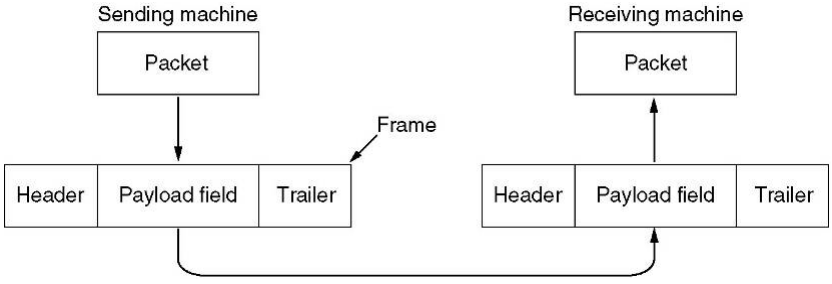
Nivelul legăturii de date

Funcțiile nivelului legăturii de date

1 – Încadrarea



Metode de încadrare

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

SYN SYN SOH header STX text ETX CRC

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

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

0110111111101111111111010

01111110 0110111111101111111111010 01111110

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

01111110 0110111111010111111011111010 01111110

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

Simplifica regula receptorului: elimina zeroul aflat dupa 5 unitati

0110111111010111111011111010

0110111111101111111111010

3 – Controlul erorilor

- secvenţa de control a cadrului - FCS - frame checking sequence
- mesaje de confirmare
- ceasuri
- numere de secvenţă

4 – Controlul fluxului

- utilizarea mesajelor de permisiune pentru transmiţător

5 – Gestiunea legăturii

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

Detecţia şi corectarea erorilor

Coduri corectoare de erori

- A = {0, 1} alfabet binar
- W_n mulţimea 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**
- distanţa Hamming, d(u,v) dintre **u** şi **v**

  $W_n = S_n \cup F_n$
- Pentru u, v ϵ S_n şi r erori:

  d(u,v) >= r+1 detecţie

  d(u,v) >= 2r+1 corecţie
- Exemplu:

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

  d(u,v) = 5 => putem corecta erori duble

  0000000111 Se corectează la 0000011111

  0000000111 Poate proveni din 0000000000

Metoda Hamming

Biţi numerotaţi de la 1 (stânga) la n (dreapta)

Codificare:

Biţii 1, 2, 4, 8, ... (puteri ale lui 2) sunt de control

Control paritate (pară sau impară)

Bitul k este controlat de biţii ale căror poziţii însumate dau k; reciproc:

- Bit 1 controlează biţii 1, 3, 5, 7, 9, 11
- Bit 2 controlează biţii 2, 3, 6, 7, 10, 11
- Bit 4 controlează biţii 4, 5, 6, 7
- Bit 8 controlează biţii 8, 9, 10, 11

Exemplu (paritate pară)

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

Biţi de control eronaţi 2, 8

8 + 2 = 10 => bit din poziţia 10 este inversat

Codul Hamming corecteaza erorile de 1 bit

Corecția erorilor in rafală

Utilizarea unui cod Hamming pentru corecția erorilor in rafală

- matricea de biti este transmisa coloana cu coloana
- poate corecta erori in rafala dintr-o coloana

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)
n-k biți de control
n biți în total

i(X) polinomul corespunzător
r(X)

r(X) se alege astfel ca

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

sa fie multiplu de g(X)

