

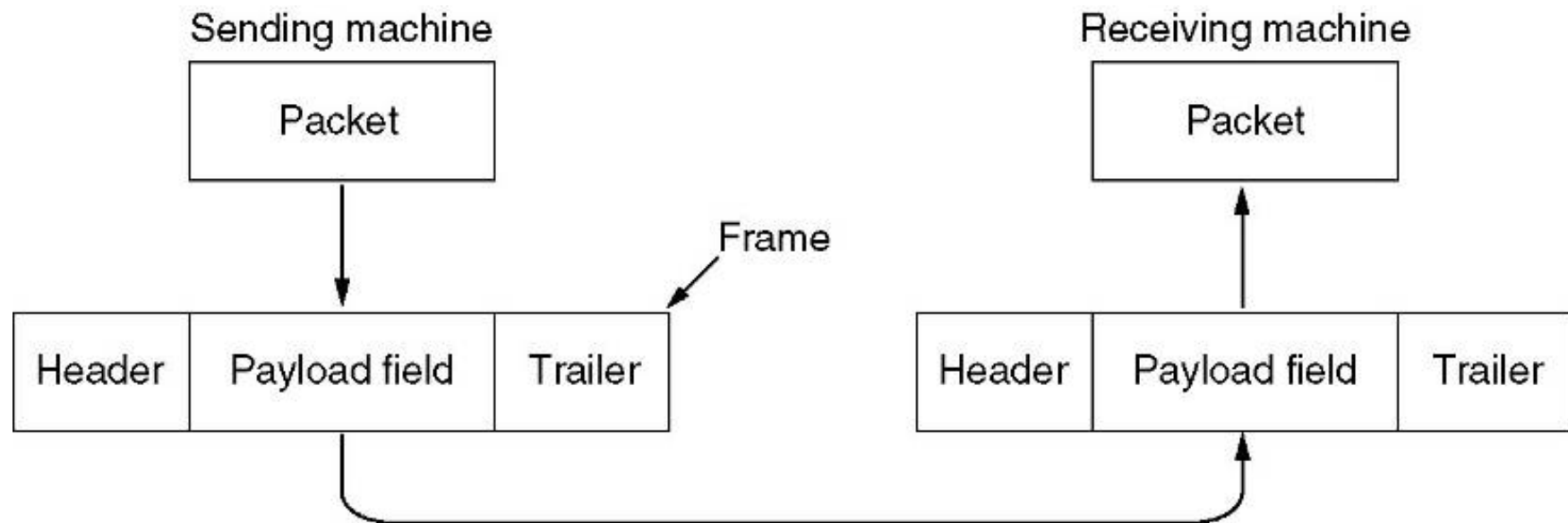


Nivelul legăturii de date

Funcțiile nivelului legăturii de date

1 – Încadrarea datelor

2 – Transmisia transparentă





3 – Controlul erorilor

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

4 – Controlul fluxului – protocol

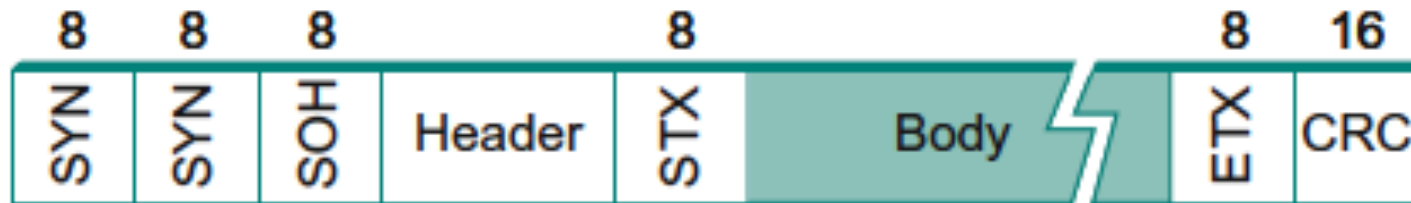
- mesaje de permisiune pentru transmițător

5 – Gestiunea legăturii – protocol

- stabilirea și desființarea legăturii
- re-inițializare după erori
- configurarea legăturii (stații primare și secundare, legături multipunct etc.)

Metode de încadrare

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



SOH - start of heading

ETX - end of text

EOT - end of transmission

ACK - acknowledge

SYN - synchronous idle

CRC - cyclic redundancy check

STX - start of text

ETB - end of transmission block

ENQ - enquiry

NAK - not acknowledge

DLE - data link escape



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 ?**

Solutie: **umplere cu caractere**

Dubleaza caracterele de control cu DLE

Defineste combinatii admise

DLE STX – start text transparent

DLE ETX – sfarsit text transparent

DLE STX **text...** **ETX** **...text** DLE ETX CRC

Dubleaza DLE la transmitere si elimina la receptie

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

Eroare: receptie DLE **x**

cu **x** diferit de STX, ETX, DLE



Umplere cu biți

Date de transmis includ un **flag fals** de terminare a cadrului

011**01111110**1111111111010

01111110 011011111101111111111010 01111110

Solutia: adaugarea unui zero dupa 5 unitati (in interiorul cadrului!)

01111110 011011111010111110111110010 01111110

Adaugarea se face indiferent daca dupa 5 unitati urmeaza 0 sau 1

Simplifica regula receptorului: **elimina zeroul aflat dupa 5 unitati**

011011111010111110111110010

011011111101111111111010



Detecția și corectarea erorilor

Noțiuni preliminare

- $A = \{0, 1\}$ alfabet binar
- W_n mulțimea cuvintelor \mathbf{w} de lungime n peste A

$$\mathbf{w} = w[0] w[1] \dots w[n-1], \text{ cu } w[i] \in A.$$

- **ponderea Hamming** a lui \mathbf{w} = numărul de unități conținute de \mathbf{w}
- **distanța Hamming**, $d(u,v)$ dintre \mathbf{u} și \mathbf{v} este

ponderea vectorului suma modulo 2 $\mathbf{u} + \mathbf{v}$



Detecția și corectarea erorilor

Ideea - utilizarea unei **submulțimi** a lui W_n pentru mesaje corecte

Condiția – erorile să producă cuvinte din restul mulțimii W_n

Multimea cuvintelor W_n de lungime n se impart in 2 categorii:

S_n este multimea cuvintelor cu sens

F_n este multimea cuvintelor fara sens

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

$$d(u,v) \geq r+1 \quad \text{pentru orice } u, v \text{ din } S_n$$

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

$$d(u,v) \geq 2r+1 \quad \text{pentru orice } u, v \text{ din } S_n$$



Exemplu

$$S_{10} = \{0000000000, 0000011111, 1111100000, 1111111111\}$$

$d(u,v) = 5 \Rightarrow$ putem corecta erori **duble**

Astfel,

00000**00**111 se corectează la 00000**11**111

Nu se pot corecta 3 erori:

0000000111 poate proveni din 0000000**000** în 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

Aceasta forma foloseste la calculul bitilor 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
Bit 1	1	?	1	?	1	0	0	?	0	0	1
Bit 2	1	0	1	?	1	0	0	?	0	0	1
Bit 4	1	0	1	1	1	0	0	?	0	0	1
Bit 8	1	0	1	1	1	0	0	1	0	0	1



Prima forma foloseste la **corectarea** erorilor

Ex.1 In loc de:

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

Sume **erodate** controlate de 2 (suma pozitiilor 2,3,6,7,10,11 nu este 0)
si de 8 (8,9,10,11)

$8 + 2 = 10$ => bitul din poziția 10 este **singurul** controlat de bitii 8 si 2 ➔ bit 10 este inversat

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

Bit 2 controlat de biții 2
 Bit 4 controlat de biții 4
 Bit 6 controlat de biții 2, 4
 Bit 8 controlat de biții 8
Bit 10 controlat de biții 2, 8



Ex. 2 În loc de:

			1	2	3	4	5	6	7	8	9	10	11
1100001	=>	1	<u>0</u>	1	1	1	0	0	1	0	0	0	1

Se primește eronat	1	<u>1</u>	1	1	1	0	0	1	0	0	0	1
--------------------	---	----------	---	---	---	---	---	---	---	---	---	---

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

1	<u>1</u>	1	1	1	1	0	0	1	0	0	0	1
---	----------	---	---	---	---	---	---	---	---	---	---	---

celelalte sume sunt corecte → bit 2 este inversat

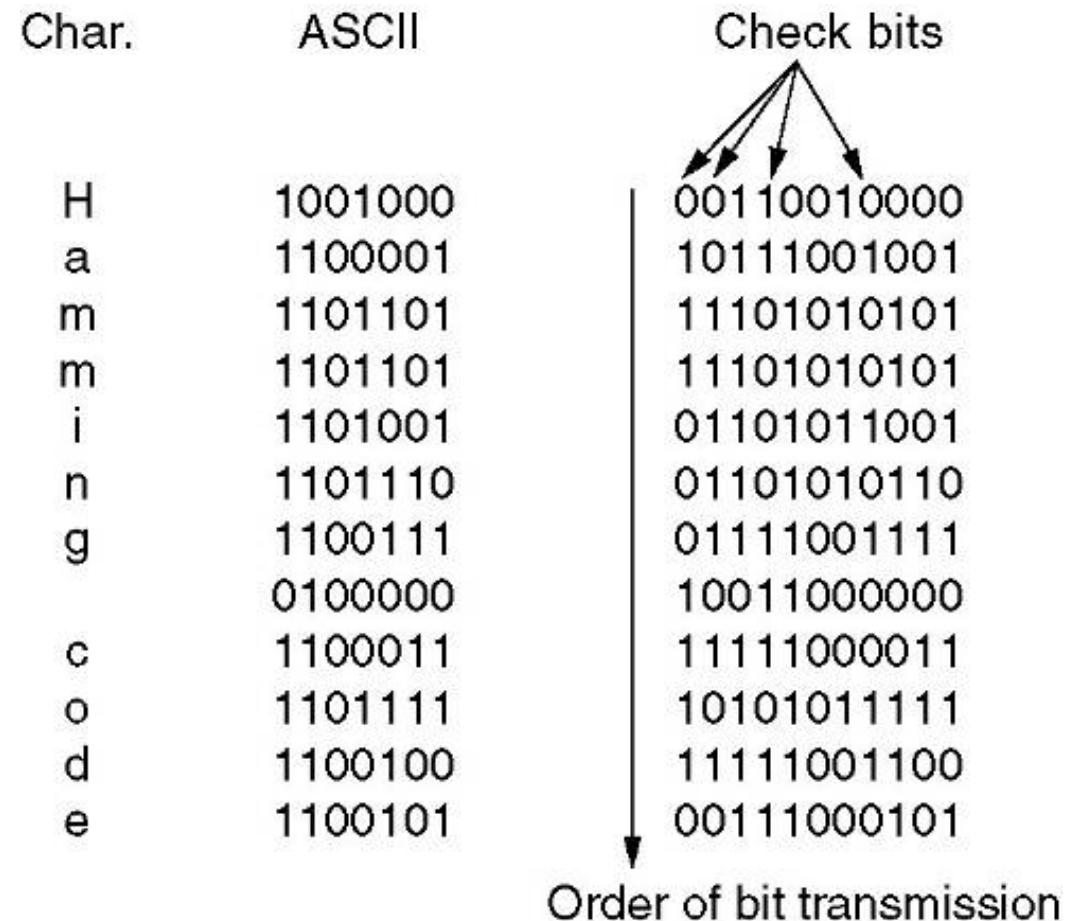
Obs.

