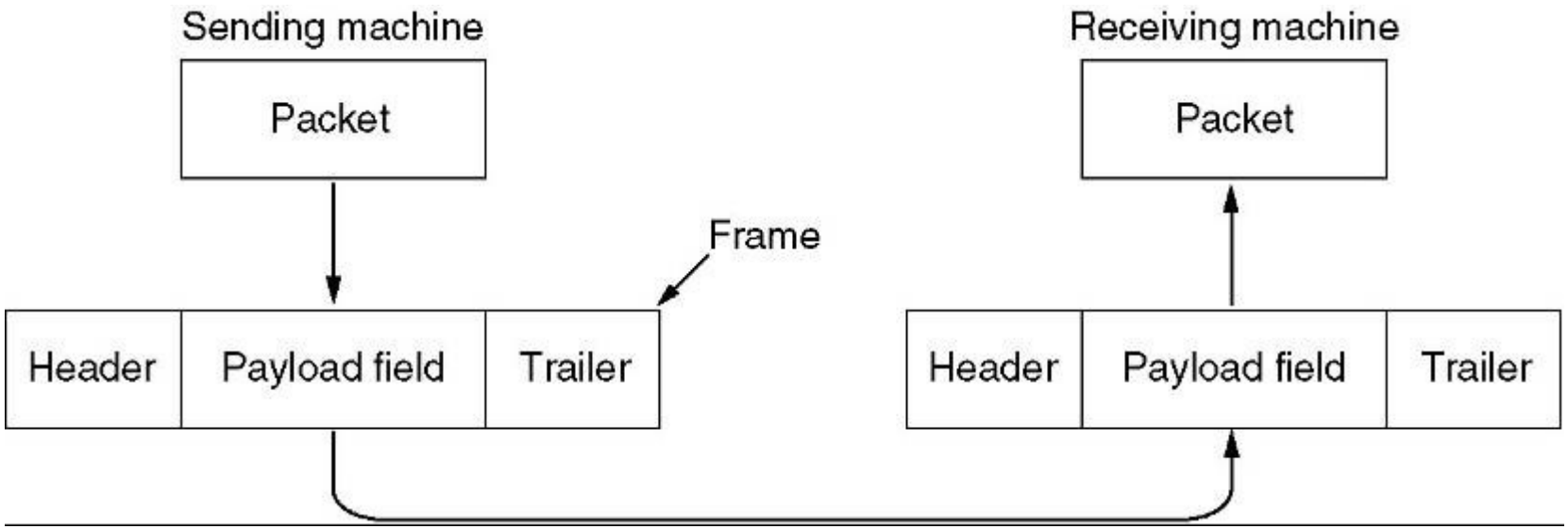
Nivelul Legăturii de Date

Funcţiile nivelului legăturii de date

1 – Încadrarea datelor

2 – Transmisia transparentă



3 – Controlul erorilor

- Coduri detectoare și corectoare de erori
  - secvenţa de control a cadrului - FCS - frame checking sequence
- Mecanisme de protocol
  - mesaje de confirmare
  - ceasuri
  - numere de secvenţă

4 – Controlul fluxului – protocol

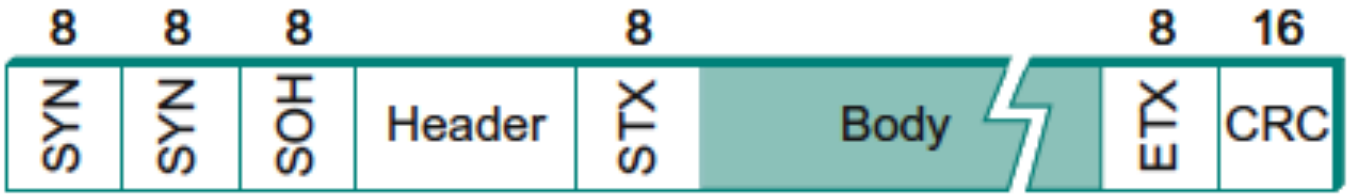
- mesaje de permisiune pentru transmiţător

5 – Gestiunea legăturii – protocol

- stabilirea şi desfiinţarea legăturii
- re-iniţializare după erori
- configurarea legăturii (staţii primare şi secundare, legături multipunct etc.)