$$w(X) = g(X).q(X)$$

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

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

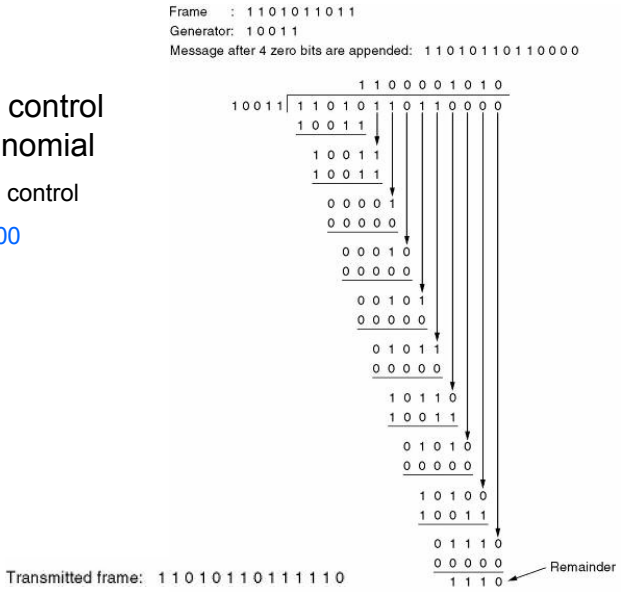
Coduri detectoare de erori

Calculul sumei de control pentru un cod polinomial

10 biti informatie + 4 biti control

Imparte 11010110110000

la 10011



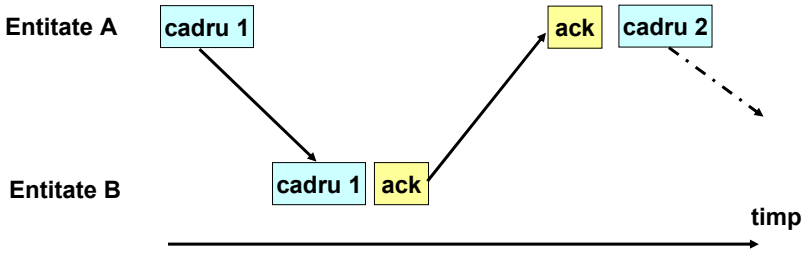
Ce erori pot fi detectate?

- Probabilitatea de detectie depinde de lungimea codului de control
- CRC si sume de control pe
  - 8 biti detecteaza 99.6094% din erori
  - 16 biti detecteaza 99.9985% din erori
  - 32 biti detecteaza 99.9999% din erori

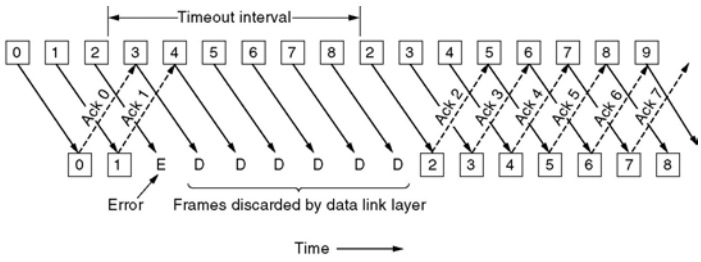
- In plus, CRC detecteaza 100% erori de
  - 1 bit;
  - 2 biti;
  - un numar impar de biti;
  - erori in rafala de lungimea codului CRC.

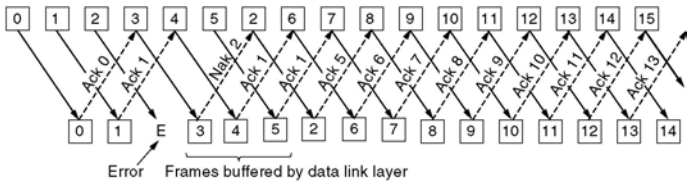
Protocoale elementare pentru legătura de date

Start-stop



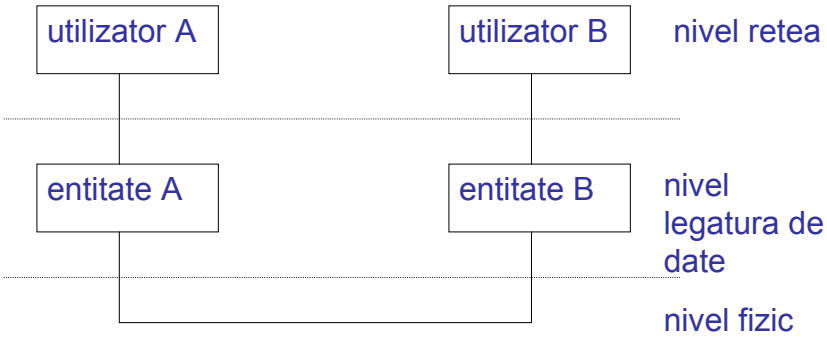
Ferestre glisante





Protocoalele 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 far* adresa;
        word lungime;
} pachet;
```

Primitivele de serviciu

- preluarea unui pachet de la retea pentru transmitere pe canal

```
pachet DeLaRetea();
```

- livrarea către retea a unui pachet

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

- trecerea unui cadru nivelului fizic pentru transmisie

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

- preluarea unui cadru de la nivelul fizic

```
cadru DeLaFizic();

