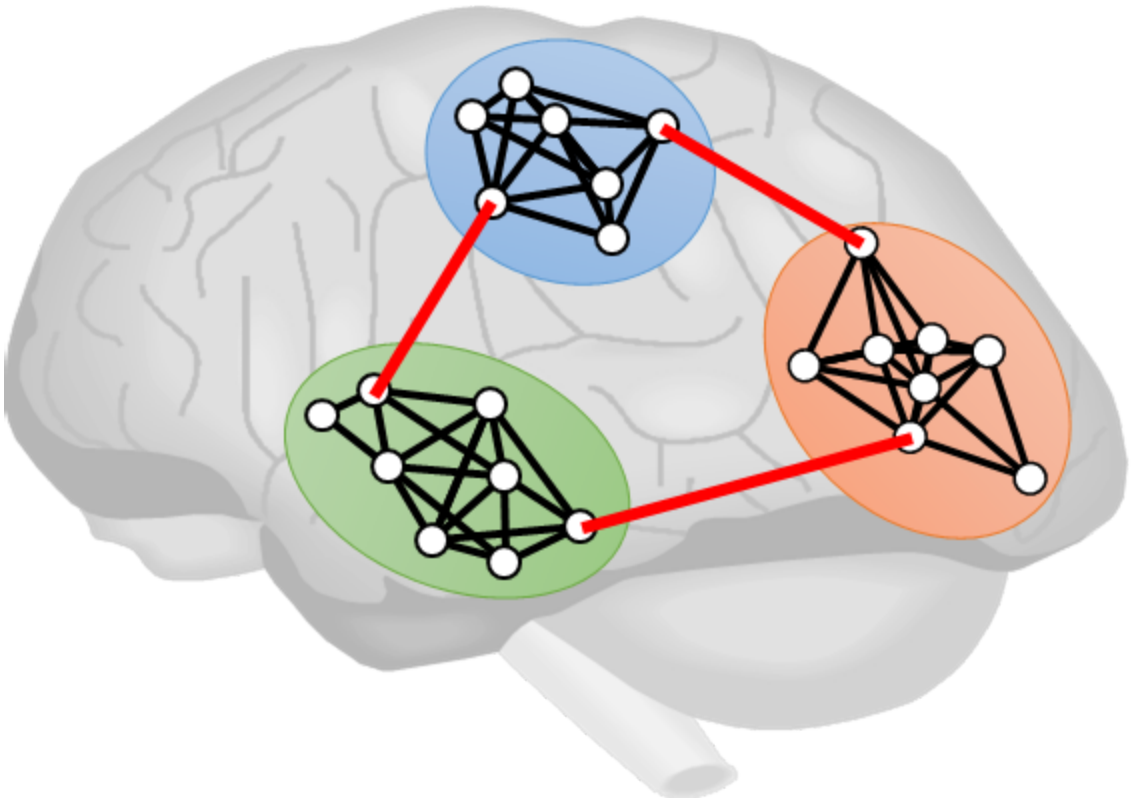
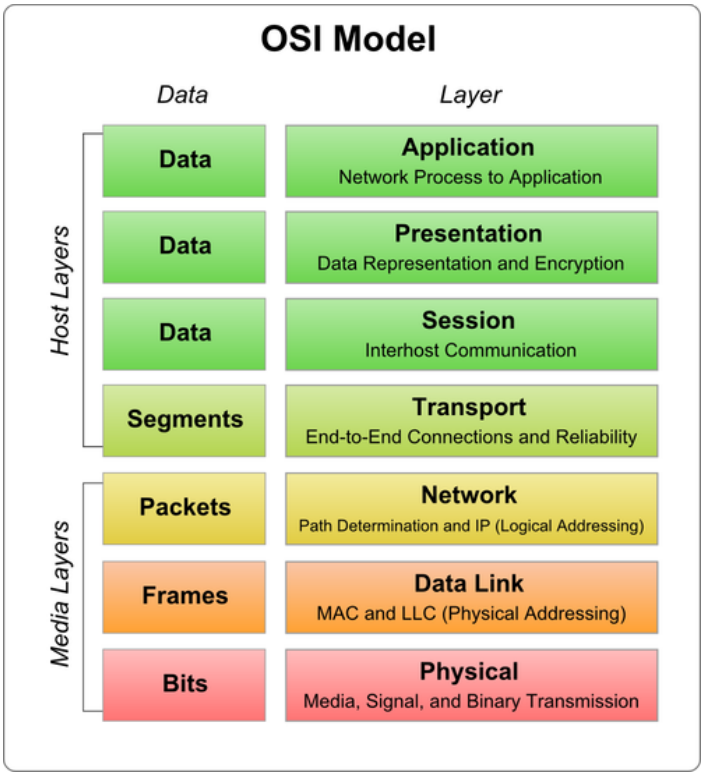
EXAMEN

PROTOCOALE DE COMUNICATII

IDEI PRINCIPALE



1. Nivelurile OSI (Open Systems Interconnection)



- Nivelul **fizic** se ocupă de transmiterea biților printr-un canal de comunicație.
  - Cati volti reprezinta un bit de 1, cati reprezinta un bit de 0.
  - E important ca un bit de 1 trimis sa fie primit ca un bit de 1.
- Nivelul **legaturii de date** trateaza erorile de transmisie produse la nivelul fizic, realizand o comunicare corecta intre doua noduri adiacente.
  - Impartirea sirului de biti in cadre, carora li se adauga informatii de control.
  - Detectia si corectia erorilor.
  - Cadrele se trimit individual si pot fi confirmate de catre receptor.
  - Controlul fluxului de date: sa nu se transmita date mai repede decat receptorul poate sa primeasca.
  - Gestiunea legaturii
    - Stabilirea legaturii
    - Controlul schimbului de date
    - Intreruperea legaturii

- Nivelul **retea** asigura dirijarea unitatilor de date intre nodurile sursa si destinatar, trecand eventual prin noduri intermediare.
    - Interconectarea retelelor cu arhitecturi diferite.
    - Se incearca sa se evite congestionarea retelei, facand sa nu fie legaturi supraincarcate si legaturi neutilizate.
- Nivelul **transport** realizeaza o comunicare sigura intre doua calculatoare gazda, detectand si corectand erorile pe care nivelul retea nu le trateaza.
    - Furnizeaza nivelelor superioare o interfata independenta de arhitectura utilizata
- Nivelul **sesiune** ofera toate serviciile pentru gestiunea jetoanelor, lasand la latitudinea utilizatorilor semnificatiile asociate acestora.
- Nivelul **prezentare** realizeaza transformari ale reprezentarii datelor, astfel incat sa se pastreze semnificatia lor, rezolvandu-se totodata diferentele de sintaxa.
- Nivelul **aplicatie**, cel mai inalt nivel al arhitecturii, are rolul de fereastra de comunicare prin care se fac toate schimburile de date intre utilizatori.