Metode de încadrare

1) Caractere de control (BSC - Binary Synchronous Communication)



SOH - start of heading

STX - start of text

ETX - end of text

ETB - end of transmission block

EOT - end of transmission

ENQ - enquiry

ACK - acknowledge

NAK - not acknowledge

SYN - synchronous idle

DLE - data link escape

CRC - cyclic redundancy check

Metode de încadrare

2) Numărarea caracterelor (DDCMP - Digital Data Communications Message Protocol)

SYN SYN SOH count flag resp seq address CRC data CRC

3) Indicatori de încadrare (HDLC - High Level Data Link Control)

flag address command data FCS flag

2 – Transmisie transparentă

STX text ETX — **text alfanumeric: OK!**

STX text… ETX …text ETX — **text binar: ETX fals ?**

Soluție: umplerea cu caractere

Dublează caracterele de control cu DLE

Definește combinații admise

DLE STX – start text transparent

DLE ETX – sfârșit text transparent

DLE STX text… ETX …text DLE ETX CRC

Dublează DLE la transmitere și elimină la recepție

DLE STX ... DLE DLE ... DLE ETX

Eroare: recepție DLE x

cu x diferit de STX, ETX, DLE

Umplere cu biţi

Datele de transmis includ un flag fals de terminare a cadrului

011011111101111111111010

01111110 011011111101111111111010 01111110

Soluția: adăugarea unui zero după 5 unități (în interiorul cadrului!)

01111110 011011111010111110111110010 01111110

Adăugarea se face indiferent dacă după 5 unități urmează 0 sau 1

Simplifică regula receptorului: elimină zeroul aflat după 5 unități

011011111010111110111110010

011011111101111111111010

Detecţia şi corectarea erorilor

Noțiuni preliminare

A = {0, 1} alfabet binar

W_n mulțimea cuvintelor **w** de lungime n peste A

w = w[0] w[1] ... w[n-1] , cu w[i] ϵ A.

ponderea Hamming a lui **w** = numărul de unități conținute de **w**

distanţa Hamming, d(u,v) dintre **u** și **v** este:

ponderea vectorului suma modulo 2 **u+v**

Detecţia şi corectarea erorilor

Ideea - utilizarea unei submulțimi a lui W_n pentru mesaje corecte

Condiția – erorile să producă cuvinte din restul mulțimii W_n

Mulțimea cuvintelor W_n de lungime n se împart în 2 categorii:

S_n este mulțimea cuvintelor cu sens

F_n este mulțimea cuvintelor fără sens

Pentru detecția a cel mult r erori se alege S_n astfel ca

$d(u,v) >= r+1$ pentru orice u, v din S_n

Pentru corecția a cel mult r erori se alege S_n astfel ca

$d(u,v) >= 2r+1$ pentru orice u, v din S_n

Exemplu

S_{10} = {0000000000, 0000011111, 1111100000, 111111111}

d(u,v) = 5 => putem corecta erori duble (max 2 biți eronați)

Astfel,

0000000111 se corectează la 0000011111

Nu se pot corecta 3 erori:

0000000111 poate proveni din 0000000000 in cazul a 3 erori

Metoda Hamming

Biți numerotați de la 1 (stânga) la n (dreapta)

Codificare: Biții 1, 2, 4, 8, ... (puteri ale lui 2) sunt de control

Control paritate (pară sau impară)

Bitul k este controlat de biții ale căror poziții însumate dau k;

Bit 1	controlat de biții	1
Bit 2	controlat de biții	2
Bit 3	controlat de biții	1, 2
Bit 4	controlat de biții	4
Bit 5	controlat de biții	1, 4
Bit 6	controlat de biții	2, 4
Bit 7	controlat de biții	1, 2, 4
Bit 8	controlat de biții	8
Bit 9	controlat de biții	1, 8
Bit 10	controlat de biții	2, 8
Bit 11	controlat de biții	1, 2, 8

Metoda Hamming (2)

Invers:

Bit 1 controlează biții 1, 3, 5, 7, 9, 11

Bit 2 controlează biții 2, 3, 6, 7, 10, 11

Bit 4 controlează biții 4, 5, 6, 7

Bit 8 controlează biții 8, 9, 10, 11

Această formă folosește la calculul biților de control

Exemplu (paritate pară) pentru **1100001**

1	2	3	4	5	6	7	8	9	10	11
?	?	1	?	1	0	0	?	0	0	1
1	?	1	?	1	0	0	?	0	0	1
1	0	1	?	1	0	0	?	0	0	1
1	0	1	1	1	0	0	?	0	0	1
1	0	1	1	1	0	0	1	0	0	1

Prima formă folosește la corectarea erorilor

Ex.1 În loc de:

		1	2	3	4	5	6	7	8	9	10	11
1100001	=>	1	0	1	1	1	0	0	1	0	0	1
Se primește eronat		1	0	1	1	1	0	0	1	0	1	1

Sume eronate controlate de 2 (suma pozițiilor 2,3,6,7,10,11 nu este 0)

și de 8 (8,9,10,11)

8 + 2 = 10 => bitul din poziția 10 este singurul controlat de biții 8 și 2 ➔ bit 10 este inversat