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

TipEven wait();
```


Protocoale start-stop

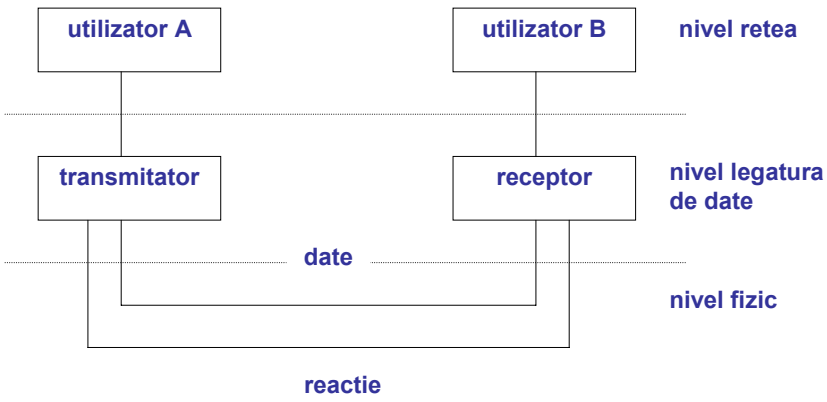
Protocol simplex fara restrictii

- 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 comunicaţie 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

Este nevoie de un ceas

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

si de numere de secventa - cadrele succesive m, m+1, m+2 au numerele de secvenţa respectiv 0, 1 si 0 (protocol cu bit alternat)

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

```
#define MaxSecv 1

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

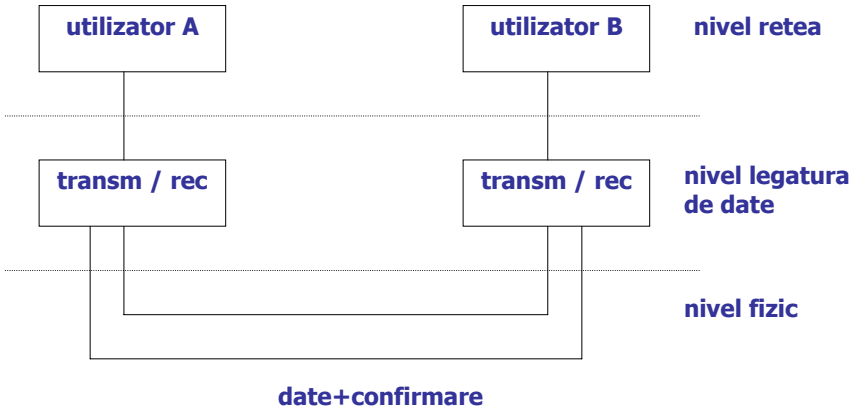
```
void transmit3() {
   NrSecv CadruUrmator=0;
   cadru s;
   TipEven even;
   s.info=DeLaRetea();
   do{
        s.secv=CadruUrmator;
        LaFizic(s);
        StartCeas(s.secv);
        even=wait();                  // poate fi SosireCadru,
                                      // TimeOut sau
                                      // 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;
}
```

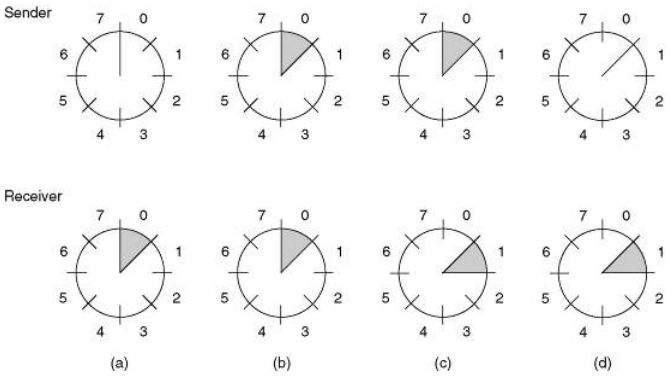
Protocoale cu fereastră glisantă

Protocol cu fereastră de dimensiune unu

Configuraţia



Protocoale cu fereastră glisantă



O fereastră de dimensiune 1, cu număr de secvență de 3 biți.

(a) Inițial.

(b) După transmiterea primului cadru.

(c) După recepția primului cadru.

(d) După recepția primei confirmări.