Codul Hamming corectează erorile de 1 bit

Corecția erorilor in rafală

Utilizarea unui cod Hamming pentru **corecția erorilor in 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





Coduri detectoare de erori

Coduri polinomiale

k biți de informație (date)

i(X) polinomul corespunzător

Ex. **k=5** **10110**

$$i(X) = 1 \cdot X^4 + 0 \cdot X^3 + 1 \cdot X^2 + 1 \cdot X^1 + 0 \cdot X^0$$

n-k biți de control

r(X)

n biți în total

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

r(X) se alege astfel ca
sa fie multiplu de **g(X)**

$$w(X) = X^{(n-k)}i(X) + r(X)$$

$$w(X) = g(X) \cdot q(X)$$

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

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

r(X) este restul împărțirii lui **$X^{(n-k)} i(X)$** 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 biti informatie + 4 biti control

Imparte **11010110110000**

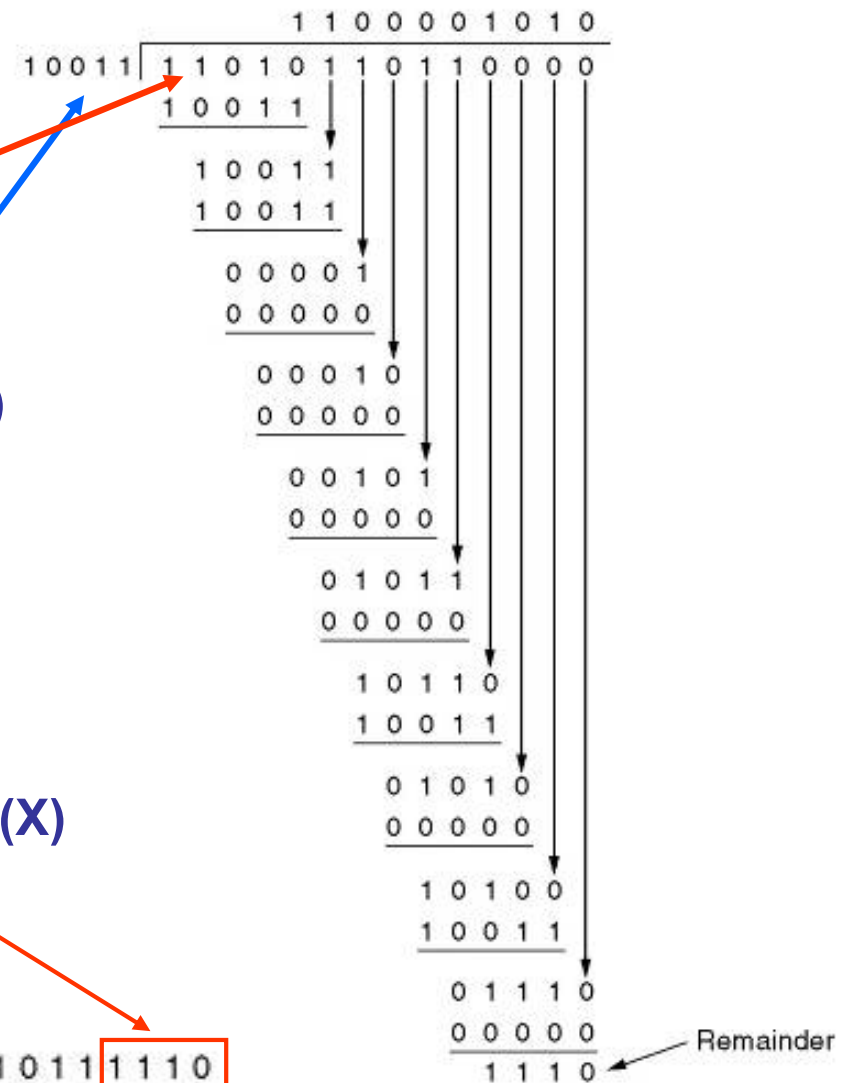
la **10011**

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

$g(X)$

$r(X)$ = rest împărțire $X^{(n-k)} i(X)$ la $g(X)$

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



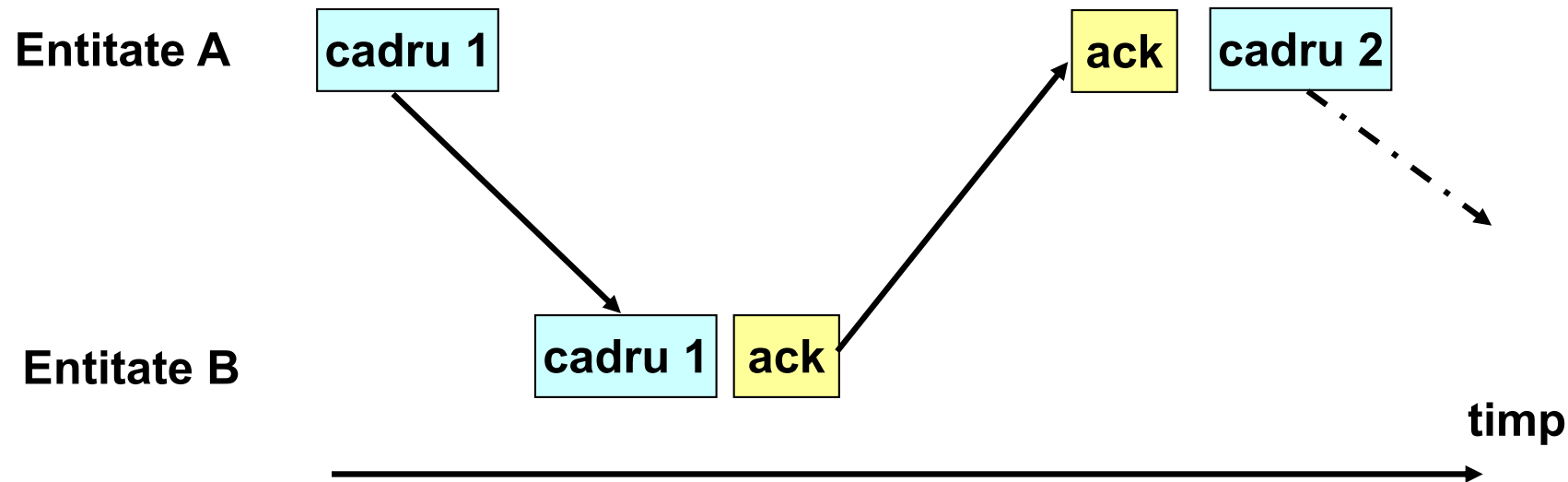


Ce erori pot fi detectate?

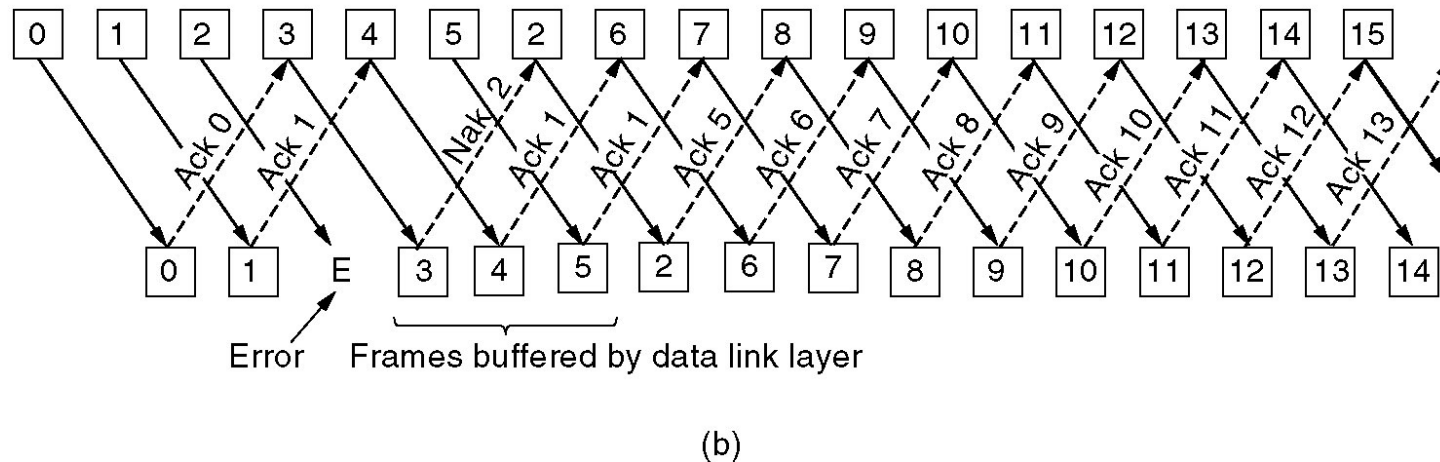
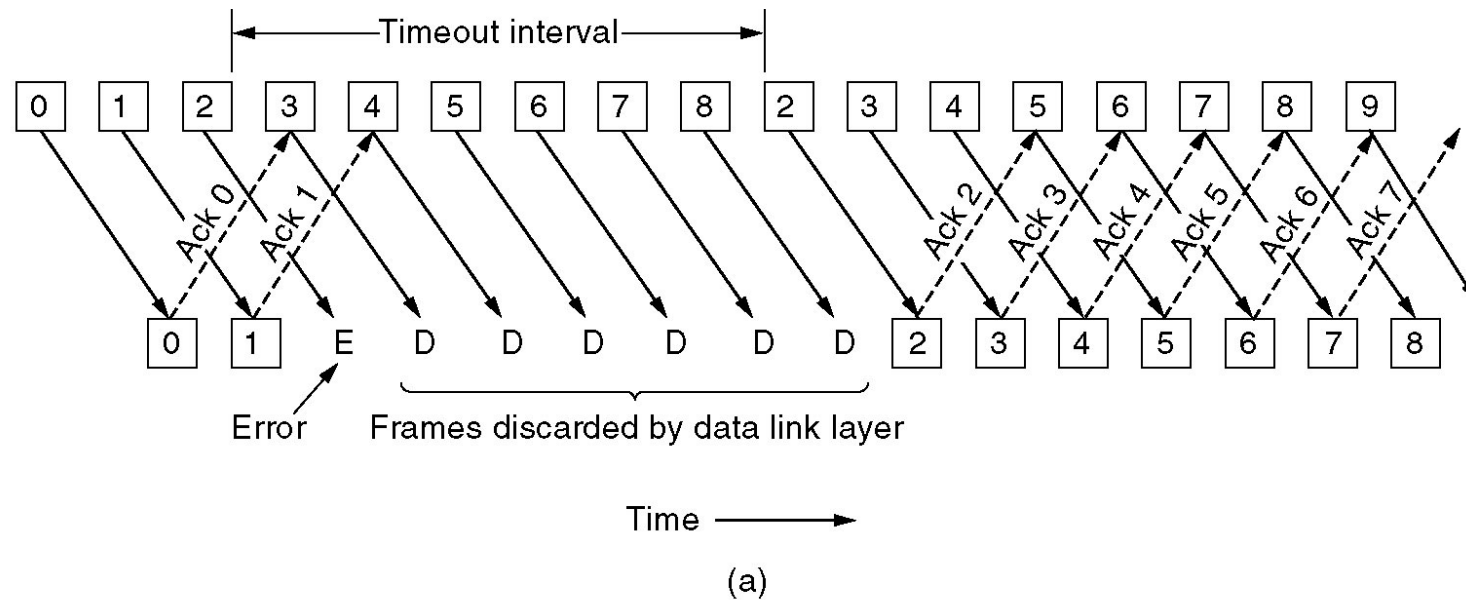
- Probabilitatea de detectie depinde de lungimea codului de control
- CRC si sume de control pe
 - 8 biti detecteaza 99.6094% din erori
 - $(x^8+x^2+x+1 \text{ CRC-8-CCITT})$
 - 16 biti detecteaza 99.9985% din erori
 - $(x^{16}+x^{10}+x^8+x^7+x^3+1 \text{ CRC-16})$
 - 32 biti detecteaza 99.9999% din erori
- In plus, CRC detecteaza 100% erori de
 - 1 bit;
 - 2 biti;
 - un numar impar de biti;
 - erori in rafala de lungimea codului CRC.