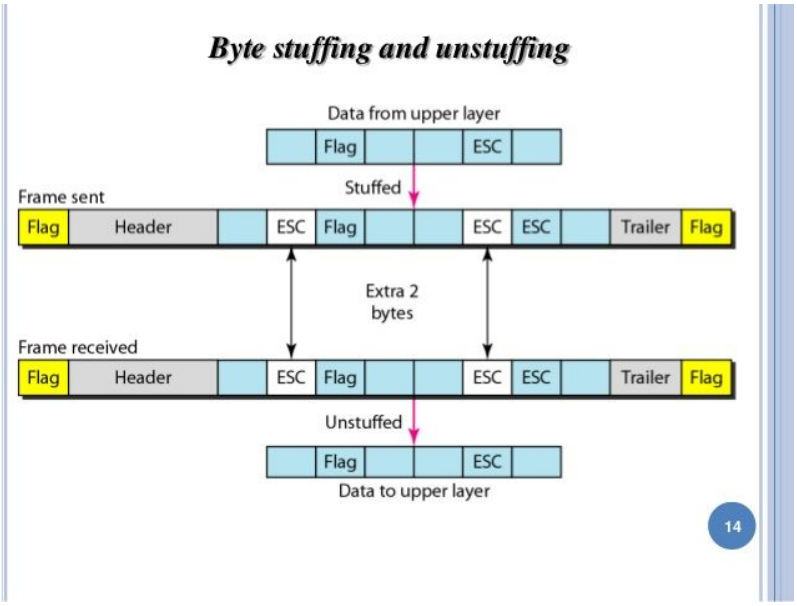
2. Nivelul Legatura de Date

- **Serviciu neconfirmat fără conexiune.**
    - mașina sursă trimite cadre independente către mașina destinație, fără ca mașina destinație să trebuiască să confirme primirea lor.
    - Utilizat cand rata de erori este foarte scăzută, iar recuperarea este lăsată în sarcina nivelurilor superioare
    - adecvat pentru traficul de timp real
    - Utilizata de majoritatea LANurilor
- **Serviciu confirmat fără conexiune.**
    - fiecare cadru trimis este confirmat individual.
    - Ar putea fi rezolvata pierderea la nivelul retea, dar e mai putin costisitor sa retrimiti cadre (relativ mici) decat pachete mari (cum sunt la nivel retea)
    - Aplicabil pe canale fara fir (nesigure), neaplicabil la fibra optica (canal sigur)
- **Serviciu confirmat orientat-conexiune.**
    - mașinile sursă și destinație stabilesc o conexiune înainte de a transfera date
    - Cadrele sunt numerotate
    - Fiecare cadru trimis este sigur receptionat o singura data

- Cadrele sunt receptionate in ordine
- **3 faze:**
  - Stabilirea conexiunii + initializare contoare & variabile
  - Transmiterea cadrelor
  - Desfiintarea conexiunii

3. Incadrarea la nivelul legatura de date

- **Numărarea caracterelor.**
  - utilizează un câmp din antet pentru a specifica numărul de caractere din cadru.
  - Problema: contorul poate fi alterat de erori de transmisie => nu stie unde incepe urmatorul cadru
- **Indicatori cu inserare de octeți.**
  - fiecare cadru începe și se termină cu o secvența specială de octeți.
  - *octet indicator*, atât ca indicator de început, cât și de sfârșit.
  - *Problema:* trimiterea datelor binare sau numere in virgula mobila
  - *Rezolvare*: se insereaza caracterul ESC in fata fiecarei aparitii a indicatorului in date (byte stuffing)
  - *Dezavantaj*: limiteaza la utilizarea caracterelor de 8 biti



- **Indicatori de început și de sfârșit, cu inserare de biți.**
  - fiecare cadru începe și se termină cu un șablon special pe biți, 01111110, numit octet indicator (flag)
  - In date, dupa oricare 5 biti de 1 se insereaza un 0 (care se scoate la primire)
  - complet transparentă pentru nivelul rețea

Figure 11.3 *A frame in a bit-oriented protocol*

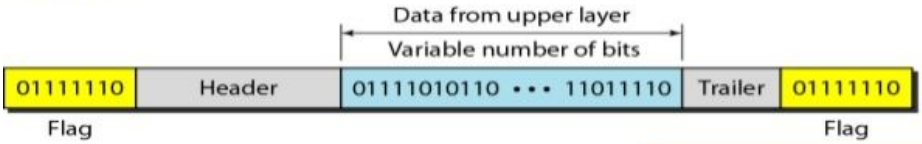
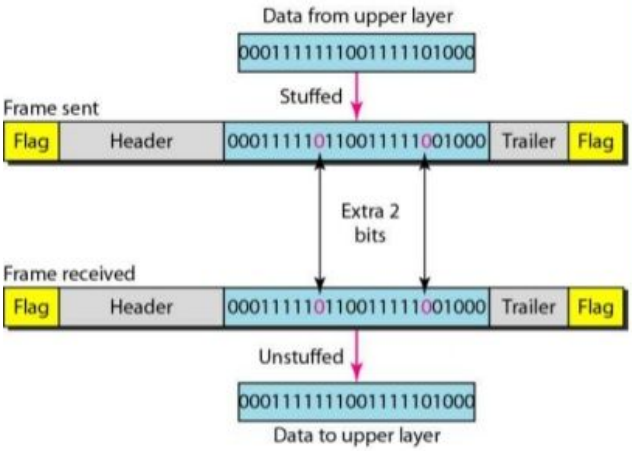


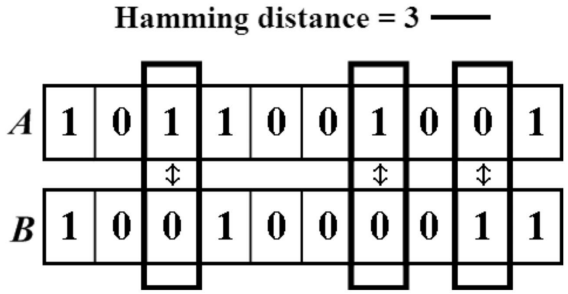
Figure 11.4 *Bit stuffing and unstuffing*



Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

4. Coduri corectoare de erori

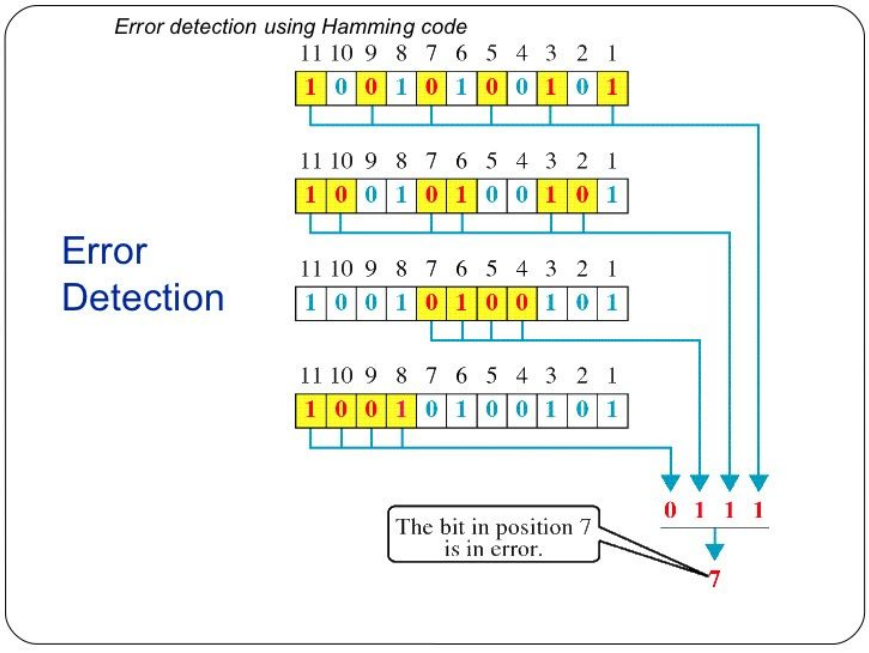
- pe lângă fiecare bloc de date trimis să se includă suficientă informație redundantă pentru ca receptorul să poată deduce care a fost caracterul transmis
- *forward error correction*
- n-bit codeword = m biti de date + r biti de informatie redundanta
- distanță Hamming *d* <=> sunt necesare *d* erori de un singur bit pentru a converti un cuvant de cod in celalalt
- Corectare *d* erori -> necesar cod cu distanta *2d + 1*



Distanta Hamming = numărul de poziții binare în care două cuvinte de cod diferă; aplicăm operatorul XOR între cele două cuvinte de cod și numărăm biții 1 din rezultat.

METODA HAMMING

- Biții cuvântului de cod sunt numerotați consecutiv, începând cu bitul 1 de la marginea din stânga.
- Biții care sunt puteri ale lui 2 (1, 2, 4, 8, 16 etc.) sunt **biți de control**.
- Restul (3, 5, 6, 7, 9 etc.) sunt completați cu cei m **biți de date**.
- Fiecare bit de control forțează ca paritatea unui grup de biți, inclusiv el însuși, să fie pară (sau impară).
- Codurile Hamming pot corecta numai **erori singulare**.



ERORI IN RAFALA

- secvență de k cuvinte de cod consecutive este aranjată ca o **matrice**, având câte un cuvânt de cod pe fiecare linie.
- la receptor, matricea este reconstruită, coloană cu coloană

5. Coduri detectoare de erori

- Folosite in cazul firelor de cupru sau a fibrei optice, unde rata de erori e atat de mica incat retransmisia e mai eficienta
- detectare *d* erori -> necesar cod cu distanta Hamming *d + 1*

BIT DE PARITATE

- **Problema:** daca se adauga un singur bit de paritate si blocul este afectat de o eroare in rafala, probabilitatea ca acea eroare sa fie detectata este de 0.5
- **Solutie:** fiecare bloc este privit ca o **matrice**; se calculeaza bitul de paritate pentru fiecare coloana si se adauga ulterior ca linie la sfarsitul blocului. Se transmite ulterior blocul linie cu linie
- Daca oricare din bitii de paritate este gresit, se cere *retransmiterea intregului bloc*
- Se pot cere **retransmiteri succesive** pana cand blocul este receptionat corect
- Probabilitatea ca un bloc eronat să fie acceptat atunci când nu ar trebui este 2^{-n}, unde n = lungimea rafalei

CODUL POLINOMIAL

- Des utilizata in practica
- A.k.a *cod de redundanta ciclica* = **CRC**
- Tratarea șirurilor de biți ca reprezentări de polinoame cu coeficienți 0 și 1
- Bitul cel mai semnificativ (cel mai din stânga) este coeficientul lui x^{k-1}
- **Ex:** 110001 = $x^5+x^4+x^0$
- Adunările și scăderile sunt identice cu XOR
- emițătorul și receptorul se pun de acord în avans asupra unui polinom generator G(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).
- Acele erori care se întâmplă să corespundă unor polinoame care îl au ca factor pe G(x) vor scăpa; toate celelalte vor fi detectate.
- **Algoritmul pentru calculul sumei de control**
  - Fie r gradul lui G(x). Se adaugă r biți 0 la capătul mai puțin semnificativ al cadrului
  - Se împarte șirul de biți ce corespund lui G(x) într-un șir de biți corespunzând lui xrM(x), utilizând împărțirea modulo 2.
  - Se scade restul. Rezultatul este cadrul cu sumă de control ce va fi transmis.

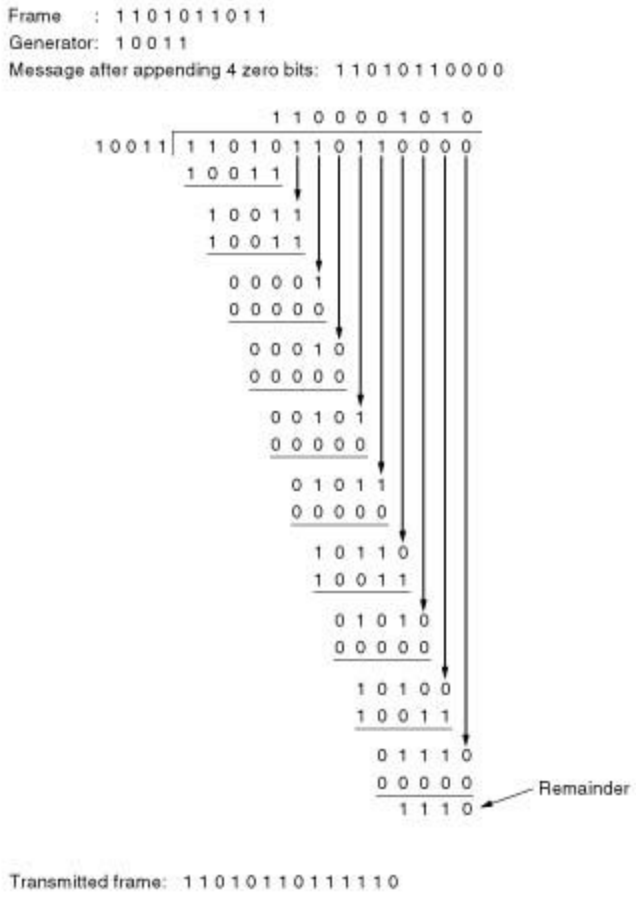
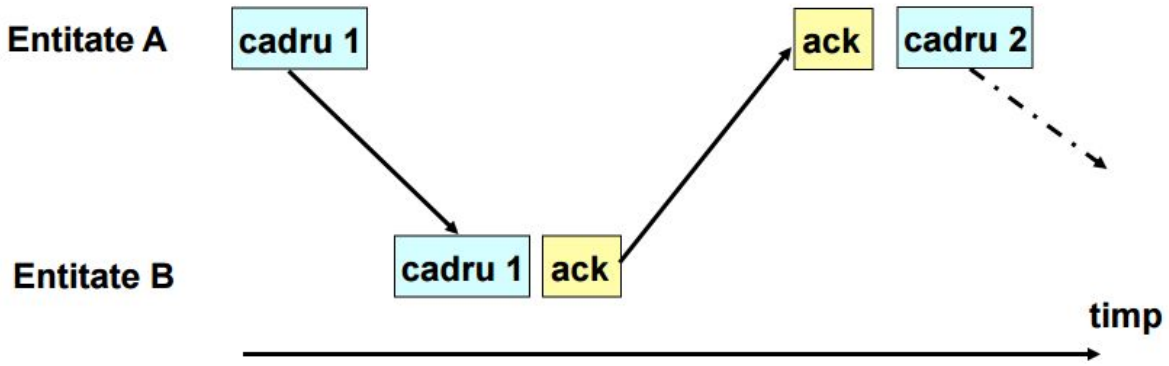