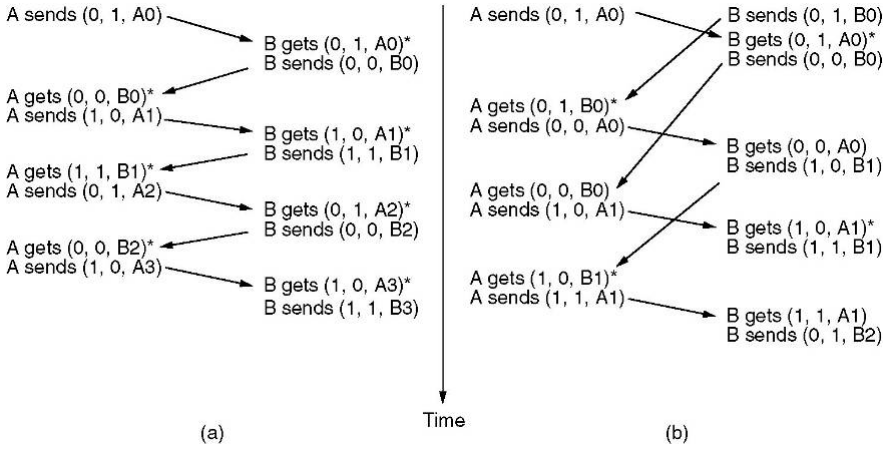
Fiecare staţie realizează ciclic următoarele operaţii:

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

```
void protocol4(){
   NrSecv CadruUrmator=0;
   NrSecv CadruAsteptat=0;
   cadru r,s;
   TipEven even;            //SosireCadru, TimeOut sau
                            //EroareControl
   s.info=DeLaRetea();
   s.secv=CadruUrmator;
   s.conf=1-CadruAsteptat;
   LaFizic(s);
   StartCeas(s.secv);
```


```
  do{
    even=wait();
    if(even==SosireCadru){
         r=DeLaFizic();
         if(r.secv==CadruAsteptat){      //prelucrare sir
                                         cadre receptionate
              LaRetea(r.info);
              inc(CadruAsteptat);
         }
         if(r.conf==CadruUrmator){       //prelucrare sir
                                         cadre transmisie
              StopCeas(r.conf);
              s.info=DeLaRetea();
              inc(CadruUrmator);
         }
    }
    s.secv=CadruUrmator;
    s.conf=1-CadruAsteptat;
    LaFizic(s);
    StartCeas(s.secv);
  }forever;
 }
```

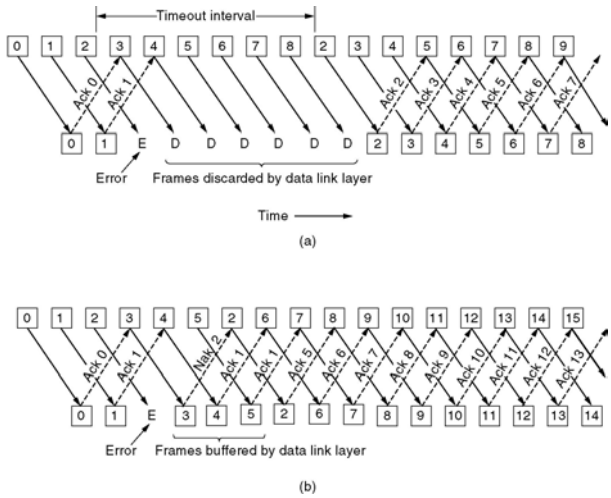
Un Protocol cu fereastră de un bit



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

- Notația este (seq, ack, packet number).
- Un asterisc arată că nivelul rețea acceptă pachetul.

Un protocol “Go Back N”



Banda de asamblare şi refacerea erorilor. Efectul erorii cand:

(a) Fereastra receptorului este 1.

(b) Fereastra receptorului este mai mare.

Protocoale cu fereastră supraunitară de transmisie

Protocol cu retransmitere neselectivă

Fereastra maxima a transmitatorului poate fi de `MaxSecv` cadre
Scenariu pentru `MaxSecv = 7`

1. Transmitatorul trimite cadrele 0..7;
2. Toate cadrele sînt receptionate si confirmate;
3. Toate confirmarile sînt 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
       CadruAsteptat,       //urmatorul cadru asteptat
       ConfAsteptata;       //cel mai vechi cadru neconfirmat
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;
}

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

```
void protocol5(){
   ActivRetea();
   CadruUrmator=0;
   CadruAsteptat=0;
   ConfAsteptata=0;
   ntampon=0;
   do{
      even=wait();

      switch(even){

         case ReteaPregatita:
                     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)){
                          ntampon--;
                          StopCeas(ConfAsteptata);
                          inc(ConfAsteptata);
                }
                break;
      case EroareControl: break;
      case TimeOut:
                CadruUrmator=ConfAsteptata;
                for(i=1;i<=ntampon;i++){
                      transmite(CadruUrmator);
                      inc(CadruUrmator);
                }
     }
     if(ntampon<MaxSecv) ActivRetea();
     else DezactivRetea();
  }forever;
 }
```