Bit 1 controlat de biții 1
Bit 2 controlat de biții 2
Bit 3 controlat de biții 1, 2
Bit 4 controlat de biții 4
Bit 5 controlat de biții 1, 4
Bit 6 controlat de biții 2, 4
Bit 7 controlat de biții 1, 2, 4
Bit 8 controlat de biții 8
Bit 9 controlat de biții 1, 8
Bit 10 controlat de biții 2, 8
Bit 11 controlat de biții 1, 2, 8

Ex. 2 În loc de:

		1	2	3	4	5	6	7	8	9	10	11
1100001	=>	1	0	1	1	1	0	0	1	0	0	1
Se primește eronat		1	1	1	1	1	0	0	1	0	0	1

Suma eronată controlată de 2 (suma pozițiilor 2,3,6,7,10,11) nu este 0

1	1	1	1	1	0	0	1	0	0	1

celelalte sume sunt corecte ➔ bit 2 este inversat

Obs.

Codul Hamming corectează erorile de 1 bit

Corecția erorilor în rafală

Utilizarea unui cod Hamming pentru corecția erorilor în rafală

- matricea de biți este transmisă coloană cu coloană
- poate corecta erori în rafală dintr-o coloană dacă există

  un bit eronat pe fiecare linie

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

Coduri detectoare de erori

Coduri polinomiale

k biți de informație (date) — **i(X)** polinomul corespunzător

Ex. k=5 10110 — $i(X) = 1*X^4+0*X^3+1*X^2+1*X^1+0*X^0$

n-k biți de control — **r(X)**

n biți în total — $\mathbf{X^{(n-k)}i(X) + r(X)}$

r(X) se alege astfel ca să fie multiplu de **g(X)** — $\mathbf{w(X) = X^{(n-k)}i(X) + r(X)}$, $\mathbf{w(X) = g(X).q(X)}$

$$X^{(n-k)}i(X) + r(X) = g(X).q(X)$$

$$X^{(n-k)}i(X) = g(X).q(X) + r(X)$$

r(X) este restul împărțirii lui $\mathbf{X^{(n-k)}}$ **i(X)** la **g(X)**

Coduri detectoare de erori

Frame : 1 1 0 1 0 1 1 0 1 1

Generator: 1 0 0 1 1

Message after 4 zero bits are appended: 1 1 0 1 0 1 1 0 1 1 0 0 0 0

Calculul sumei de control pentru un cod polinomial

10 biți informație + 4 biți control

Împarte **11010110110000**

la 10011

$X^{(n-k)}$ i(X)

g(X)

```
                1 1 0 0 0 1 0 1 0
          ------------------------
10011   | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
          1 0 0 1 1
          -----------
            1 0 0 1 1
            1 0 0 1 1
            ---------
              0 0 0 0 1
              0 0 0 0 0
              ---------
                0 0 0 1 0
                0 0 0 0 0
                ---------
                  0 0 1 0 1
                  0 0 0 0 0
                  ---------
                    0 1 0 1 1
                    0 0 0 0 0
                    ---------
                      1 0 1 1 0
                      1 0 0 1 1
                      ---------
                        0 1 0 1 0
                        0 0 0 0 0
                        ---------
                          1 0 1 0 0
                          1 0 0 1 1
                          ---------
                            0 1 1 1 0
                            0 0 0 0 0
                            ---------
                              1 1 1 0   <- Remainder
```

r(X) = rest împărţire X(n-k) i(X) la g(X)

Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Ce erori pot fi detectate?

- Probabilitatea de detecție depinde de lungimea codului de control
- CRC și sume de control pe
  - 8 biți detectează 99.6094% din erori