Protocoale elementare pentru legătura de date

Start-stop



Ferestre glisante





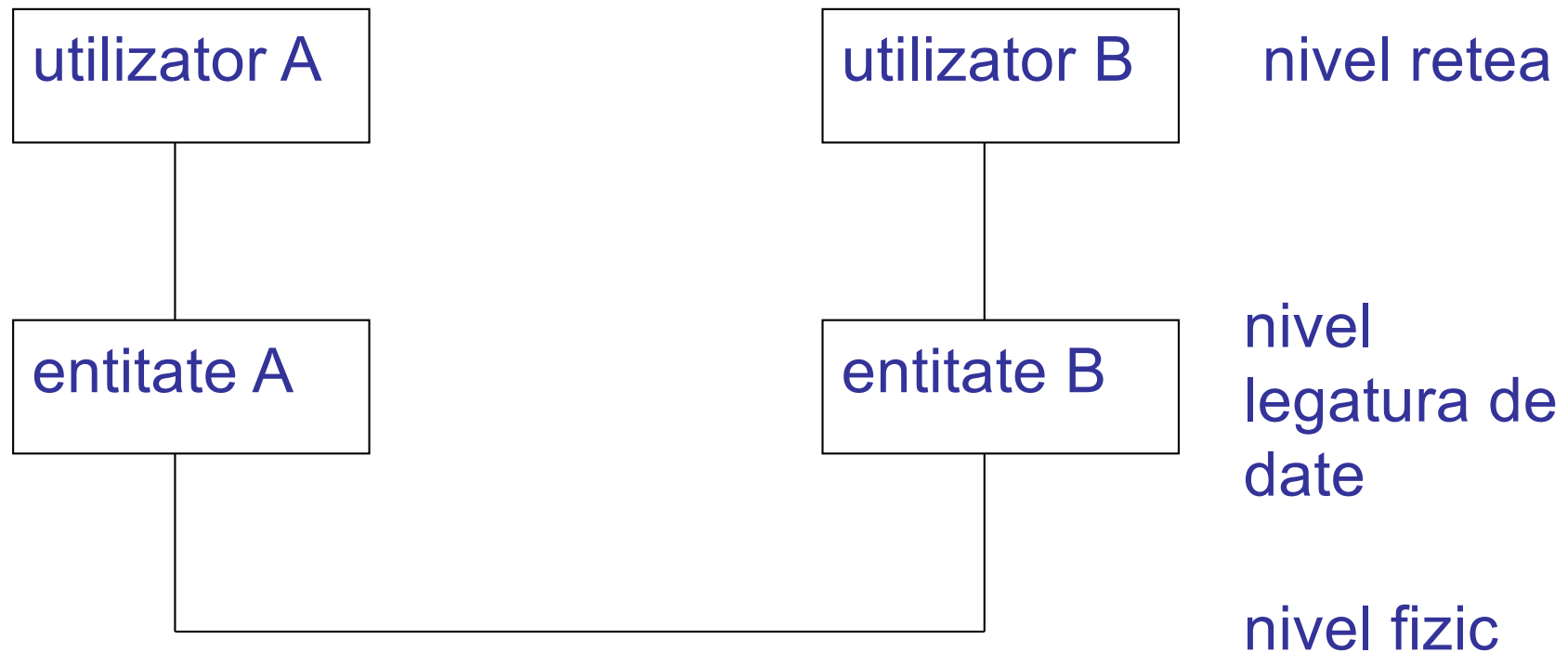
Specificație Protocol

- 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



Protocoloalele legăturii de date

Configurația entităților 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 interfata cu nivelul superior (retea)

- preluarea unui pachet de la retea pentru transmitere pe canal

pachet DeLaRetea () ;

- livrarea către retea a unui pachet

void LaRetea (pachet) ;

La interfata 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 fara restrictii

- 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 transmit1(){
    cadru s;
    do{
        s.info = DeLaRetea(); //preia pachet
        LaFizic(s);           //transmite cadru
    } forever;
}

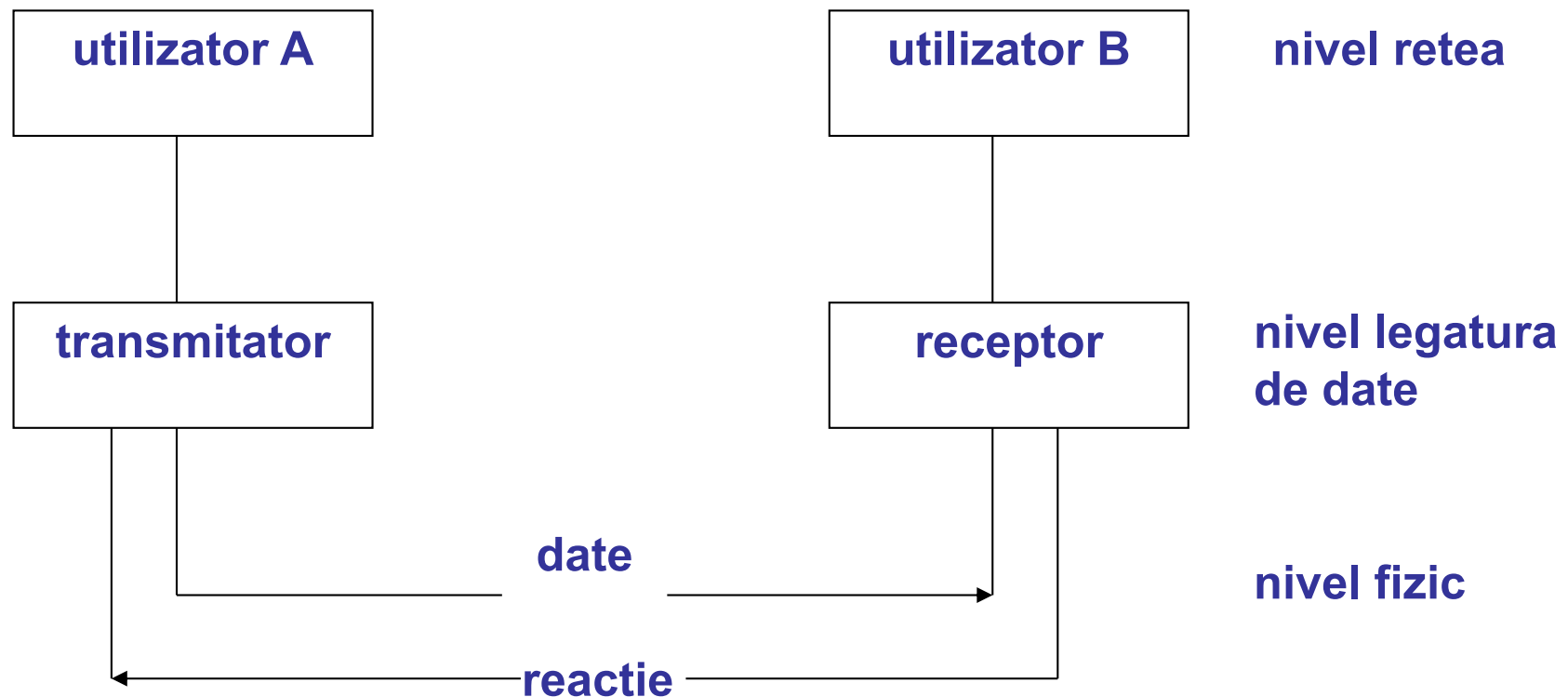
// entitatea din sistemul receptorului
void recept1(){
    cadru r;
    TipEven even;
    do{
        even = wait(); //asteapta cadru
        r = DeLaFizic(); //primeste cadru
        LaRetea(r.info); //preda pachet
    } forever;
}
```



Protocol simplex start-stop

canalul fara erori

utilizatorul B nu poate accepta date în orice ritm