Fig. 3-7. Calculation of the polynomial code checksum.

6. Protocoalele elementare ale legaturii de date

START-STOP

- transmițătorul trimite un nou cadru numai după recepția confirmării pozitive a cadrului precedent.



PROTOCOLUL SIMPLEX FARA RESTRICTII

- A transmite catre B
- A are o sursa inepuizabila de date
- B este un consumator ideal
- Canal fizic de comunicatie lipsit de erori

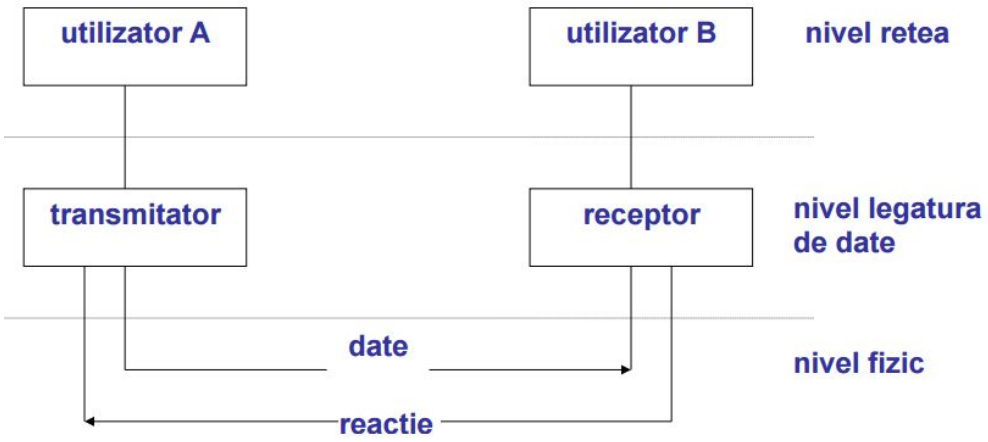
```
// entitatea din sistemul transmitatorului
```

```
void transmit1() {
      cadru s;
      do {
            s.info = DeLaRetea(); //preia pachet
            LaFizic(s); //transmite cadru
      } while(1);
 }
```

```
// entitatea din sistemul receptorului
void recept1() {
      cadru r;
      TipEven even;
      do {
            even = wait(); //asteapta cadru
            r = DeLaFizic(); //primeste cadru
            LaRetea(r.info); //preda pachet
      } while(1);
}
```

PROTOCOLUL SIMPLEX START-STOP

- canalul fără erori
- utilizatorul B nu poate accepta date în orice ritm.
- Necesar controlul fluxului
- B transmite un cadru fictiv (reactia receptorului catre transmitator)




```
// entitatea din sistemul transmitatorului
void transmit2() {
      cadru s;
      TipEven even;
      do {
            s.info = DeLaRetea(); //preia pachet
            LaFizic(s); //transmite cadru
            even = wait(); //asteapta permisiunea
      } while(1);
 }
```

```
// entitatea din sistemul receptorului
```

```
void recept2() {
      cadru r, s;
      TipEven even;
      do {
            even = wait(); //asteapta cadru
            r = DeLaFizic(); //primeste cadru
            LaRetea(r.info); //preda pachet
            LaFizic(s); // deblocheaza
      } while(1);
}
```

PROTOCOLUL SIMPLEX PENTRU UN CANAL CU ERORI

- Receptorul trimite un cadru de confirmare doar pentru cadrele primite corect
- Daca transmitatorul nu primeste confirmarea dupa un anumit **timp**, retrimite pachetul
- **Problema:** duplicate daca se pierde confirmarea mesajului
- **Solutie:** se include numarul de secventa in cadru; este suficient sa fie un bit (0/ 1) (*PROTOCOL CU BIT ALTERNAT*)

```
// entitatea din sistemul transmitatorului
```

```
void transmit3() {
      NrSecv CadruUrmator=0;
      cadru s;
      TipEven even;
      s.info = DeLaRetea();
      do {
            s.secv=CadruUrmator; //adauga nr secventa
            LaFizic(s);
            StartCeas(s.secv);
            even=wait(); // SosireCadru,TimeOut sau Eroarecontrol
            if(even==SosireCadru) { //confirmare intacta
                  StopCeas(s.secv);
                  s.info=DeLaRetea();
                  inc(CadruUrmator);
            }
      } while(1);
 }
```

```
// entitatea din sistemul receptorului
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
            }
      } while(1);
}
```

PROTOCOL CU FEREASTRA GLISANTA

- nu așteaptă confirmarea cadrelor precedente pentru a transmite cadre noi.
- Pentru o transmisie duplex putem utiliza cadrele de confirmare pentru transmiterea datelor în sens opus.
- multe feluri de cadre, diferențiabile printr-un cîmp din antet
- **Protocol cu numar de secventa de un bit**
  - (seq, ack, message)
  - **Caz anormal**: A si B transmit in acelasi timp

```
// comun pentru transmitator si receptor
void protocol4() {
      NrSecv CadruUrmator = 0; // 0 sau 1
      NrSecv CadruAsteptat = 0; // 0 sau 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
        StartCeas(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);
                        }
                }
                //construiteste si transmite un nou cadru
                s.secv = CadruUrmator;
                s.conf = 1 - CadruAsteptat;
                LaFizic(s);
                StartCeas(s.secv);
        } forever;  //reia de la asteptarea unui cadru
}
```

❑ FEREASTRA GLISANTA

- ❑ Integreaza controlul erorilor si controlul fluxului intr-un mod convenabil
- ❑ fereastra este un **sub-şir de numere de secvenţă**
- ❑ pe parcursul transmiterii cadrelor, fereastra **glisează**
- ❑ la transmitator, fereastra contine numerele cadrelor transmise si ne-confirmate
- ❑ dimensiunea ferestrei **transmitatorului** este **variabila**
  - ❑ creste cand se trimite un nou cadru;
  - ❑ scade cand se primeste o confirmare;
- ❑ la **receptor**, fereastra specifica numerele cadrelor ce pot fi acceptate

