0 și 1. Un cadru de k biți este văzut ca o listă de coeficienți pentru un polinom cu k termeni, de la x^{k-1} la x^0 . Se spune că un astfel de polinom este de gradul k-1. Bitul cel mai semnificativ (cel mai din stânga) este coeficientul lui x^{k-1} ; următorul bit este coeficientul lui x^{k-2} ș.a.m.d. De exemplu, 110001 are șase biți și ei reprezintă un polinom cu șase termeni cu coeficienții 1, 1, 0, 0, 0 și 1: $x^5 + x^4 + x^0$. Aritmetica polinomială este

de tip modulo 2, în conformitate cu regulile teoriei algebrice. Nu există transport la adunare și nici împrumut la scădere. Atât adunările cât și scăderile sunt identice cu SAU EXCLUSIV.

Atunci când este utilizată metoda codului polinomial, emițătorul și receptorul se pun de acord în avans asupra unui polinom generator G(x). Atât bitul cel mai semnificativ cât și cel mai puțin semnificativ trebuie să fie 1. Pentru a calcula suma de control pentru un cadru cu m biți, corespunzător polinomului M(x), cadrul trebuie să fie mai lung decât polinomul generator. Ideea este de a adăuga o sumă de control la sfârșitul cadrului, astfel încât polinomul reprezentat de cadrul cu sumă de control să fie divizibil prin G(x). Când receptorul preia cadrul cu suma de control, încearcă să-l împartă la G(x). Dacă se obține un rest, înseamnă că a avut loc o eroare de transmisie.

2.c. Protocoale elementare pentru legatura de date

Un cadru de date este compus din patru câmpuri: kind, seq, ack și info, dintre care primele trei conțin informații de control, iar ultimul poate conține datele efective care trebuie transferate. Ansamblul acestor câmpuri de control este numit antetul cadrului **(frame header)**.

Câmpul kind (tip) spune dacă există sau nu date în cadru, deoarece unele protocoale fac distincție între cadrele care conțin exclusiv informații de control și cele care conțin și date. Câmpurile seq și ack sunt utilizate pentru numere de secvență și, respectiv, confirmări (acknowledgements); utilizarea lor va fi descrisă în detaliu mai târziu. Câmpul info al unui cadru de date conține un singur pachet de date; câmpul info al unui cadru de control nu este utilizat.

Un protocol simplex fără restricții

Într-un protocol foarte simplu, datele sunt transmise într-o singură direcție. Cele două niveluri rețea, de transmisie și de recepție, sunt considerate tot timpul pre-gătite. Timpul de prelucrare poate fi ignorat. Memoria de stocare disponibilă este infinită. Și, cel mai bun lucru dintre toate, canalul de comunicație între niveluri legătură de date nu pierde și nu alterează niciodată cadrele. Acest protocol poate fi considerat total nerealist.

Protocolul constă din două proceduri distincte, una de emisie și cealaltă de recepție. Emițătorul lucrează la nivelul legătură de date al mașinii sursă, iar receptorul la nivelul legătură de date al mașinii de destinație.

Emițătorul este într-un ciclu infinit care doar inserează datele pe linie cât poate de repede. Ciclul constă din trei acțiuni: preluarea unui pachet de date de la nivelul rețea (care este întotdeauna serviabil), construirea unui cadru de ieșire și trimiterea cadrului pe drumul său. Acest protocol utilizează numai câmpul info al cadrului, deoarece celelalte câmpuri se referă la erori și secvențe de control, iar în acest caz nu există erori sau restricții de control.

Receptorul este la fel de simplu. Inițial el așteaptă să se întâmple ceva, singura posibilitate fiind sosirea unui cadru nealterat. În cele din urmă, cadrul ajunge, iar procedura de tip wait_for_event se întoarce cu event setat la frame_arrival (care este oricum ignorat). Apelul from_physical_layer mută cadrul nou sosit din zona tampon a echipamentului într-o anumită variabilă. În cele din urmă pachetul de date este trimis nivelului rețea și

nivelul legătură de date revine la starea de așteptare a cadrului următor, autosuspendându-se pur și simplu până la sosirea unui nou cadru.