```
void transmit2() {  
    cadru s;  
    TipEven even;  
    do{  
        s.info=DeLaRetea();  
        LaFizic(s);  
        even=wait();           //asteapta permisiunea  
    } forever;  
}
```

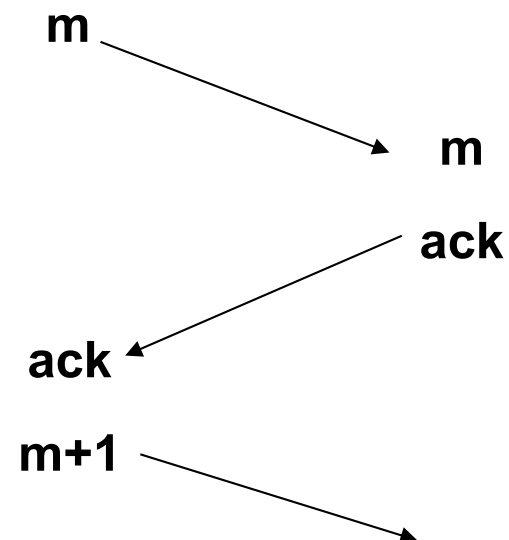
```
void recept2() {  
    cadru s,r;  
    TipEven even;  
    do{  
        even=wait();           //poate fi doar SosireCadru  
        r=DeLaFizic();  
        LaRetea(r.info);  
        LaFizic(s);           //transmite permisiunea  
    } forever;  
}
```



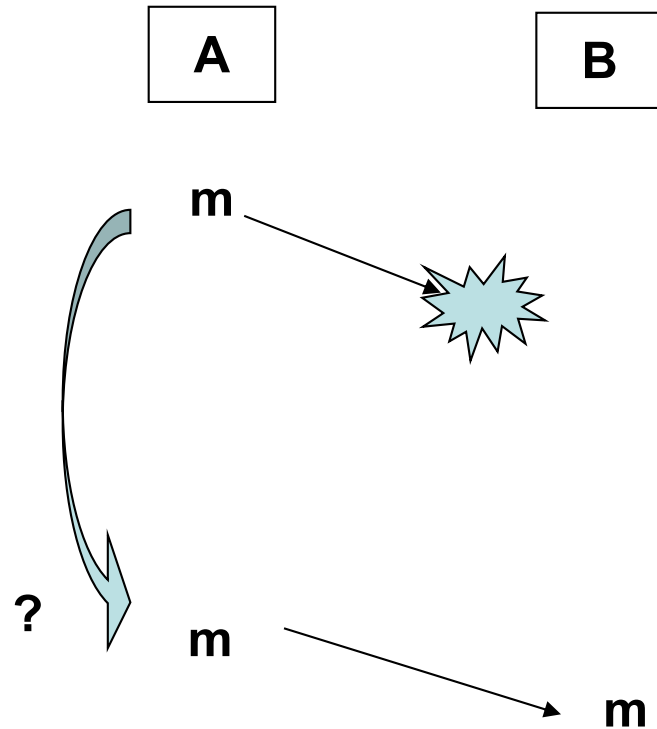
Protocol simplex pentru un canal cu erori



Entități legătură de date



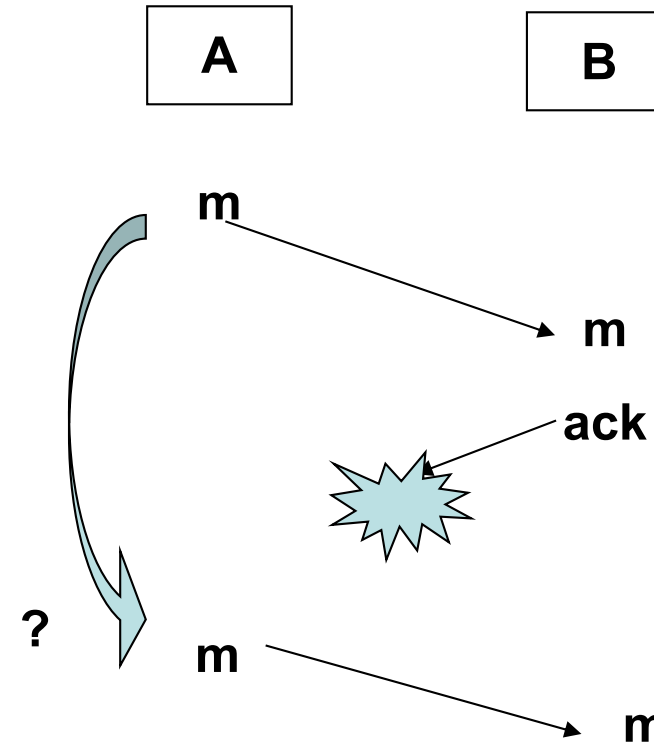
Transmisie corecta



Pierdere m

La **time-out** A retransmite m

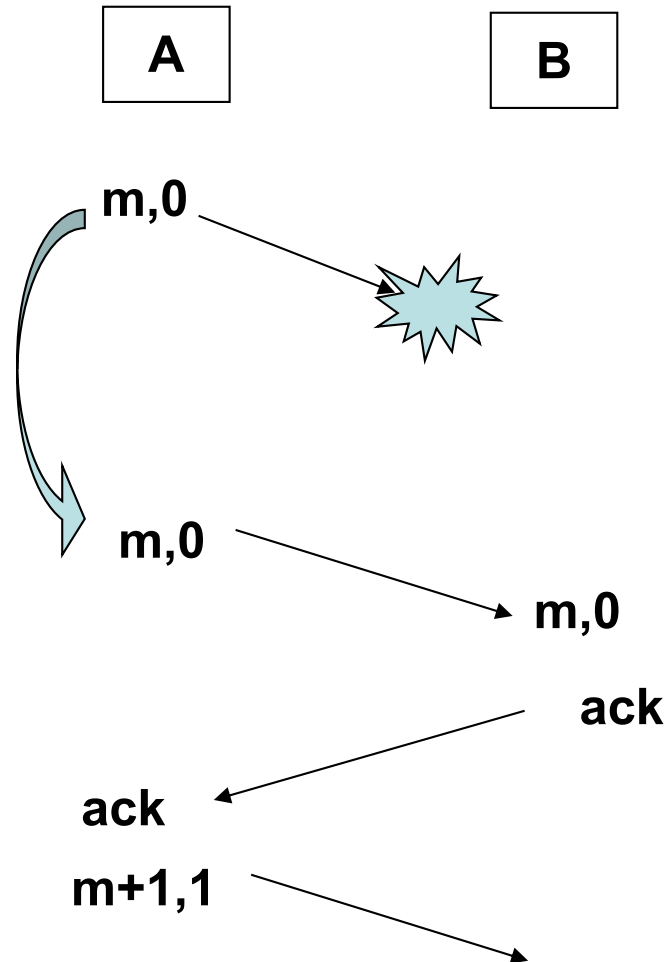
Care este acceptat corect de B



Pierdere ack

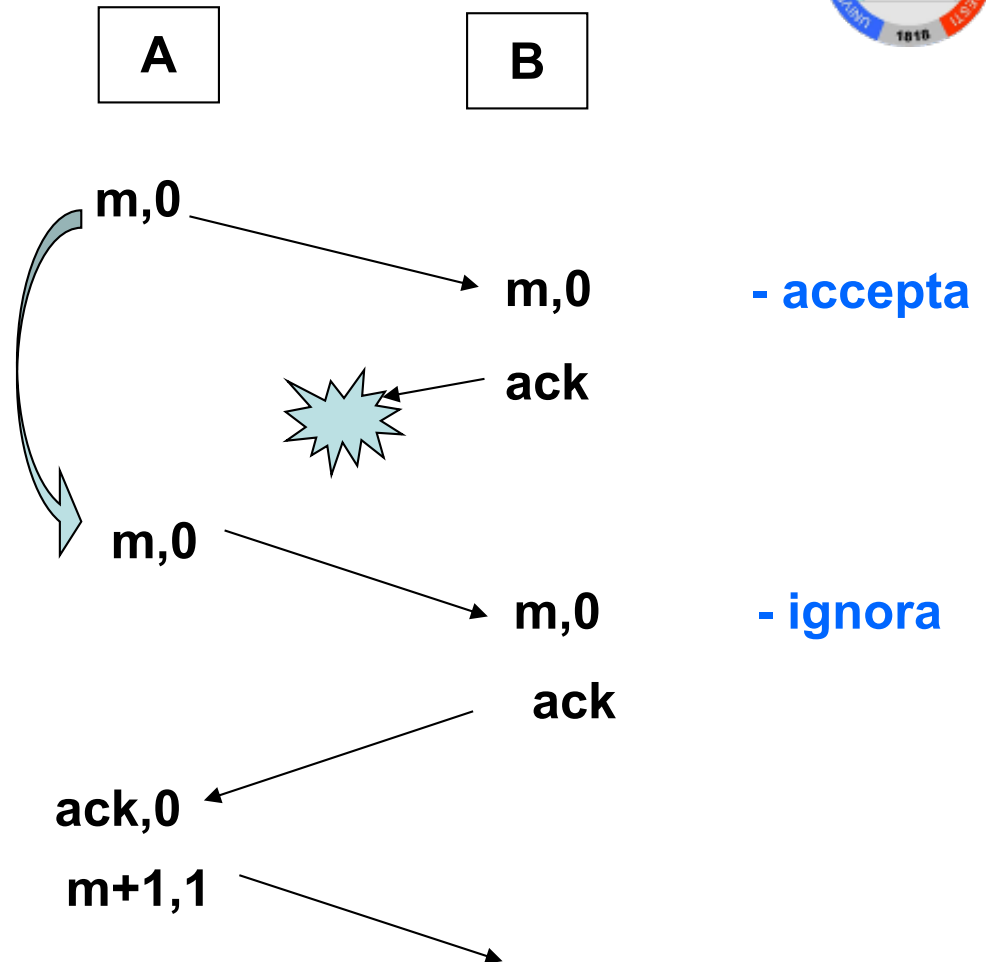
La **time-out** se retrimite m

Care este acceptat incorect, ca mesaj nou de B !



se adauga un **numar de secventa**
la time-out se re-transmite ultimul
cadru

B accepta daca este corect



B ignora daca este dublura



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 secventa** - cadrele succesive $m, m+1, m+2$ au numerele de secvență alternante 0, 1, 0 ... (**protocol cu bit alternat**)

fiecare cadru are campurile **info** si **secv**; al doilea modificat prin:

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