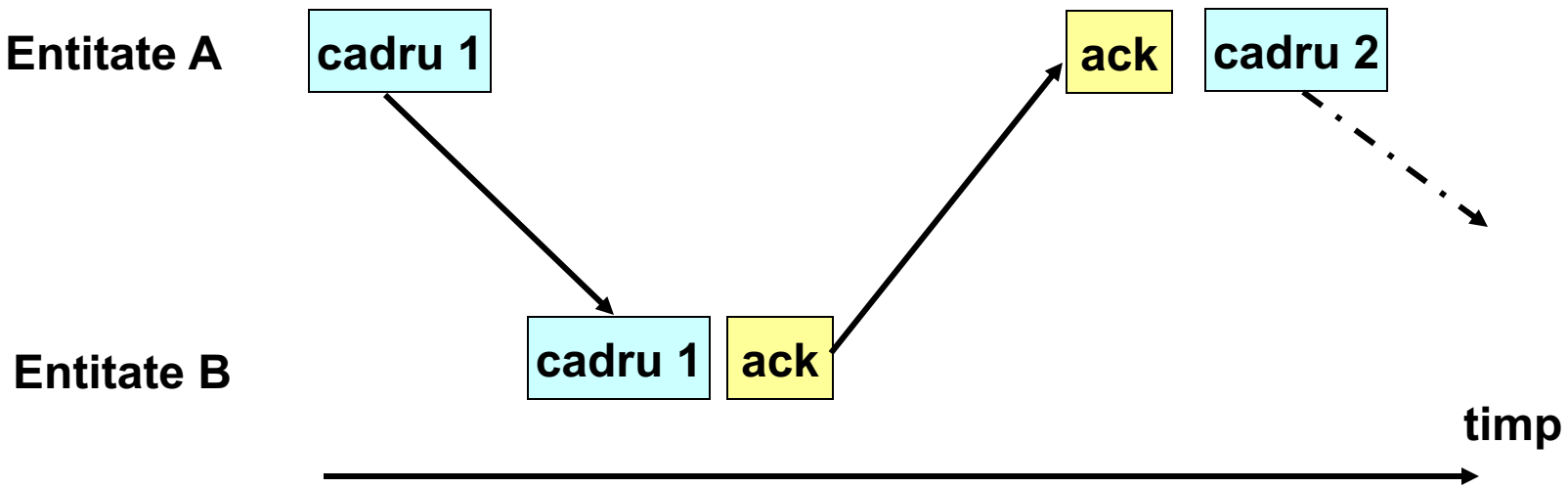
    $x^8 + x^7 + x^5 + x^2 + x + 1$ -> Bluetooth
  - 16 biți detectează 99.9985% din erori

    $x^{16} + x^{15} + x^2 + 1$ -> USB
  - 32 biți detectează 99.9999% din erori

    $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ -> Ethernet
- În plus, CRC detectează 100% erori de
  - 1 bit;
  - 2 biți;
  - un număr impar de biți;
  - erori în rafală de lungimea codului CRC

Protocoale elementare pentru legătura de date

Start-stop

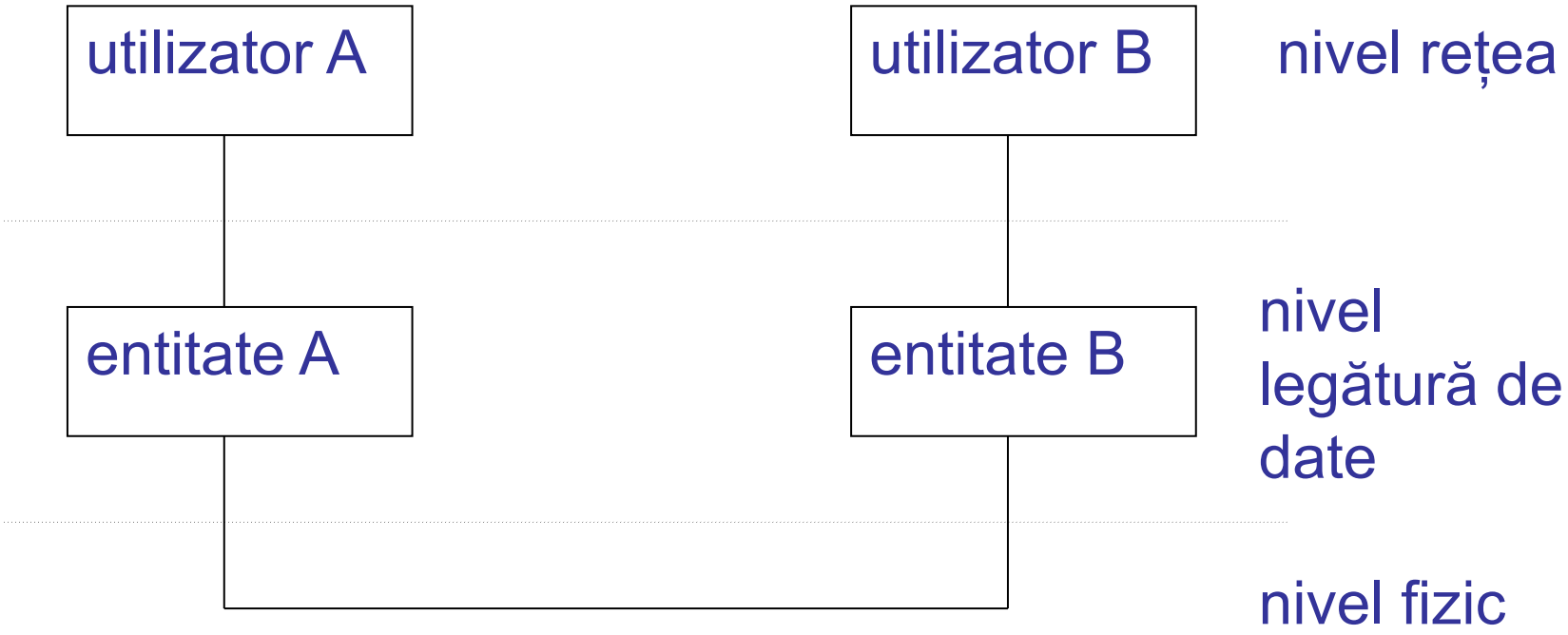


Specificaţie Protocol

- scop și funcții
- servicii oferite
- servicii utilizate din nivel inferior
- structura internă (entități și relații)
- tipuri și formate mesaje schimbate între entități
- reguli de reacție a fiecărei entități la comenzi, mesaje și evenimente interne