Protocol cu retransmitere selectiva

Fereastra receptorului nu poate fi egală cu cea a transmiţătorului

1. Transmitatorul trimite cadrele 0..6
2. Cadrele sunt receptionate si confirmate. Fereastra receptoruui devine 7, 0, 1, 2, 3, 4, 5
3. Toate confirmarile sunt pierdute (se strica sincronizarea intre transm si rec)
4. Transmitatorul retrimite cadrul 0 la time-out
5. Receptorul accepta drept cadru nou aceasta copie (cadrul 0) care se potriveste in fereastra sa; cere cadrul 7 dinaintea lui 0 (care lipseste)
6. Transmitatorul interpreteaza ca a trimis corect cadrele de la 0 la 6 si trimite 7, 0, 1, 2, 3, 4, 5
7. Receptorul accepta cadrele, cu exceptia lui 0, pentru care are deja un cadru receptionat. Ca urmare, ignora acest cadru nou, luînd în locul lui duplicatul cadrului 0 anterior.

```
void protocol6(){
  initializari_contoare;
  do{   even=wait();
        switch (even) {
                case ReteaPregatita:
                    accepta_salveaza_si_transmite_un_cadru;
                    break;
                case SosireCadru:  r=DeLaFizic();
                   if (r.fel == data){
                        transm_nak_daca_r_dif_de_cadru_asteptat;
                        accepta_cadru_daca_in_fereastra_receptie;
                        livreaza_pachetele_sosite;
                        actualizeaza_fereastra_receptie;
                   }
                   if (r.fel == nak) retransmite_cadru_cerut;
                   trateaza_confirmare_cadre_eliberind_buffere;
                   break;
                case EroareControl: transmite_nak; break;
                case TimeOut: retransmite_cadrul_corespunzator; break;
                case ReteaLibera: transmite_confirmare_ack;
        }
        activeaza_sau_dezactiveaza_nivel_retea;
  }forever;
}
```

Exemple Protocoale Data Link

- HDLC – High-Level Data Link Control
- Legatura de date in Internet

HDLC – procedura LAPB

HDLC este o familie de protocoale

Tipuri statii

primara	genereaza comenzi
secundara	genereaza raspunsuri
combinata	genereaza ambele, comenzi si raspunsuri

Tipuri legatura

balansata	cu doua statii conbinate
nebalansata	o statie primara, una sau mai multe secundare

Moduri de transfer

NRM - Normal Response Mode (legatura nebalansata)
ABM - Asynchronous Balanced Mode
ARM - Asynchronous Response Mode

Procedura **LAPB** (Link Access Protocol Balanced) coresponde une legaturi balansate cu statii 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	(a) Cadru de informatie
(b)	1	0	Type	P/F	Next (b) Cadru supervizor
(c)	1	1	Type	P/F	Modifier (c) Cadru nenumerotat

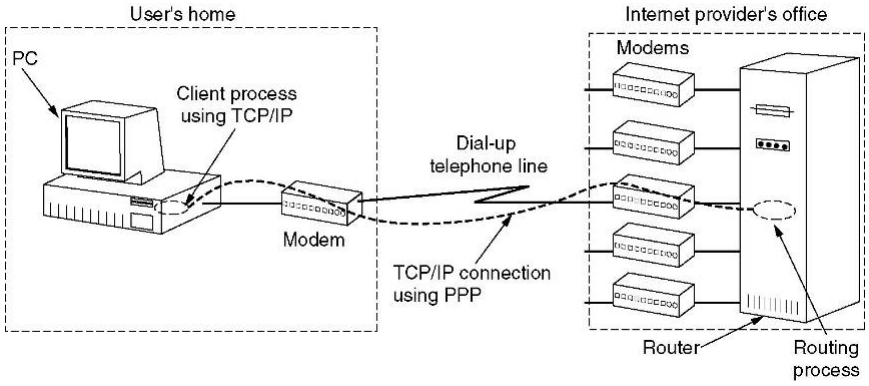
Atentie la semnificatie!
Seq – numar de secventa cadru transmis (mod 8 sau 128)
Next - numar de secventa urmatorul cadru asteptat
P/F – poll/final – invitatie la transmisie sau sfarsit de transmisie