```
#define MaxSecv 1
```

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



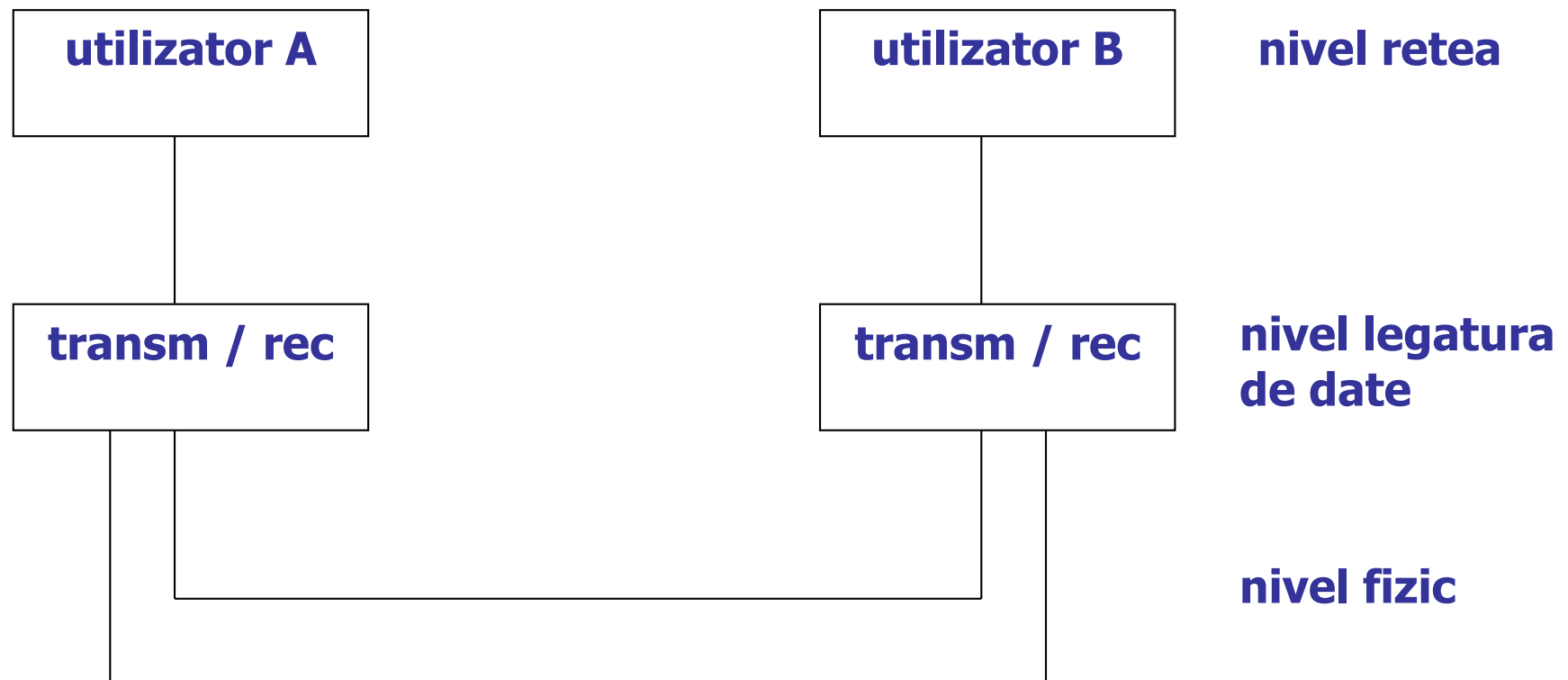
```
void transmit3() {  
    NrSecv CadruUrmator=0;  
    cadru s;  
    TipEven even;  
    s.info=DeLaRetea();           //pregateste un cadru  
    do{  
        s.secv=CadruUrmator;     //adauga nr secventa  
        LaFizic(s);  
        StartCeas(s.secv);  
        even=wait();             // poate fi SosireCadru,  
                                // TimeOut sau  
                                // Eroarecontrol  
        if(even==SosireCadru) {   //confirmare intacta  
            StopCeas(s.secv);  
            s.info=DeLaRetea();  
            inc(CadruUrmator);  
        }  
    }forever;  
}
```



```
void recept3() {  
    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 secventa  
                inc(CadruAsteptat);  
            }  
            LaFizic(s);      //transmite oricum confirmarea  
        }  
    }forever;  
}
```

Protocoloale cu fereastră glisantă

Configurația



2 legaturi pentru date+confirmare



Protocol cu numar de secventa de un bit

Fiecare stație

are o secventa de **initializare** in care trimite un prim cadru

si realizează **ciclic următoarele operații:**

receptia unui cadru,
prelucrarea sirului de cadre receptionate,
prelucrarea sirului de cadre transmise,
transmiterea sau retransmiterea unui cadru impreuna cu
confirmarea cadrului receptionat corect.

Fiecare cadru are campurile: info, secv, conf



```
void protocol4() {  
    NrSecv CadruUrmator=0; // 0 sau 1  
    NrSecv CadruAsteptat=0; // 0 saau 1  
    cadru r,s;  
    TipEven even; //SosireCadru, TimeOut  
                  //sau EroareControl  
  
    //pregateste cadru initial  
    s.info=DeLaRetea();  
    s.secv=CadruUrmator;  
    s.conf=1-CadruAsteptat;  
    LaFizic(s); //transmite cadrul  
    StartCear(s.secv);  
}
```



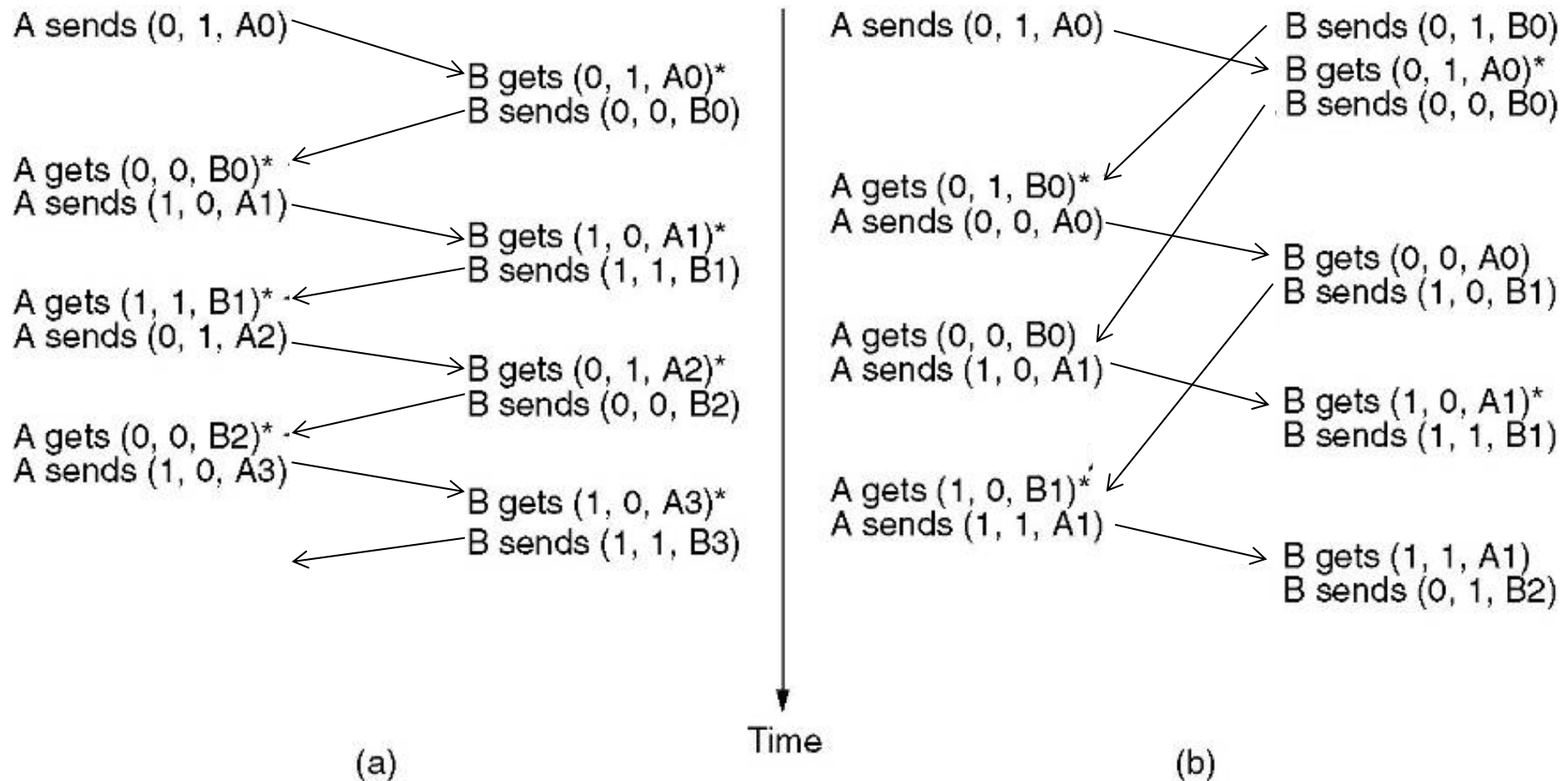
```
do{
    even=wait() ;
    if (even==SosireCadru) {
        r=DeLaFizic() ;
        //cand este cadrul asteptat, livreaza-l entitatii retea
        if (r.secv==CadruAsteptat) {
            LaRetea(r.info) ;
            inc(CadruAsteptat) ;
        }
        //cand cadrul transmis este confirmat, pregateste
        // urmatorul cadru
        if (r.conf==CadruUrmator) {
            StopCeas(r.conf) ;
            s.info=DeLaRetea() ;
            inc(CadruUrmator) ;
        }
    }
}
```



//construitiese si transmite un nou cadru