- dimensiunea ferestrei **receptorului** este **constanta**
- fereastra gliseaza cand unul sau mai multe cadre din stanga ferestrei sunt livrate utilizatorului

PROTOCOL CU TRANSMITERE NESELECTIVA "GO BACK N"

- Fereastra maximă a transmițătorului poate fi de MaxSecv cadre, deși există MaxSecv+1 numere de secvență distincte; daca ar fi MaxSecv + 1 si s-ar pierde toate cadrele, la retransmitere, dupa timeout, receptorul ar accepta duplicatele
- Functii:
  - *ReteaPregatita:* cand utilizatorul are un pachet de trimis
  - *DezactivRetea:* MaxSecv cadre neconfirmate
  - *ActivRetea:* dupa ce se mai confirma din cadre
- Confirmarea cadrului n provoacă automat confirmarea cadrelor n-1, n-2,... anterioare (compenseaza cu pierderea confirmarilor)
- se utilizează ceasuri separate pentru diferitele cadre
- La Timeout, toate cadrele din buffer sunt retransmise
- Se utilizeaza **banda de asamblare**
- **Carte:** receptorul elimina cadrele care urmeaza, netrimitand confirmari pentru cadrele eliminate. El refuza sa accepte orice cadru exceptandu-l pe urmatorul care trebuie livrat catre nivelul retea. Asteapta astfel pana la timeoutul din fereastra emitatorului ca sa isi primeasca cadrul dorit si cadrele care urmeaza dupa acesta.

RETRANSMITERE SELECTIVA (Selective repeat) :

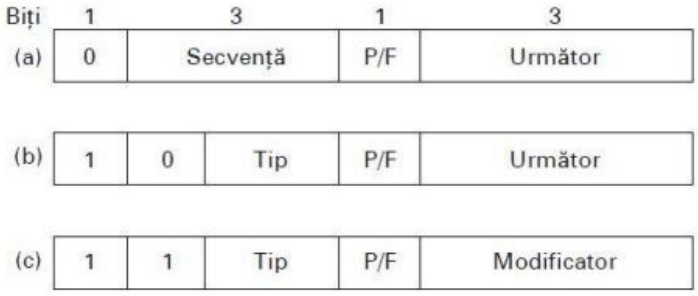
- Fereastra receptorului nu poate fi egală cu cea a transmițătorului (ferestrele succesive ale receptorului au numere de secvență comune)
- fereastra receptorului trebuie să fie **cel mult jumătate** din gama numerelor de secvență.
- un cadru incorect este respins, dar toate cadrele corecte care il urmeaza sunt memorate.
- La timeout, cel mai vechi cadru neconfirmat este transmis
- Uneori se pot transmite **confirmari negative** (NAK) - simuleaza retransmisia inainte de expirarea timeoutului

- Daca fereastra e mare, poate necesita un **spatiu mare de memorie**

7. HDLC

- orientate pe biți
- Folosesc inserarea de biti pentru transparenta datelor
- folosește o fereastră glisantă, cu un număr de secvență reprezentat pe 3 biți.

- ADRESA
  - pentru liniile cu terminale multiple, folosit pentru a **identifica terminalul**
  - Pentru liniile punct-la-punct, este folosit pentru a **deosebi comenzile de raspunsuri**
- CONTROL
  - Folosit pentru numere de secventa, confirmari, etc



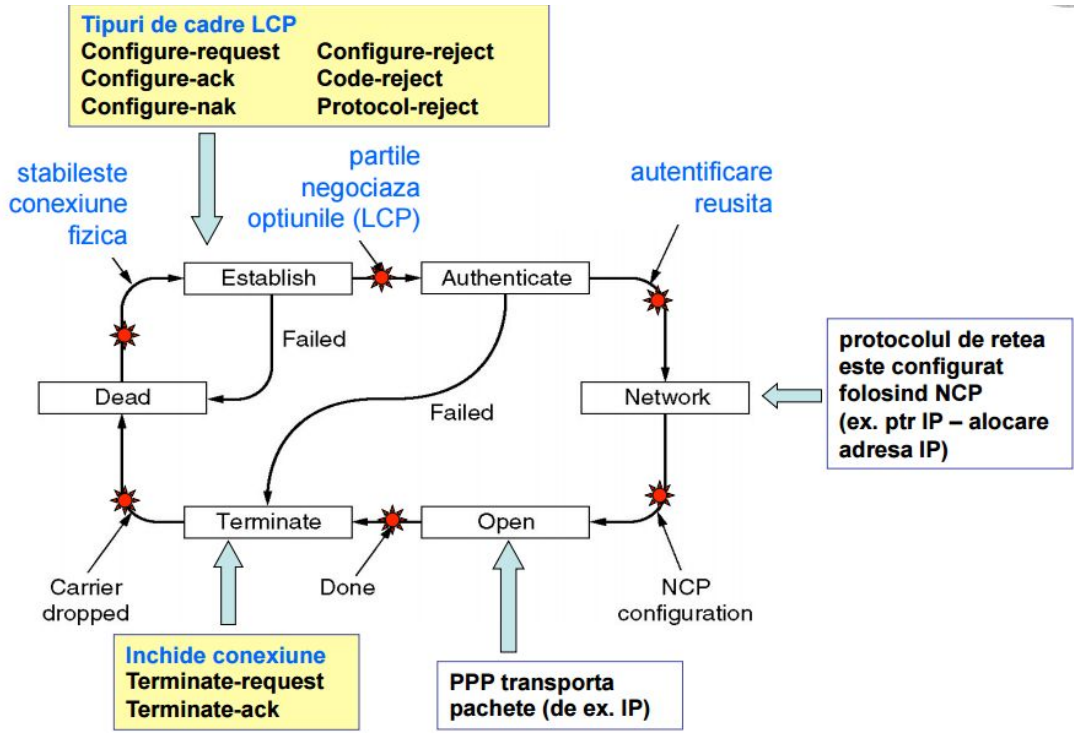
  - **Informatie** (a)
    - Secventa: numarul de secventa al cadrului
    - Urmator: numarul cadrului asteptat (uneori se foloseste ultimul cadru receptionat corect)
    - P/F (Poll/ Final): folosit la interogarea unui grup de terminale;
      - in *comenzi*, P = invitatie la transmisie

- in *raspunsuri* toate cadrele au P, ultimul are F
- **Supervizor** (b)
  - Tipul 0 - cadru de confirmare (RECEIVE READY); folosit atunci când nu există flux invers
  - Tipul 1 - confirmare negativă (REJECT); indica detecția unei erori de transmisie
  - Tipul 2 - RECEIVE NOT READY; spune transmițătorului să oprească transmisia.
  - Tipul 3 - SELECTIVE REJECT; cere retransmiterea doar pentru cadrul specificat
- **Nenumerotat** (c)
  - folosit uneori în scopuri de control
  - poate fi folosit și pentru transportul datelor atunci când se recurge la un **serviciu nesigur, neorientat pe conexiune**