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

În acest caz se renunță la cea mai nerealistă restricție utilizată în protocolul anterior (protocolul 1): posibilitatea ca nivelul rețea receptor să prelucreze datele de intrare cu viteză infinită (sau echivalent, prezența în nivelul legăturii de date receptor a unui tampon infinit în care să fie memorate, cât timp își așteaptă rândul, toate cadrele sosite). Totuși, se presupune în continuare că nu se produc erori pe canalul de comunicație și că traficul de date este încă simplex.

Principala problemă care trebuie rezolvată aici este cum să se evite ca emițătorul să inunde receptorul cu date care sosesc mai rapid decât poate acesta să prelucreze. În esență, dacă receptorul are nevoie de un timp Δt ca să execute from_physical_layer și to_network_layer, atunci emițătorul trebuie să transmită la o viteză medie mai mică de un cadru la fiecare interval de timp de Δt. Mai mult, dacă se presupune că echipamentul receptor nu realizează automat memorarea în zona tampon și gestiunea cozii de așteptare, atunci emițătorul nu trebuie să transmită niciodată un nou cadru până când cel vechi nu a fost preluat de rutina from_physical_layer, ca nu cumva cel nou să se scrie peste cel vechi.

O soluție mult mai generală a acestei dileme este ca receptorul să furnizeze o reacție către emițător. După trimiterea unui pachet către nivelul său rețea, receptorul trimite un mic cadru fictiv către emițător care, de fapt, îi dă emițătorului permisiunea să transmită următorul cadru. După ce a transmis un cadru, emițătorul este obligat de protocol să intre în așteptare un timp, până când sosește micul cadru fictiv (deci confirmarea). Utilizarea reacției de la receptor pentru a anunța emițătorul că poate trimite date este un exemplu de control al fluxului menționat anterior.

2.D.1. PROTOCOALE CU FEREASTRĂ GLISANTĂ

În protocoalele anterioare, cadrele cu date erau transmise într-o singură direcție. În cele mai multe situații practice, este necesar să se transmită date în ambele direcții. În acest model, cadrele cu date de la A la B sunt amestecate cu cadrele de confirmare de la A la B. Uitându-se la câmpul kind din antetul cadrului ce a sosit, receptorul poate spune dacă este vorba de un cadru de date sau de confirmare.

Cu toate că întrepătrunderea cadrelor de date și control pe același circuit constituie o îmbunătă- ţire faţă de cazul utilizării a două circuite fizice separate, mai este posibilă încă o îmbunătățire. Atunci când sosește un cadru cu date, în locul emiterii imediate a unui cadru de control separat, receptorul stă și așteaptă până când nivelul rețea îi dă următorul pachet. Confirmarea este atașată cadrului cu date de ieșire (utilizând câmpul ack din antetul cadrului). De fapt confirmarea este transportată pe gratis de către următorul cadru cu date de ieșire. Tehnica întârzierii confirmării, astfel încât să poată fi agățată de următorul cadru de date, este cunoscută ca **ataşare (piggybacking)**.

Următoarele protocoale sunt protocoale bidirecționale care aparțin unei clase de protocoale numite protocoale cu **fereastră glisantă (sliding window)**.

Esența protocoalelor cu fereastră glisantă este aceea că, la orice moment de timp, emițătorul menține o mulțime de numere de secvență care corespund cadrelor pe care are permisiunea să le trimită. Se spune că aceste cadre aparțin **ferestrei de transmisie (sending window)**. Similar, receptorul menține de asemenea o **fereastră de recepție (receiving window)**, ce corespunde mulțimii de cadre care pot fi acceptate. Fereastra emițătorului și fereastra receptorului nu trebuie să aibă aceleași limite minime și maxime și nici măcar aceeași dimensiune. În unele protocoale ele au dimensiune fixă, dar în altele ele pot crește sau scădea pe măsură ce cadrele sunt emise sau recepționate.