```
s.secv=CadruUrmator;  
s.conf=1-CadruAsteptat;  
LaFizic(s);  
StartCeas(s.secv);  
} forever; //reia de la asteptarea unui cadru  
}
```


Functionare protocol cu număr de secvență de un bit

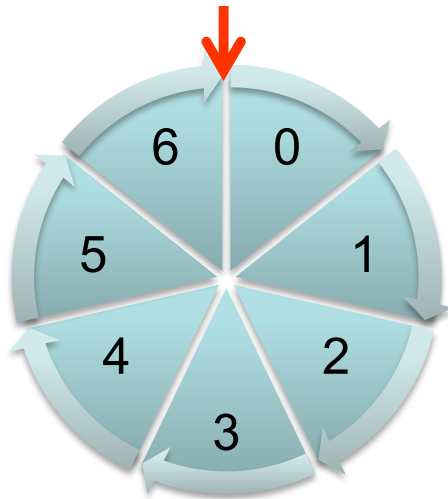


Două scenarii pentru protocolul 4. (a) Cazul normal. (b) Caz anormal.

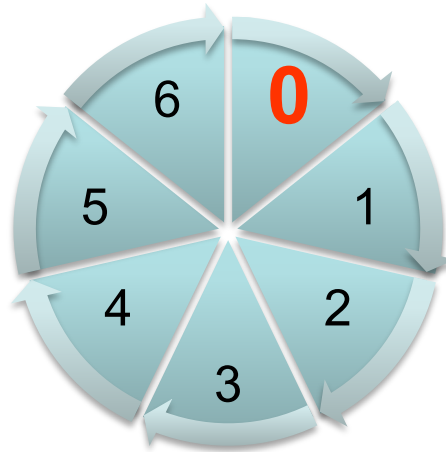
Notăția este (seq, ack, packet number).

Un asterisc arată că nivelul rețea acceptă pachetul.

Fereastra transmitatorului



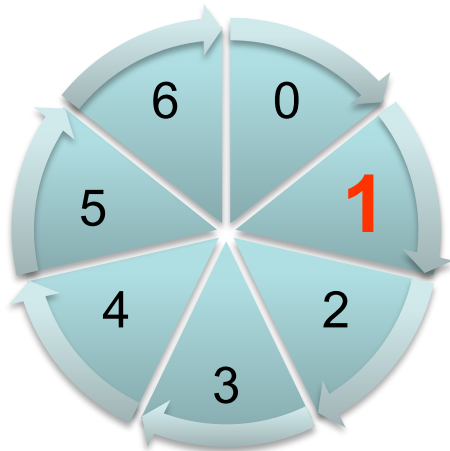
Initial
Fereastra Φ



Trimis cadru 0
Fereastra 0



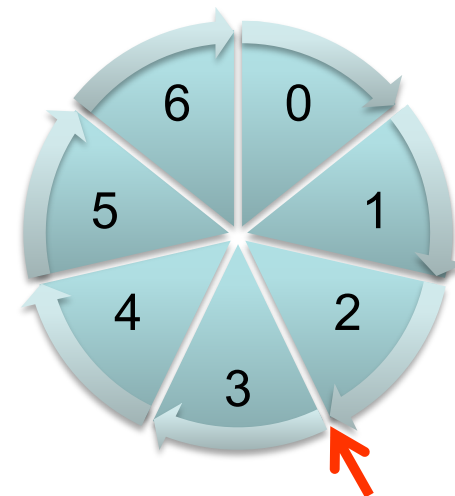
Trimis cadru 1
Fereastra 0,1



Primit ack 0
Fereastra 1

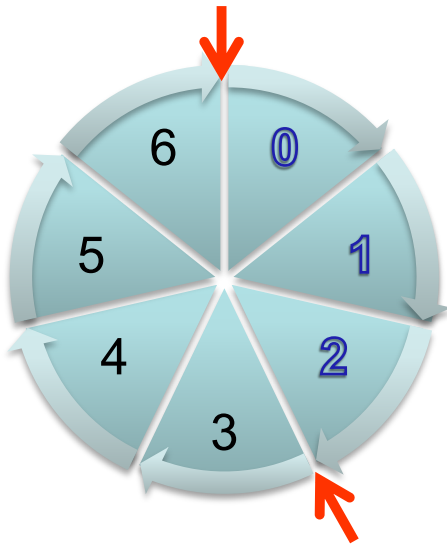


Trimis cadru 2
Fereastra 1,2

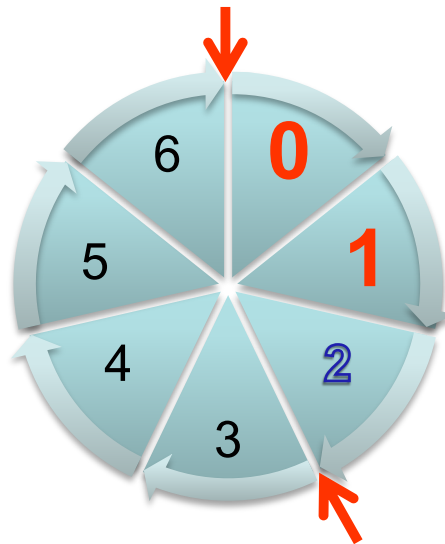


Primit ack 2
Fereastra Φ

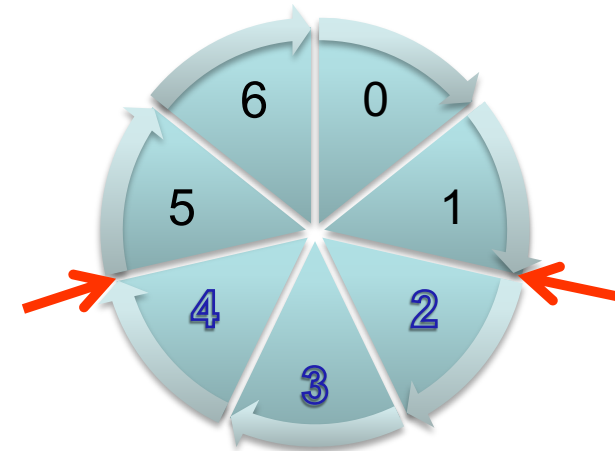
Fereastra receptorului



Loc ptr 3 cadre
Fereastra 0,1,2



Primit cadru 1 apoi 0



Livrat cadre 0 și 1
Fereastra 2,3,4



Fereastra glisantă

- fereastra este un sub-șir de numere de secvență

0, 1, 2, **3, 4, 5, 6**, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...

- pe parcursul transmiterii cadrelor, fereastra **glisează**

0, 1, 2, 3, 4, 5, **6, 7, 0, 1**, 2, 3, 4, 5, 6, 7, 0, 1 ...

- la **transmitator**, fereastra contine numerele cadrelor transmise si ne-confirmate

- dimensiunea ferestrei transmitatorului este **variabila**

– **crește** cand se trimite un nou cadru; ex. 7

0, 1, 2, **3, 4, 5, 6, 7**, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...

– **scade** cand se primește o confirmare; ex. pentru 3 si 4

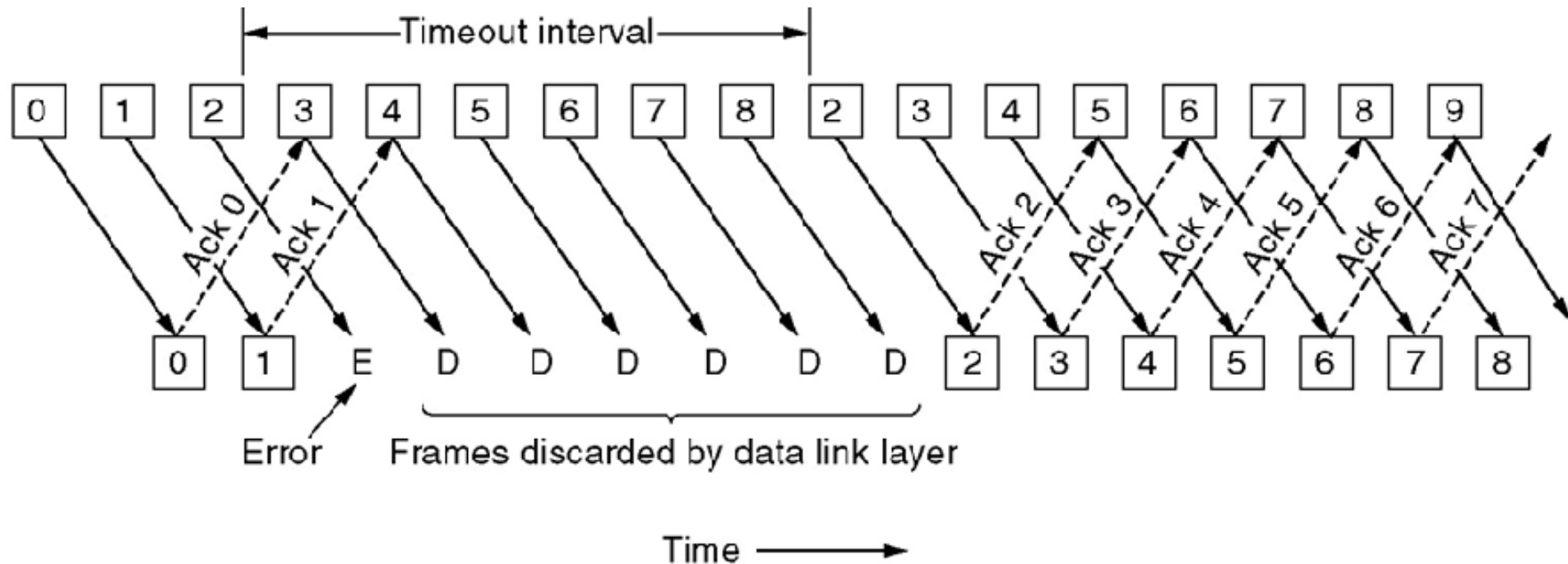
0, 1, 2, 3, 4, **5, 6, 7**, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...

Fereastra receptorului

- la **receptor**, fereastra specifica numerele cadrelor ce pot fi acceptate; in exemplu, se accepta doar cadrele 3, 4, 5, 6, 7
0, 1, 2, **3, 4, 5, 6, 7**, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...
- dimensiunea ferestrei receptorului este **constanta**
- fereastra **gliseaza** cand unul sau mai multe cadre din stanga ferestrei sunt livrate utilizatorului (livrarea se face in ordinea numerelor de secventa ale cadrelor!)
 - de ex. s-a livrat cadrul 3
0, 1, 2, 3, **4, 5, 6, 7, 0**, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...
 - apoi s-au livrat cadrele 4 si 5
0, 1, 2, 3, 4, 5, **6, 7, 0, 1, 2**, 3, 4, 5, 6, 7, 0, 1 ...

Un protocol cu retransmitere neselectivă “Go Back N”

Efectul erorii cand fereastra receptorului este 1.





Protocoale cu fereastră supraunitară de transmisie

Protocol cu retransmitere neselectivă

Sunt $\text{MaxSecv} + 1$ numere de secventa diferite

Fereastra maxima a transmitatorului poate fi de MaxSecv cadre

Demonstratie pe scenariu cu $\text{MaxSecv} = 7$ si fereastra de 8

1. Transmitatorul trimite cadrele 0..7;
2. Toate cadrele sunt receptionate si confirmate;
3. Toate confirmarile sunt pierdute;
4. Transmitatorul retrimite la **time-out** toate cadrele;
5. Receptorul **accepta** duplicatele.