- DATE
  - Contine informatii arbitrare de lungimi arbitrare
  - Eficienta sumei de control scade cu cresterea lungimii
- SUMA DE CONTROL
  - Varianta a CRC
- SECVENTA INDICATOR 01111110
- **Comenzi:**
  - DISC (DISConnect) - masina anunta ca se va opri
  - SNRM (Set Normal Response Mode) - masina se reconecteaza, reseteaza la 0 nr de secventa; asincron
  - SABM (Set Asynchronous Balanced Mode) - reseteaza numetele de secventa si declara ambii parteneri ca fiind egali
  - SABME și SNRME - identice cu cele de mai sus; format extins pentru cadru (nr de secventa pe 7 biti)
- UA (Unnumbered Acknowledgement) - cadru special de control pentru confirmarea cadrelor de control
- UI (Unnumbered Information) - cadru de control care contine informatii arbitrare
- Cadrele de control nu sunt livrate nivelului rețea, ci sunt destinate a fi primite chiar de nivelul legătură de date.

PPP – Point to Point Protocol

- Trafic ruter - ruter / trafic utilizator - ISP
- Principalul protocol al legaturii de date pe liniile de tip punct-la-punct
- Face **detectia** erorilor
- Termite negocierea adreselor IP in momentul conectarii
- Permite autentificarea
- Permite incadrarea fara ambiguitati
- **Control al legaturii** pentru a obtine liniile: **LCP**; suporta circuite sincrone si asincrone
- Negocierea optiunilor nivelului retea in mod independent de protocolul nivelului retea: **NCP**
- Fara de HDLC, este, mai degraba, orientat pe caractere decat pe biti
- Cadrele PPP pot fi transmise atat pe linii telefonice comutate, cat si pe linii HDLC sau SONET



8. Servicii furnizate de nivelul retea nivelului transport

1. Serviciile trebuie să fie **independente de tehnologia ruterului**.

2. Nivelul transport trebuie să fie **independent** de **numărul, tipul și topologia ruterelor** existente.

3. **Adresele de rețea** disponibile la nivelul transport trebuie să folosească o **schemă de numerotare uniformă,** chiar în cadrul rețelelor LAN și WAN.

- 2 tabere:
  - Internet: nivelul retea trebuie doar sa transfere pachete (SEND PACKET, RECV PACKET + alte chestii minore); reteaua se presupune inerent nesigura si calculatoarele gazda trebuie sa faca controlul erorilor ele insele; neorientat pe conexiune
  - Companiile de telefoane: propun un serviciu orientat pe conexiune, in care nivelul retea sa realizeze controlul erorilor; se pune accent pe calitatea serviciilor

9. Subretea datagrama si subretea cu circuite virtuale

1) **Serviciu neorientat conexiune ⇔ datagrame**

Pachetele sunt trimise individual in retea si dirijate independent unul de celalalt. Nu este necesara nicio initializare prealabila. Daca un mesaj este mai mare decat dimensiunea maxima a unui pachet, el este spart in mai multe pachete care sunt transmise individual.

Fiecare router are o tabela interna care ii spune unde sa trimita pachete pentru fiecare destinatie posibila. Cand pachetele ajung la un router, sunt memorate temporar pentru a se verifica suma de control si sunt transmise mai departe. Pachetele in care e spart mesajul pot urma cai diferite. Algoritmul care administreaza tabelele de rutare se numeste algoritm de rutare.

2) **Serviciu orientat conexiune ⇔ circuite virtuale**

Se utilizeaza o retea de circuite virtuale (prin analogie cu liniile de telefonie). Toate pachetele in care este spart un mesaj urmeaza aceeasi cale (acelasi circuit virtual). Initial se alege o cale intre sursa si destinatie si este memorata in tabelele ruterelor. Cand conexiunea este eliberata, se inchide si circuitul virtual. Fiecare pachet poarta un identificator care determina carui circuit virtual ii apartine. Se asociaza un identificator de conexiune pe care ruterele trebuie sa fie capabile sa il modifice.

Problemă	Subrețea datagramă	Subrețea cu circuite virtuale (CV)
Stabilirea circuitului	Nu este necesară	Obligatorie
Adresare	Fiecare pachet conține adresa completă pentru sursă și destinație	Fiecare pachet conține un număr mic de CV
Informații de stare	Ruterele nu păstrează informații despre conexiuni	Fiecare CV necesită spațiu pentru tabela ruterului per conexiune
Dirijare	Fiecare pachet este dirijat independent	Calea este stabilită la inițierea CV; toate pachetele o urmează
Efectul defectării ruterului	Nici unul, cu excepția pachetelor pierdute în timpul defectării	Toate circuitele virtuale care trec prin ruterul defect sunt terminate
Calitatea serviciului	Dificil	Simplu, dacă pentru fiecare CV pot fi alocate în avans suficiente resurse
Controlul congestiei	Dificil	Simplu, dacă pentru fiecare CV pot fi alocate în avans suficiente resurse

10. Algoritmi de dirijare

Principala functie a nivelului retea este dirijarea pachetelor.

Algoritm de dirijare = alegerea liniei de iesire pe care trebuie trimis un pachet receptionat

Dirijare = completarea si actualizarea tabelei de dirijare

Retransmitere = preia un pachet, cauta in tabela de dirijare si il transmite mai departe conform acesteia

Proprietati:

1) Corectitudine
2) Simplitate

3) Robustete - e necesar sa functioneze chiar daca se defecteaza o parte componenta (ex. Un router din retea) si sa nu fie necesara reinitializarea retelei la fiecare schimbare de topologie.
4) Stabilitate - algoritmii trebuie sa convearga catre o stare de echilibru
5) Echitate - toate cererile sa fie tratate la fel
6) Optimalitate - dirijarea sa se desfasoare optim

Compormis intre echitate si optimalitate: se cauta minimizarea numarului de salturi.

Principiul optimalitatii: Daca ruterul J este pe calea cea mai buna de la I la K, atunci calea optima de la J la K este pe aceeasi ruta.

Multimea rutelor de la o anumita sursa la toate destinatiile formeaza un arbore (sink tree).