Protocoalele legăturii de date

Configuraţia entităţilor de protocol



Datele

```
typedef unsigned char byte;
typedef unsigned int word;
typedef byte NrSecv;

enum FelCadru {data, ack, nak};

typedef struct {
    FelCadru fel;
    NrSecv secv, conf;
    pachet info;
} cadru;

typedef struct {void *adresa;
                word lungime;
} pachet;
```

Primitivele de serviciu

La interfața cu nivelul superior (rețea):

- preluarea unui pachet de la rețea pentru transmitere pe canal

```
pachet DeLaRetea();
```

- livrarea către rețea a unui pachet

```
void LaRetea (pachet);
```

La interfața cu nivelul inferior:

- trecerea unui cadru nivelului fizic pentru transmisie

```
void LaFizic (cadru);
```

- preluarea unui cadru de la nivelul fizic

```
cadru DeLaFizic();
```

Tratarea evenimentelor

```
enum TipEven { SosireCadru,
               EroareControl,
               TimeOut,
               ReteaPregatita};

TipEven wait();
```

Protocoale start-stop

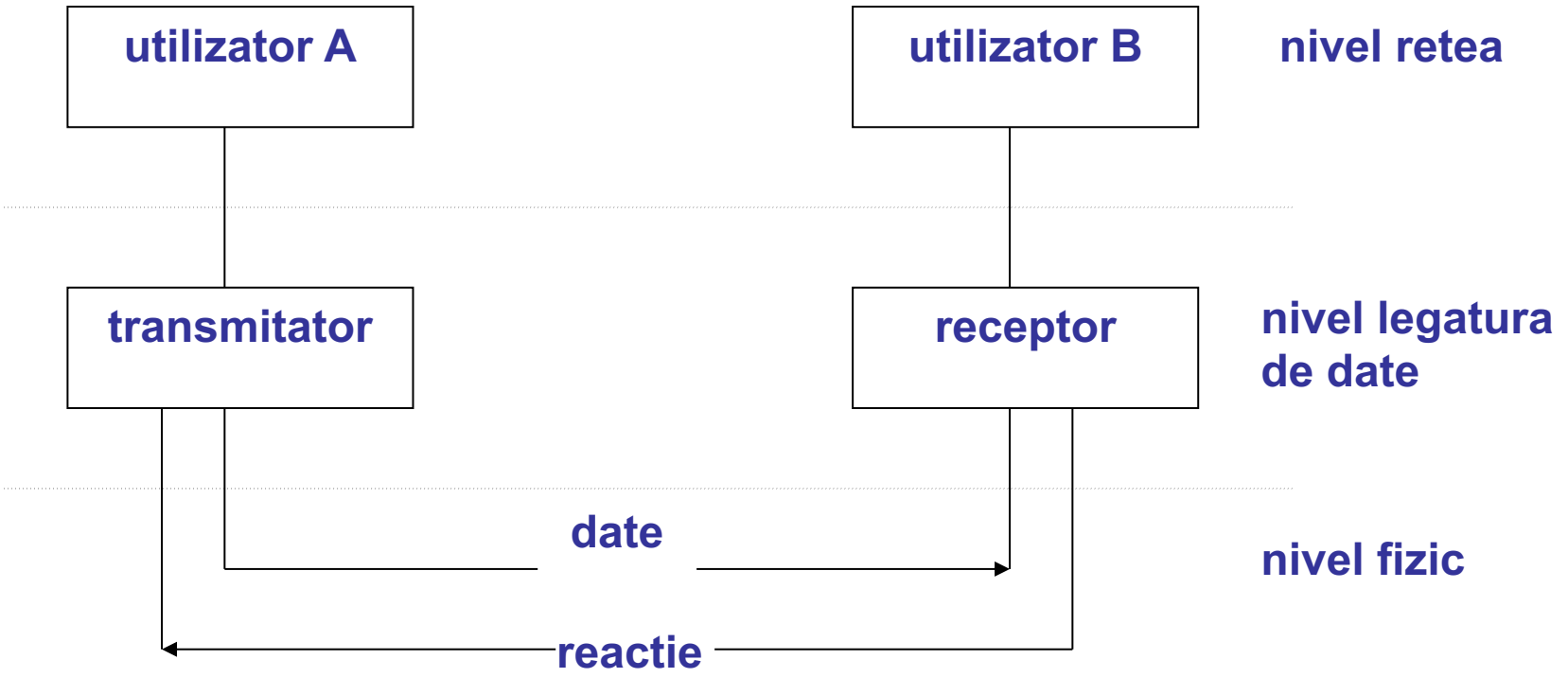
Protocol simplex fără restricții

- utilizatorul A vrea să transmită date lui B folosind o legătură sigură, simplex;
- A reprezintă o sursă inepuizabilă de date;
- B reprezintă un consumator ideal;