```
#define MaxSecv 7
void ActivRetea();
void DezactivRetea();
NrSecv CadruUrmator, //urmatorul cadru de transmis
    ConfAsteptata, //cel mai vechi cadru neconfirmat
    // impreuna desemneaza fereastra transmisie
    CadruAsteptat;    //urmatorul cadru asteptat
cadru r,s;
pachet tampon[MaxSecv+1];
NrSecv ntampon,i;
TipEven even;
```




```
short intre(NrSecv a, NrSecv b, NrSecv c) {  
    //intoarce 1 daca a<=b<c circular  
    return a<=b && b<c || c<a && a<=b || b<c && c<a;  
}
```

ex.1

0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...
 a b c

ex.2

0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...
 a b c

ex.3

0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...
 a b c



```
void transmite(NrSecv nrcadru) {  
    //construieste si transmite un cadru de date  
    //pentru pachetul din tampon[nrcadru]  
  
    s.info=tampon[nrcadru] ;  
    s.secv=nrcadru ;  
    s.conf=(CadruAsteptat+MaxSecv) % (MaxSecv+1) ;  
    //confirmarea trimisa in cadrul de date!  
    LaFizic(s) ;  
    StartCeas(nrcadru) ;  
}
```



```
void protocol5() {
    ActivRetea();           // permite even. "ReteaPregatita"
    CadruAsteptat=0;        // fereastră de receptie 1 cadru
    CadruUrmator=0;         // margine sup fereastră transmisie
    ConfAsteptata=0;        // margine inf fereastră transmisie
    ntampon=0;              // initial nici un pachet in tampon
    do{
        even=wait();        // functionare total dirijata
                             // de evenimente

        switch(even) {
            case ReteaPregatita: // rețeaua are de transmis
                //memoreaza pachet in tampon (ptr retransmisie)
                //si trimite-1 pe legatura de date
                tampon[CadruUrmator]=DeLaRetea();
                ntampon++;
                transmite(CadruUrmator);
                inc(CadruUrmator);
                break;
        }
    }
}
```

```
case SosireCadru:
```

```
  r=DeLaFizic();
```

```
  if (r.secv==CadruAsteptat) {
```

```
    LaRetea(r.info);
```

```
    inc(CadruAsteptat);
```

```
  }
```

```
  while(intre(ConfAsteptata, r.conf, CadruUrmator))
```

```
  {
```

```
    //confirma cadre intre ConfAsteptata si r.conf
```

```
    ntampon--;
```

```
    StopCeas(ConfAsteptata);
```

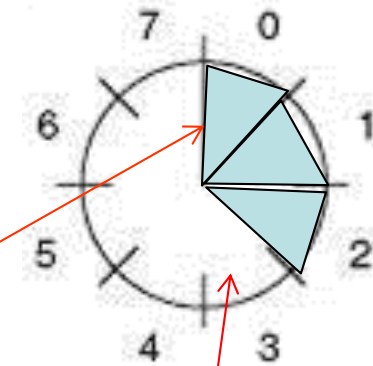
```
    inc(ConfAsteptata);
```

```
  }
```

```
  //la iesire din bucla ConfAsteptata = r.conf+1
```

```
  break;
```

```
case EroareControl: break;    // ignora cadru eronate
```



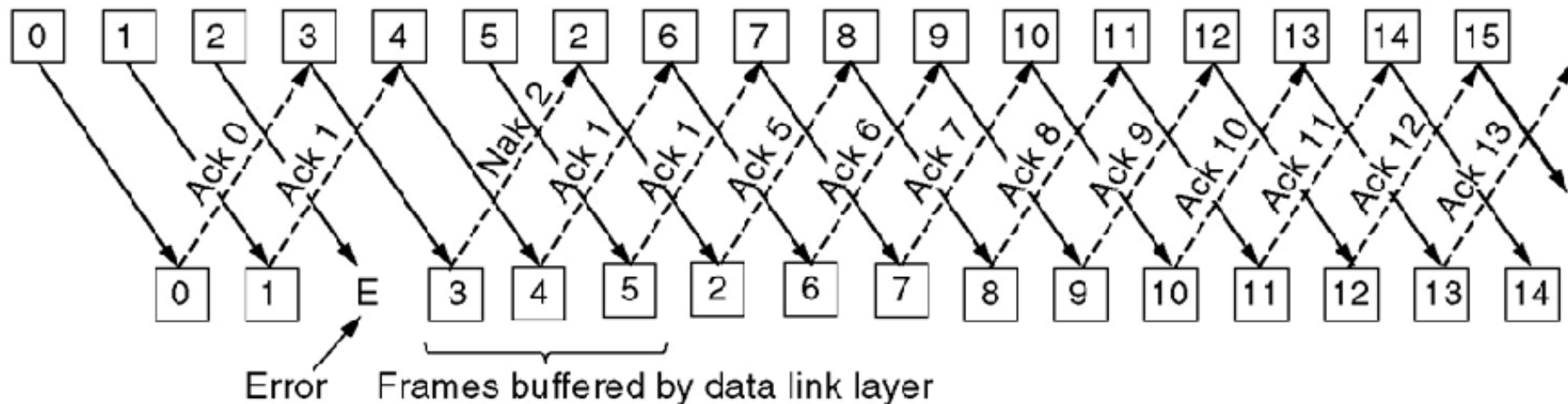


```
case TimeOut:  // retransmite toate cadrele din
               // tampon (fereastra transmitatorului)
               //incepand cu pozitia ConfAsteptata
CadruUrmator=ConfAsteptata;
for (i=1;i<=ntampon;i++) {
               transmite (CadruUrmator) ;
               inc (CadruUrmator) ;
           }
}               // sfarsit switch
if (ntampon<MaxSecv) ActivRetea() ;
else DezactivRetea() ;
}forever;
}               // sfarsit protocol5
```

Protocol cu retransmitere selectiva

Fereastra receptorului este supraunitara

- transmite **Nak** cu numarul **2** pentru cadru eronat
- apoi reconfirma ultimul cadru corect receptionat (**Ack 1**)
- dupa re-primirea cadrului eronat, **Ack 5** confirma toate cadrele pastrate in tampon





Dimensiunea ferestrei de receptie

Fereastra receptorului nu poate fi egală cu cea a transmițătorului

1. Transmitatorul trimite cadrele 0..6
2. Cadrele sunt receptionate si confirmate. Fereastra receptorului devine
7, 0, 1, 2, 3, 4, 5
3. Toate confirmările sunt pierdute (**se strica sincronizarea** între transmițător și receptor)
4. Transmitatorul **retrimite** cadrul **0** la time-out
5. Receptorul accepta drept **cadru nou** aceasta copie (cadrul **0**) care se potrivește în fereastra sa actuală (7,**0**,1,2,3,4,5); cere cadrul 7 (dinaintea lui **0**) care lipsește
6. Transmițătorul interpretează că a trimis corect cadrele 0..6 și trimite
7, **0**, 1, 2, 3, 4, 5
7. Receptorul acceptă cadrele, cu excepția lui **0**, pentru care are deja un cadru receptionat. → **Ignora acest cadru 0** (a luat în loc duplicatul **0** primit anterior).



```
void protocol6() {  
    initializari_contoare;  
    do{even=wait();  
        switch (even) {  
            case ReteaPregatita:  
                accepta_salveaza_si_transmite_un_cadru (+-ack) ;  
                //ack se poate include in cadru de date  
                //se poate trimite si separat in cadru ack  
            break;  
        }  
    }  
}
```




```
case SosireCadru:
    r=DeLaFizic();
    if (r.fel == data){
        transm_nak_daca_r_difera_de_cadru_asteptat;
        //pentru a semnala rapid eroarea!
        //NU mai multe nak-uri pentru acelasi cadru
        accepta_cadru_daca_in_fereastrareceptie;
        if (sunt date de trimis)
            marcheaza_trebuie_trimis_ack;
            //ack se va trimite cu un cadru de date
        else
            trimite_ack;
        livreaza_in_ordine_pachetele_sosite;
        actualizeaza_fereastrareceptie;
    }
```



```
//continua case SosireCadru
```

```
    if (r.fel == nak) retransmite_cadru_cerut;  
    trateaza_confirmare_cadre_eliberind_buffere;  
    break;
```