Comenzi si raspunsuri

Comenzi	Raspunsuri
I = information	(suspended)
RR = receive ready	RR
RNR = receive not ready	RNR
REJ = reject	REJ
SABM = set asynchronous balanced mode	UA = unnumbered acknowledge
DM = disconnected mode	DISC = disconnect
	FRMR = frame reject

Legatura de date in Internet

Un calculator domestic actionand drept gazda Internet



Serial Link Internet Protocol – SLIP

- nu este standard Internet
- protocol de incadrare a pachetelor
- folosit pentru conexiuni seriale punct la punct peste care ruleaza TCP/IP intre gazde si rutere

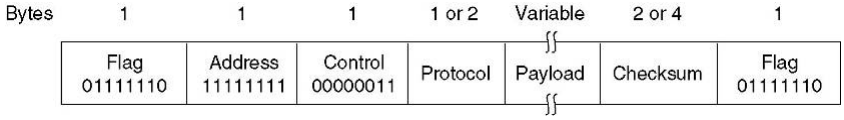
Reguli Protocol

- defineste doua caractere speciale: END si ESC
- o gazda SLIP trimite date in pachet
- END in pachet inlocuit cu ESC si octal 334
- ESC in packet inlocuit cu ESC si octal 335
- dupa ultimul octet din pachet se transmite END.

PPP – Point to Point Protocol

Ofera

incadrare
Link Control Protocol, LCP
Network Control Protocol, NCP



Format de cadru PPP pentru modul nenumerotat

Addresa 11111111 = toate statiile accepta cadrul

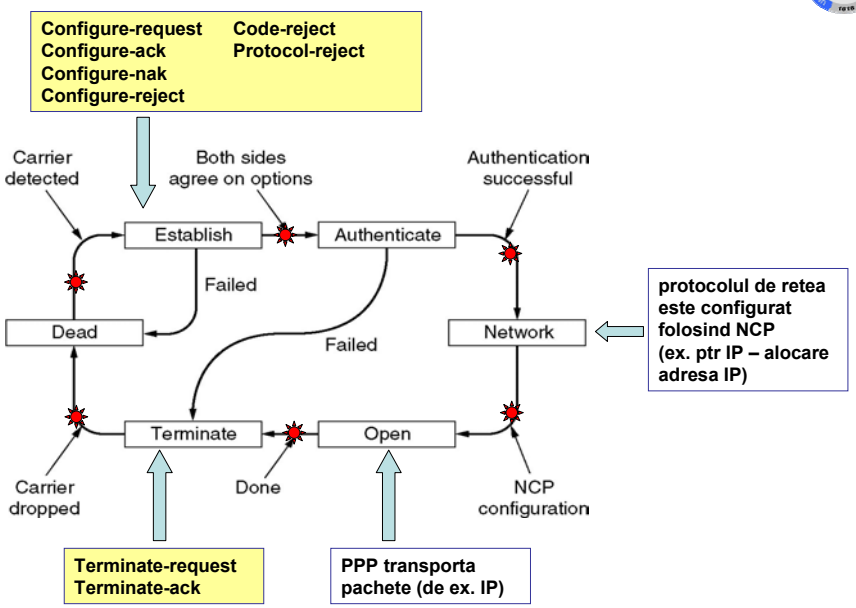
Control 00000011 = nenumerotat

Protocol = selecteaza dintre

LCP, NCP

IP, IPX, OSI CLNP, XNS

PPP – Point to Point Protocol (2)



Tipuri de cadre LCP

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

I - Initiator **R - Responder**

Sumar

- Funcțiile nivelului legătura de date (încadrare, transmisie transparenta, controlul erorilor, controlul fluxului, gestiunea legăturilor)
- Detecția şi corectarea erorilor (codul Hamming, coduri polinomiale - CRC)
- Protocoalele legăturii de date (date, functii, entitati)
- Protocol start-stop simplex fara restrictii / cu restrictii
- Protocol simplex pentru un canal cu erori
- Protocoale cu fereastră glisantă
  - Protocol cu fereastră de un bit
  - Protocol “Go Back N”
- Protocoale cu fereastră supraunitară de transmisie
- Protocol cu retransmitere selectiva
- Exemple Protocoale Legatura de date: HDLC
- Legatura de date in Internet
  - Serial Link Internet Protocol – SLIP
  - PPP – Point to Point Protocol