- canalul fizic de comunicație este fără erori.

```
# define forever while(1)
// entitatea din sistemul transmitatorului
void transmit-simplex(){
    cadru s;
    do{
        s.info = DeLaRetea();   //preia pachet
        LaFizic(s);             //transmite cadru
     } forever;
}

// entitatea din sistemul receptorului
void recept-simplex(){
    cadru r;
    TipEven even;
    do{
         even = wait();         //așteaptă cadru
         r = DeLaFizic();       //primește cadru
         LaRetea(r.info);       //predă pachet
     } forever;
}
```

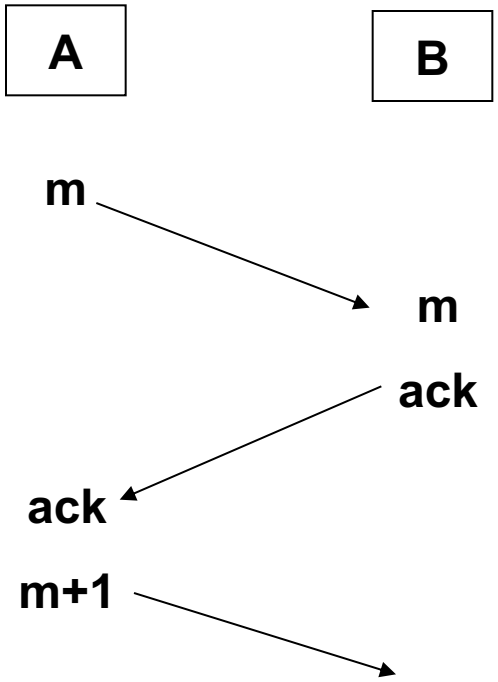
Protocol simplex start-stop

canalul fără erori
utilizatorul B nu poate accepta date în orice ritm



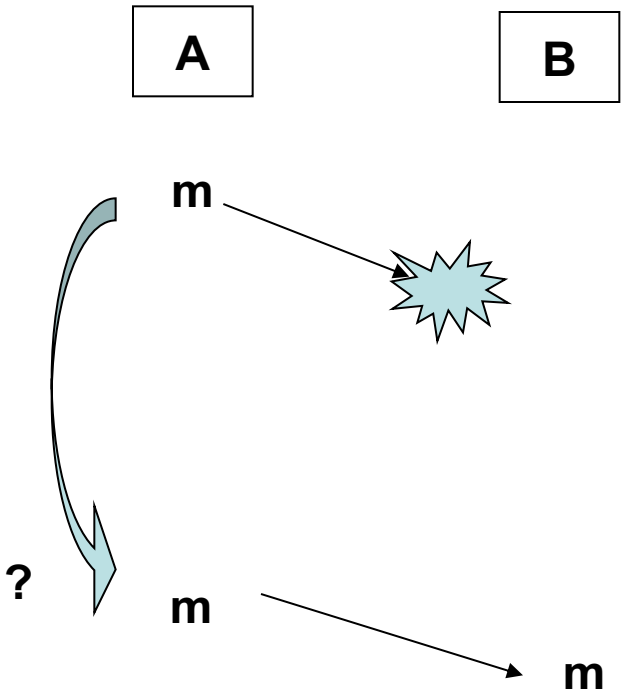
```
void transmit-start-stop(){
    cadru s;
    TipEven even;
    do{
        s.info=DeLaRetea();
        LaFizic(s);
        even=wait();                    //așteaptă permisiunea
     } forever;
}

void recept-start-stop(){
    cadru s,r;
    TipEven even;
    do{
        even=wait();                  //poate fi doar SosireCadru
        r=DeLaFizic();
        LaRetea(r.info);
        LaFizic(s);                     //transmite permisiunea
    } forever;
}
```