ALGORITMI NEADAPTIVI (STATICI)

- **Dirijarea pe calea cea mai scurta (shortest path routing)**
  - **Metrici:**
    - Numarul de salturi
    - Distanta minima in km
    - Valori medii de asteptare in coada
    - Intarzieri de transmisie
    - Latime de banda
  - **Algoritmi:**
    - Dijkstra + everything about it, inclusiv pseudocod
- **Inundarea (flooding)**
  - Fiecare pachet este transmis mai departe pe toate liniile, mai putin pe cea de pe care tocmai a venit
  - PROBLEMA: numar mare de pachete duplicate, chiar infinit
  - SOLUTII:
    - Pastrarea unui **contor de salturi** - un pachet este distrus cand contorul ajunge la 0; contorul trebuie initializat cu lungimea drumului de la sursa la destinatie sau, daca acesta nu este cunoscut, cu diametrul retelei

- Identificarea pachetelor care au fost deja inundate pentru a nu mai fi retransmise (numar de secventa)

- **Selective flooding:** daca tu vrei ca pachetul tau sa ajunga in est, nu-l trimite si pe drumurile care merg spre vest
- Inundarea alege intotdeauna calea cea mai scurta, pentru ca incearca in paralel toate caile
- Se genereaza o supraincarcare a retelei

ALGORITMI ADAPTIVI (DINAMICI)

- **Dirijare cu vectori de distanta**
  - Fiecare ruter mentine un vector cu cea mai buna distanta cunoscuta catre toate destinatiile si linia care trebuie urmata pentru a ajunge acolo
  - Algoritmi:
    - Bellman-Ford
    - Ford-Fulkerson
  - Se presupune ca fiecare ruter cunoaste distanta catre vecinii sai
  - Metrica salturilor: distanta catre fiecare vecin este 1
  - Metrica intarzierii (ms) : se masoara prin transmiterea de pachete ECHO
  - Fiecare ruter primeste aproximari de la celelalte rutere si isi contruieste propriile aproximari pe baza acelora
  - **Problema numararii la infinit**
    - Vestile bune circula repede (intr-o retea in care calea cea mai lunga are N noduri, va dura N schimburi pana cand informatiile sunt actualizate)
    - Vestile proaste circula greu (atunci cand un router este scos in functiune, routerele conectate direct la acesta (notate B) vor considera ca nu stiu cale catre el. Totusi, routerele mai indepartate vor afirma ca ele cunosc o cale catre routerul scos din functiune si o trimit catre B; aceasta cale trece de fapt prin B). Pana cand toate ruterele isi vor da seama ca nu mai exista cale catre ruterul scos din functiune for trece INF pasi; astfel, este recomandat ca INF sa fie codificat ca lungimea celei mai mari cai + 1

- SOL: **split horizon**: noile distante nu se trimit vecinului prin care trec actualele rute – B are ruta de distanta 2 catre E prin A – B nu include noua distanta catre E in actualizarea trimisa lui A
- SOL: **split horizon with poison reverse:** trimite o valoare f. mare – B trimite distanta ∞ catre A – A nu va mai alege o cale prin B

- **Dirijarea folosind starea legaturilor**
  - Fata de algoritmul anterior, in care metrica cel mai frecvent utilizata era numarul de pachete aflate in coada de asteptare, acum se ia in calcul si latimea de banda
  - Fiecare router trebuie sa faca urmatoarele:
    - Sa descopere care sunt vecinii sai si sa afle adresele de retea ale acestora
      - Se transmite un pachet special HELLO pe toate liniile;
      - routerele raspund cu un pachet in care isi anunta identitatea;
      - Identitatea trebuie sa fie unica global
    - Sa masoare intarzierea sau costul pana la fiecare vecin
      - Se transmite un pachet special ECHO pe toate liniile conectate
      - Routerele care il primesc il transmit imediat inapoi
      - Se poate considera sau nu incarcarea retelei (numarul de pachete aflate in coada)
        - Daca se tine cont, atunci reteaua oscileaza puternic
    - Construirea pachetelor cu starea legaturilor
      - Pachetul contine identitatea expeditorului, varsta si lista vecinilor, cu intarzierile catre acestia
      - Ele pot fi create si transmise periodic sau atunci cand se produce o schimbare majora in retea
    - Distribuirea pachetelor cu starea legaturilor
      - Se utilizeaza **inundarea**
      - Fiecare pachet contine un numar de secventa care este incrementat la fiecare pachet transmis; el e cautat in lista

pachetelor deja vazute si, daca este nou, este transmis mai departe

- **PROBLEME**:
  - Daca un router se defecteaza, pierde numarul de secventa
  - Daca numerele ajung la valoarea maxima acceptata si sunt reluate de la 0, pot aparea confuzii
  - Pot aparea erori de transmisie care sa modifice un bit in numarul de secventa
- **SOLUTII :**
  - Includerea **varstei** si decrementarea acesteia la fiecare salt

- Calcularea noilor rute
  - Cand un ruter acumuleaza un set complet de pachete cu starea legaturlor, el poate construi graful intregii subretele
  - Se aplica local Dijkstra

- **Dirijarea ierarhica**
  - Reteaua se imparte in regiuni, regiunile pot fi grupate in clustere si tot asa;
  - Pentru o retea de N routere, numarul optim de niveluri este de ln N
- **Dirijarea prin difuzare (broadcasting)**
  - Folosita atunci cand calculatarele au nevoie sa transmita mesaje catr mai multe sau catre toate celelalte calculatoare gazda (gen rapoarte meterologice)
  - Metode posibile:
    - Trimiterea cate unui pachet diferit catre fiecare destinatie; consuma multa latime de banda si cere ca sursa sa detina o lista a tuturor destinatiilor
    - Inundarea; are ca dezavantaj faptul ca genereaza foarte multe pachete si consuma latime de banda prea mare
    - Dirijarea multidestinatie;
      - O linie de iesire este selectata doar daca reprezinta cea mai buna cale catre cel putin o destinatie

- atunci cand mai multe pachete trebuie sa urmeze aceeasi cale, unul dintre ele plateste tot drumul si celelalte calatoresc gratis
- Arbore de acoperire: daca un ruter cunoaste care dintre liniile sale fac parte din arborele de acoperire, este suficient sa transmita mesajul doar pe liniile respective
- Transmiterea pe cale inversa: daca un pachet a ajuns acolo pe calea pe care ajung de obicei pachete de la sursa difuzarii, atunci probabil drumul este minim si acesta este primul exemplar, asa ca il transmite mai departe; altfel, il elimina; algoritmul e eficient si usor de implementat

- **Dirijarea cu trimitere multipla (multicast)**
  - Atunci cand vrei sa transmiti la grupuri
  - Nu se poate folosi inundarea deoarece e super ineficient sa transmiti la toti si informatiile pot fi senzitive
  - Necesita managementul grupului: fiecare proces se ataseaza unui grup si el trebuie sa informeze gazda.
  - Ruterele trebuie sa stie caror grupuri le apartim calculatoarele gazda asociate
  - El calculeaza arbori de acoperire pentru fiecare astfel de grup si se foloseste de pruning pentru a taia ramurile ce nu sunt necesare, deci m arbori retezati => n * m arbori
  - Consuma foarte multa memorie pentru retinerea arborilor
  - Solutie: **core trees** retin un singur arbore pentru fiecare grup, cu radacina langa mjlocul grupului; drumul nu e optim, dar e un compromis bun
- **Dirijarea pentru calculatoare gazda mobile**
  - Pentru a dirija un pachet catre un calculator mobil, reteaua trebuie sa il localizeze
  - Calculatoare:
    - Stationare
    - Migratoare
    - Calatoare