Numerele de secvență din cadrul ferestrei emițătorului reprezintă cadre transmise sau cadre ce pot fi transmise, dar încă neconfirmate. De fiecare dată când de la nivelul rețea sosește un nou pachet, acestuia îi este atribuit următorul număr de secvență, iar marginea superioară a ferestrei este avansată cu unu. Atunci când sosește o confirmare, crește cu unu limita inferioară a ferestrei. În acest mod, fereastra menține continuu o listă de cadre neconfirmate. Un exemplu este prezentat în figura de mai jos.

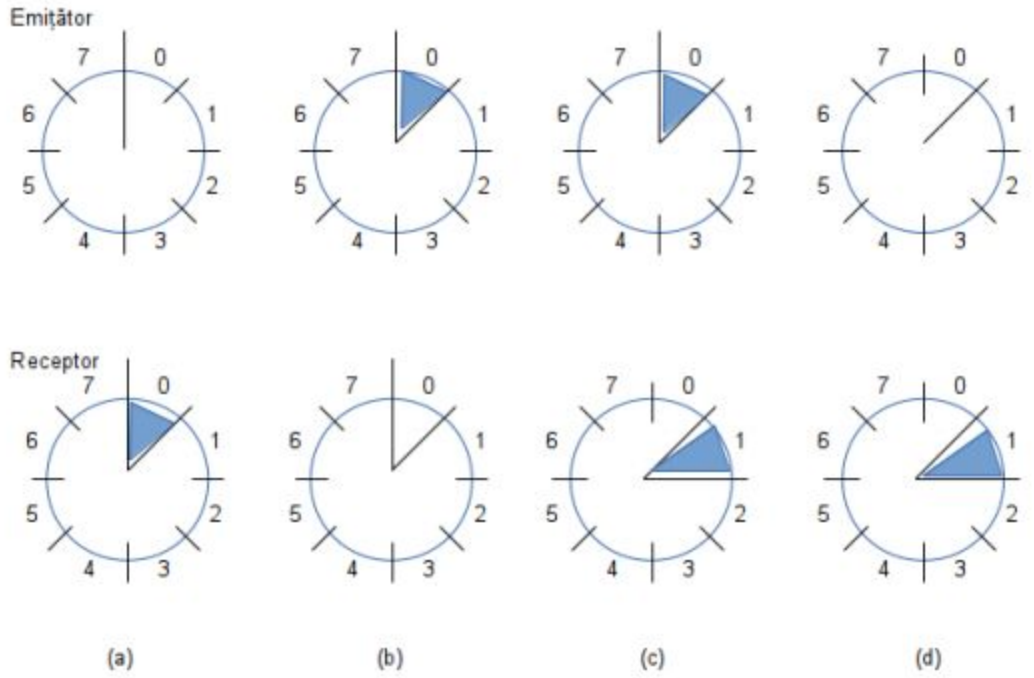


Figura 4-9 – O fereastră glisantă de dimensiune 1, cu număr de secvenţă de 3 biţi.

(a) Iniţial. (b) După ce a fost transmis primul cadru. (c) După ce a fost recepţionat primul cadru.

(d) După ce a fost recepţionată prima confirmare.

2.D.2. Protocoale de revenire cu n pași (Go Back n)

Utilizarea benzii de asamblare în cazul unui canal de comunicație nesigur ridică probleme serioase. Mai întâi să vedem ce se întâmplă dacă un cadru din mijlocul unui șir lung este modificat sau pierdut. Multe cadre succesive vor ajunge la receptor înainte ca emițătorul să observe că ceva este greșit. Atunci când un cadru

modificat ajunge la receptor este evident că el trebuie eliminat, dar ce trebuie să facă receptorul cu toate cadrele corecte care urmează? Nivelul legătură de date receptor este obligat să livreze pachete către nivelul rețea în secvență. În Figura 4-10, se prezintă efectele utilizării benzii de asamblare asupra revenirii în caz de eroare.