Protocol simplex pentru un canal cu erori



Entitaţi legătură de date

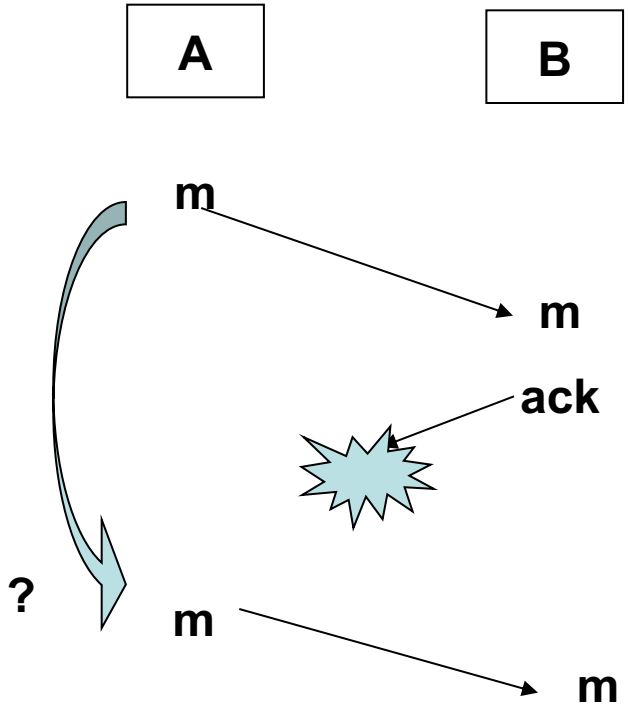
Transmisie corectă



Pierdere m

La time-out A retransmite m

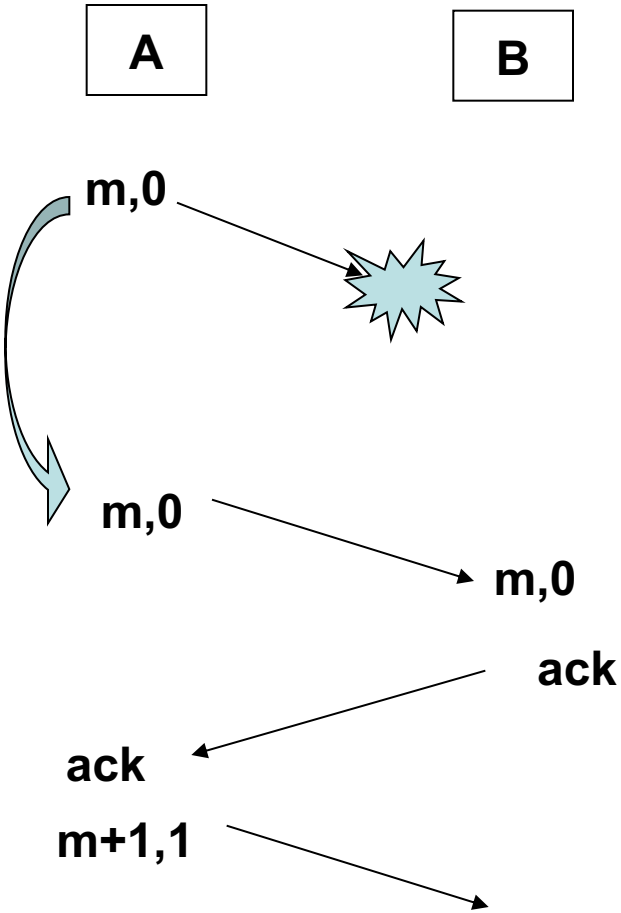
Care este acceptat corect de B

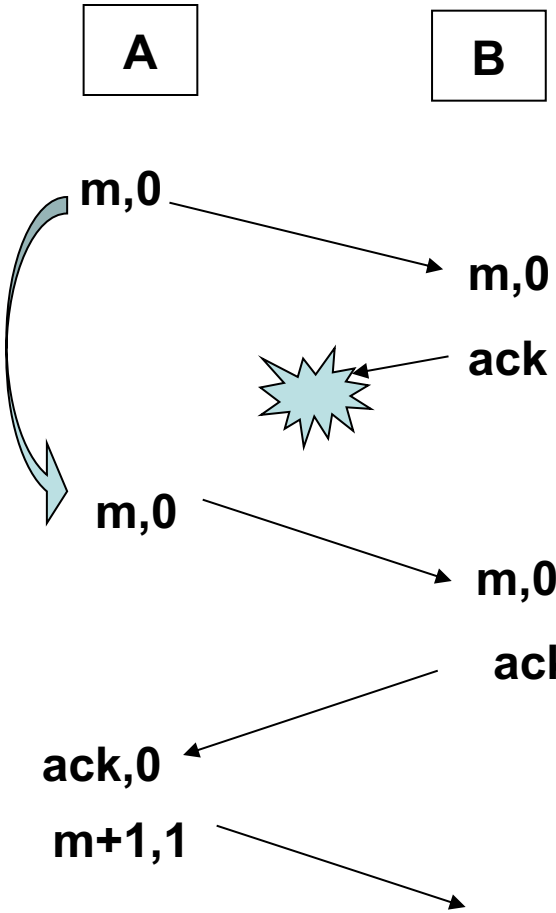


Pierdere ack

La time-out se retrimite m

Care este acceptat incorect, ca mesaj nou de B !