```
case EroareControl:
```

```
    transmite_nak;
```

```
    break;
```

```
case TimeOut:
```

```
    retransmite_cadrul_corespunzator;
```

```
    // timeout este asociat nr. secv al unui cadru
```

```
}
```

```
    activeaza_sau_dezactiveaza_nivel_retea;
```

```
}forever;
```

```
}
```



Exemple Protocole Data Link

- IEEE 802.11Qbb - Priority flow control
- Legatura de date in Internet



Priority Flow Control – IEEE 802.1Qbb

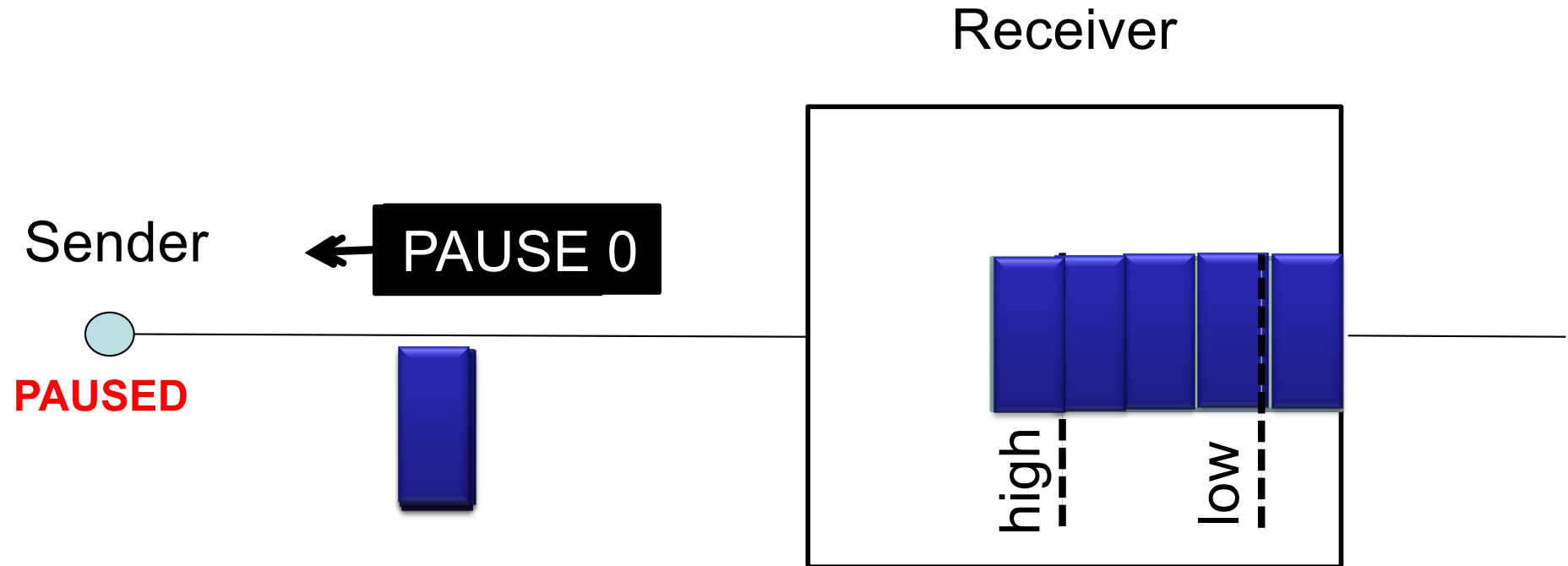
- Standardul Ethernet (IEEE 802.3) este folosit pe scara larga
 - Asigura conexiuni punct-la-punct
 - In mod standard livrarea este best-effort.

Format Cadru Ethernet

Preambul	SFD	DST MAC	SRC MAC	EtherType	Payload	FCS
----------	-----	---------	---------	-----------	---------	-----

- Priority Flow Control este o varianta a Ethernet care garanteaza livrarea cadrelor.

Functionare Priority Flow Control



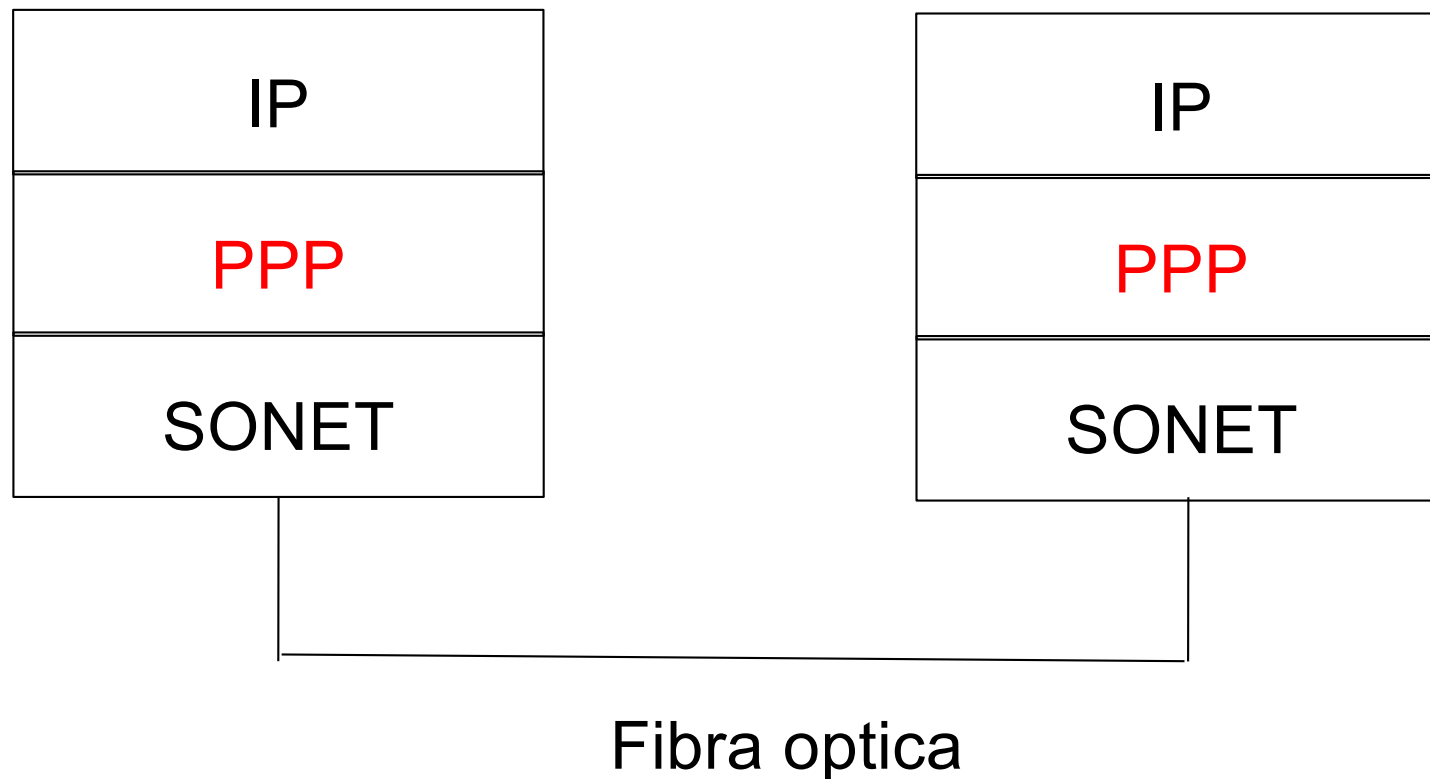


Priority Flow Control

- PFC este Folosit pentru a transporta RDMA peste Ethernet (ROCEv2)
 - RDMA necesita o retea fara pierderi.
 - Folosit de Microsoft Azure pentru montarea diskurilor in retea
- PFC are probleme de performanta
 - Tree saturation problem.
 - Head of line blocking.

Legatura de date in Internet

Comunicarea pe fibra optica



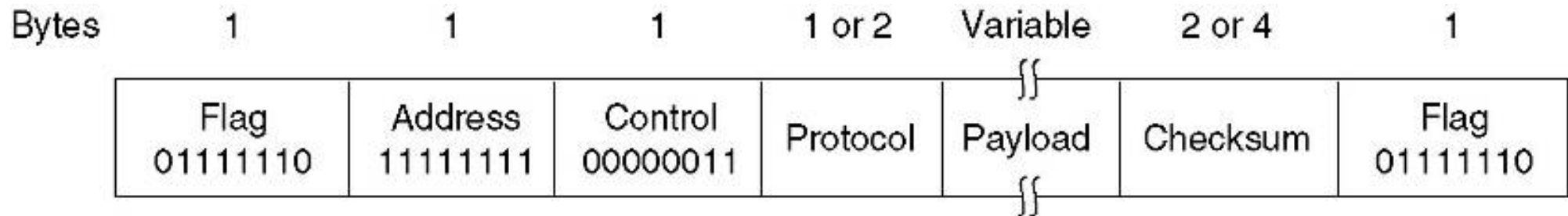


PPP – Point to Point Protocol

Ofera incadrare

Link Control Protocol, LCP

Network Control Protocol, NCP



Format de cadru PPP pentru modul nenumerotat

Addressa 11111111 = toate statiile accepta cadrul

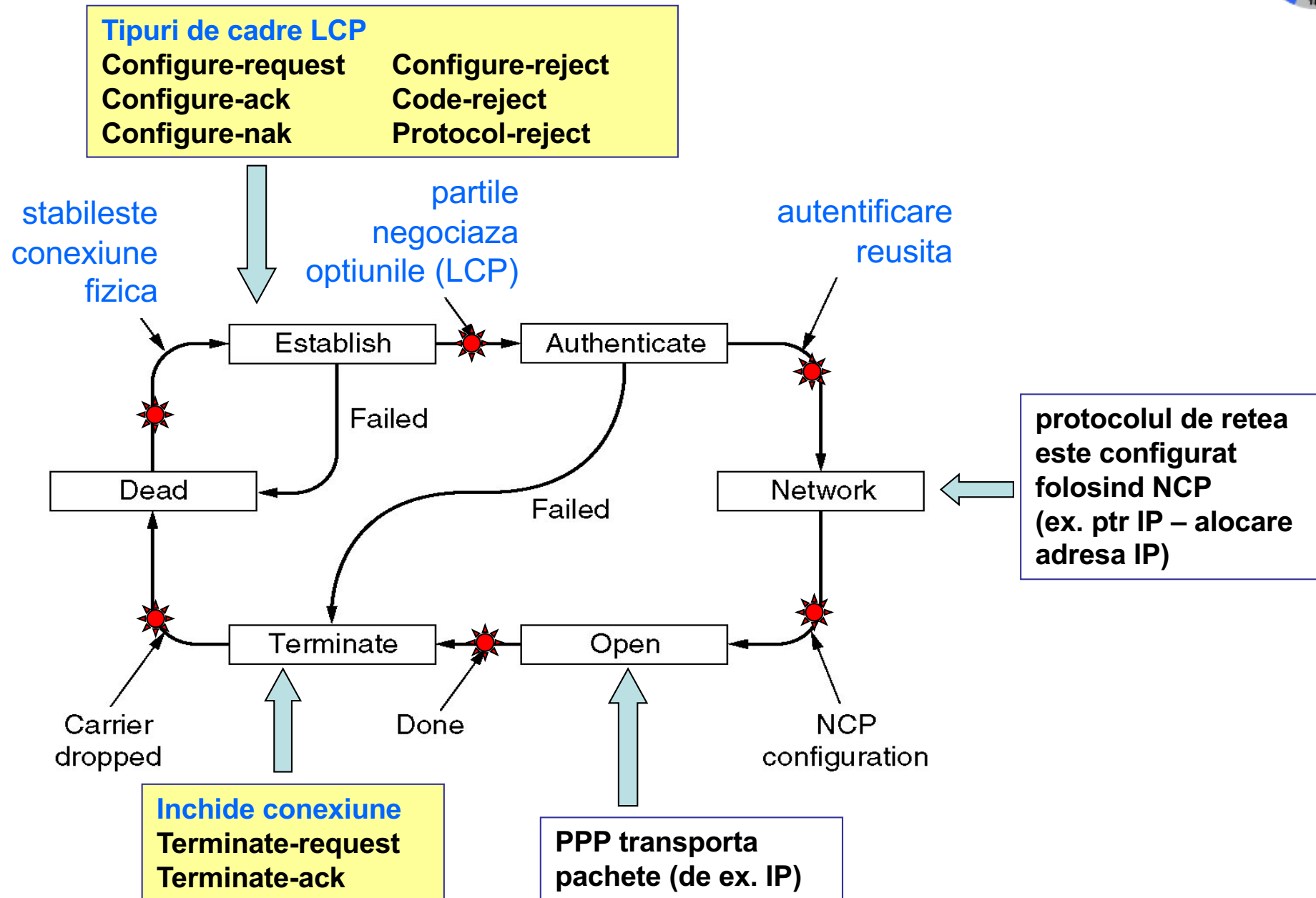
Control 00000011 = nenumerotat

Protocol = selecteaza dintre

LCP, NCP

IP, IPX (Internetwork Packet eXchange), OSI CLNP, XNS (Xerox Network Services)

PPP – Point to Point Protocol (2)





Tipuri de cadre LCP

Name	Direction	Description
Configure-request	$I \rightarrow R$	List of proposed options and values
Configure-ack	$I \leftarrow R$	All options are accepted
Configure-nak	$I \leftarrow R$	Some options are not accepted
Configure-reject	$I \leftarrow R$	Some options are not negotiable
Terminate-request	$I \rightarrow R$	Request to shut the line down
Terminate-ack	$I \leftarrow R$	OK, line shut down
Code-reject	$I \leftarrow R$	Unknown request received
Protocol-reject	$I \leftarrow R$	Unknown protocol requested
Echo-request	$I \rightarrow R$	Please send this frame back
Echo-reply	$I \leftarrow R$	Here is the frame back
Discard-request	$I \rightarrow R$	Just discard this frame (for testing)

I - Initiator

R - Responder



PPPoE

- Varianta a PPP encapsulata peste Ethernet; folosit pe scara larga
- PPPoE are doua faze:
 - Descoperire (discovery)
 - Sesiune



PPPoE Discovery

- Clientul trimite catre server un pachet de initiere (PADI)
 - Frame Ethernet catre adresa broadcast

```
Frame 1 (44 bytes on wire, 44 bytes captured)
Ethernet II, Src: 00:50:da:42:d7:df, Dst: ff:ff:ff:ff:ff:ff
PPP-over-Ethernet Discovery
  Version: 1
  Type 1
  Code Active Discovery Initiation (PADI)
  Session ID: 0000
  Payload Length: 24
PPPoE Tags
  Tag: Service-Name
  Tag: Host-Uniq
    Binary Data: (16 bytes)
  Tag: Host-Uniq
    Binary Data: (16 bytes)
```



Studiu individual

A. S. Tanenbaum Rețele 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