Există două moduri de bază de tratare a erorilor în prezența benzii de asamblare. Un mod, numit **revenire cu n pași (go back n)**, este ca receptorul să elimine pur și simplu cadrele care urmează, netrimițând confirmări pentru cadrele eliminate. Această strategie corespunde unei ferestre de recepție de dimensiune 1. Cu alte cuvinte, nivelul legătură de date refuză să accepte orice cadru exceptându-l pe următorul care trebuie livrat către nivelul rețea. Dacă fereastra emițătorului se umple înaintea expirării contorului de timp, banda de asamblare va începe să se golească. În cele din urmă, timpul emițătorului va expira și se vor retransmite toate cadrele neconfirmate, în ordine, începând cu cadrul pierdut sau modificat. Dacă rata erorilor este mare, această abordare poate risipi o mare parte din lărgimea de bandă.

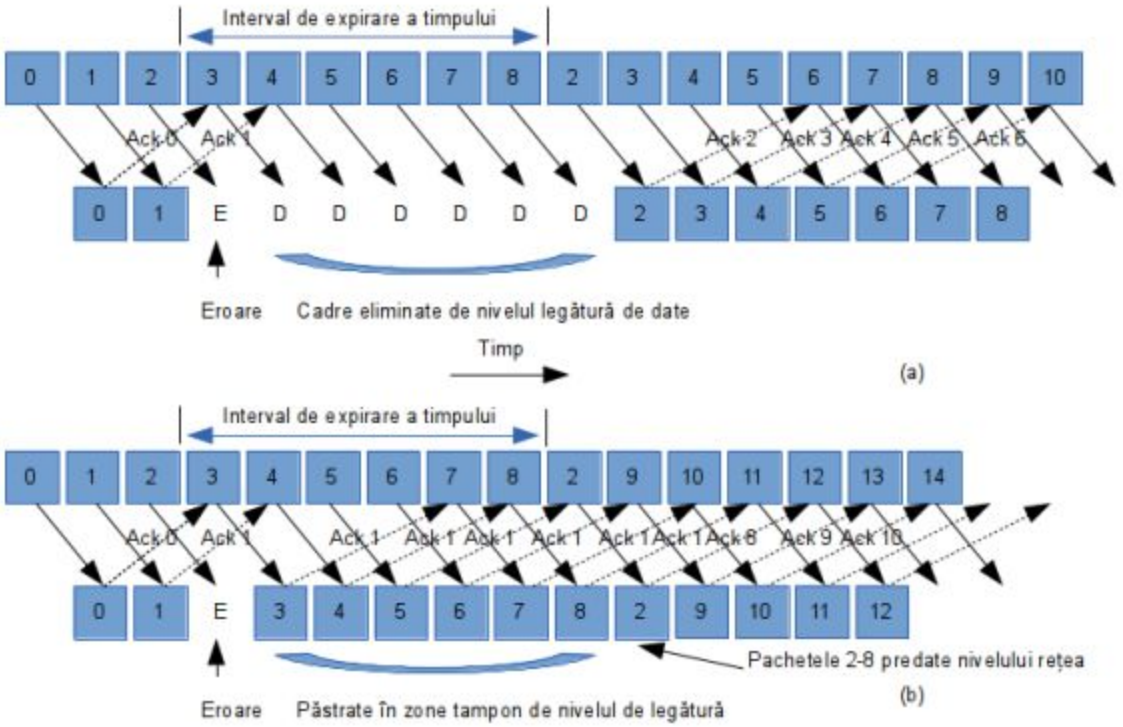


Figura 4-10 – Folosirea benzii de asamblare şi revenirea din eroare. Efectul unei erori când (a) dimensiunea ferestrei receptoare este 1şi (b) dimensiunea ferestrei receptorului este mare.

În Figura 4-10 (a) este prezentat protocolul de revenire cu n pași pentru cazul în care fereastra receptorului are dimensiune unu. Cadrele 0 și 1 sunt primite și confirmate corect. Cadrul 2, totuși, este alterat sau pierdut. Emițătorul, care nu știe de această problemă, continuă să trimită cadre până când timpul pentru cadrul 2 expiră. Apoi se întoarce la cadrul 2 și o ia de la început cu el, trimițând din nou cadrele 2, 3, 4 etc.