- acceptă

- ignoră

se adaugă un număr de secvență

la time-out se re-transmite ultimul cadru

B acceptă daca este corect

B ignoră dacă este dublură

Protocol simplex pentru un canal cu erori (2)

Este nevoie de un ceas

```
void StartCeas(NrSecv);
void StopCeas (NrSecv);
```

de eveniment TimeOut

si de numere de secvență - cadrele succesive m, m+1, m+2 au numerele de secvență alternante 0, 1, 0 … (protocol cu bit alternat)

fiecare cadru are câmpurile info și secv; al doilea modificat prin:

```
void inc (NrSecv&);
```

```
#define MaxSecv 1

void inc(NrSecv& k){
      k==MaxSecv ? k=0 : k++;
}
```

```
void transmit-start-stop-ARQ() {
    NrSecv CadruUrmator=0;
    cadru s;
    TipEven even;
    s.info=DeLaRetea();          //pregătește un cadru
    do{
        s.secv=CadruUrmator;     //adaugă nr secvență
        LaFizic(s);
        StartCeas(s.secv);
        even=wait();             // poate fi SosireCadru,
                                 // TimeOut sau
                                 // Eroarecontrol
        if(even==SosireCadru) {      //confirmare intactă
            StopCeas(s.secv);
            s.info=DeLaRetea();
            inc(CadruUrmator);
        }
    }forever;
}
```

```
void recept-start-stop-ARQ(){
    NrSecv CadruAsteptat=0;
    cadru r,s;
    TipEven even;
    do{
        even=wait();      //SosireCadru sau EroareControl
        if(even==SosireCadru){
            r=DeLaFizic();
            if(r.secv==CadruAsteptat){
                LaRetea(r.info);      //cadru în secvență
                inc(CadruAsteptat);
            }
            LaFizic(s);    //transmite oricum confirmarea
        }
    }forever;
}
```

Studiu individual

A. S. Tanenbaum Reţele de calculatoare, ed 4-a, BYBLOS 2003

3.1 ASPECTE ALE PROIECTĂRII NIVELULUI LEGĂTURĂ DE DATE

3.2.2 Coduri detectoare de erori

3.3 PROTOCOALE ELEMENTARE PENTRU LEGĂTURA DE DATE

3.4 PROTOCOALE CU FEREASTRĂ GLISANTĂ

3.6 EXEMPLE DE PROTOCOALE ALE LEGĂTURII DE DATE

A. S. Tanenbaum Computer networks, 5-th ed. PEARSON 2011

3.1 DATA LINK LAYER DESIGN ISSUES

3.2.2 Error-Detecting Codes

3.3 ELEMENTARY DATA LINK PROTOCOLS

3.4 SLIDING WINDOW PROTOCOLS

3.5.1 Packet over SONET

Exemple Protocoale Data Link

- HDLC – High-Level Data Link Control
- Legătura de date în Internet – cursul următor

HDLC – procedura LAPB

HDLC este o familie de protocoale

Tipuri stații (entități de protocol)

primară	(controlează) generează comenzi
secundară	(controlate) generează răspunsuri
combinată	generează ambele, comenzi și răspunsuri

Tipuri legătură

balansată	cu două stații combinate
nebalansată	o stație primară, una sau mai multe secundare

Moduri de transfer

NRM - Normal Response Mode (legătură nebalansată)
ABM - Asynchronous Balanced Mode
ARM - Asynchronous Response Mode (legătură nebalansată)

Procedura **LAPB** (Link Access Protocol Balanced) corespunde unei legături balansate cu stații combinate

High-Level Data Link Control

Format cadru

Bits	8	8	8	≥ 0	16	8
	0 1 1 1 1 1 1 0	Address	Control	Data	Checksum	0 1 1 1 1 1 1 0

Camp de Control pentru

Bits	1	3	1	3
(a)	0	Seq	P/F	Next
(b)	1	0	Type	P/F	Next
(c)	1	1	Type	P/F	Modifier

(a) Cadru de informație

(b) Cadru supervizor

(c) Cadru nenumerotat – gestiune legătură

Seq – număr de secvență cadru transmis (mod 8 sau 128)

Next - număr de secvență următorul cadru așteptat

P/F – poll/final – în comenzi, P = invitație la transmisie

– în răspunsuri toate cadrele au P, ultimul are F