- Gazda mobile = migratoare + calatoare
- Toate calculatoarele gazda au o locatie de domiciliu
- Fiecare domeniu are un **agent local** si un **agent pentru straini**
- Fiecare agent pentru strani difuzeaza un pachet anuntandu-si existenta si adresa; gazdele mobile pot si sa ceara asta.
- Calculatorul mobil se inregistreaza la agentul pentru straini
- Acesta ia legatura cu agentul local al domeniului din care provine calculatorul mobil si, dupa verificari de securitate, agentul local ii ofera confirmarea agentului pentru straini care poate acum sa inregistreze calculatorul mobil

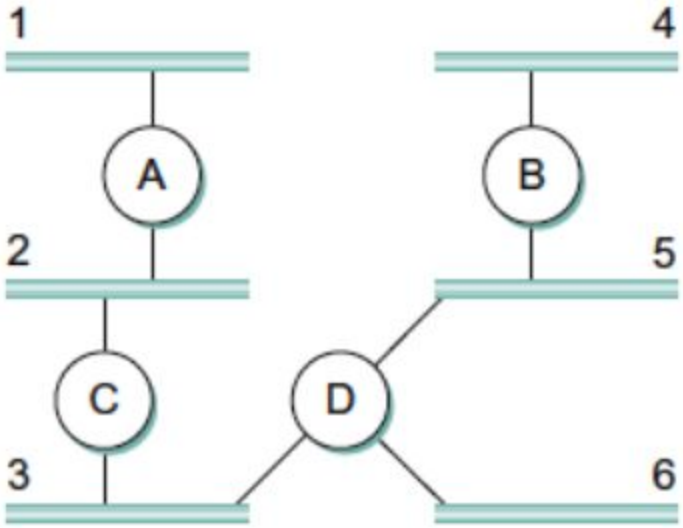
- Cand se transmite un pachet catre o gazda mobila, acesta este trimis la domiciliu; ulterior, agentul local cauta noua locatie temporara, il trimite acolo si anunta expeditorul unde sa transmita mesaje de acum inainte

- **Dirijarea in retele AD-HOC**
  - Retele de noduri de intamplator se afla aproape unul de celalalt se numesc ad-hoc sau MANET
  - Algoritm de dirijare: **AODV** (Ad hoc On demand Distance Vector)
  - Este un algoritm la cerere, aplicat doar atunci cand cineva doreste sa transmita un pachet
  - Doua noduri ale unei retele ad-hoc sunt conectate daca pot comunica direct folosind radioul
  - Descoperirea rutei: se cauta prin difuzari catre nodurile unde se poate ajunge; se retine si drumul
  - Algoritmul poate genera multe difuzari, asa ca poate fi modificat;
  - Se trimit pachete cu durata de viata 1, apoi cu 2, apoi cu 3 etc.
  - In AODV, spre deosebire de Bellman Ford, nodurile nu difuzeaza periodic intreaga lor tabela de dirijare, economising latime de banda

RIP - Routing Information Protocol

- fiecare legatura are cost 1
- foloseste **distante la retele** (nu la noduri) – ruterul C are distanta 0 la reteaua 2 si 2 la reteaua 4

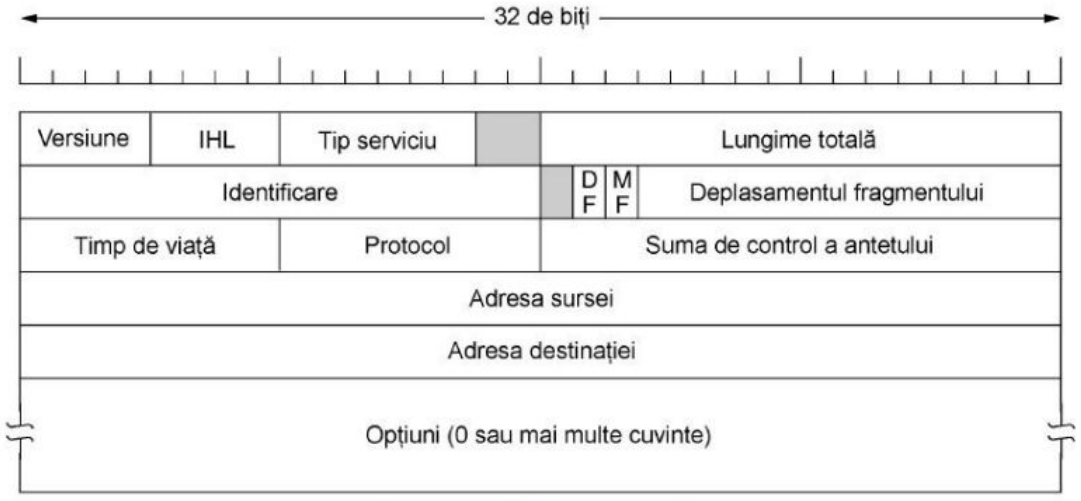
- transmit vectorii distantelor la fiecare 30 secunde
- distante maxime de 15 hop-uri (**16 inseamna infinit**) – rețele de mici dimensiuni



15. Protocolul IP

Principala sarcina a nivelului retea este aceea de a transfera datagrame de la sursa la destinatie in cel mai bun mod posibil, fara a tine cont daca fac sau nu parte din aceeasi retea sau daca intre ele exisa sau nu alte retele.

O **datagramă IP** constă dintr-o parte de **antet** și o parte de **text**. Antetul are o parte fixă de 20 de octeți și o parte opțională cu lungime variabilă. Transmis în ordinea big endian .



- **Versiune** memorează cărei versiuni de protocol îi aparține datagrama (IPv4 / IPv6)
- **IHL** - lungimea antetului
- **Tip serviciu** - sunt posibile diferite combinatii intre fiabilitate si viteza (ex: pentru voce digitalizata, e mult mai important sa fie livrat rapid decat corect; pentru transfer de fisiere, trebuie transmise fara erori, in detrimentul vitezei)
- **Lungime totala** - lugimea totata a datagramei
- **Identificare** - din ce datagrama face parte
- **DF / MF** - don't fragment / more fragments
- **Deplasamentul fragmentului** - indica pozitia fragmentului in cadrul datagramei
- **Timp de viata** - limiteaza durata de viata a pachetelor
- **Protocol** - UDP, TCP si altele
- **Suma de control a antetului** - utilizata pentru a evita erorile generate de locatii de memorie proaste din interiorul unui ruter
- **Adresa sursa/ adresa destinatie** - numarul de retea si numarul de gazda
- **Optiuni** - permite versiunilor urmatoare sa includa optiuni ce nu sunt prevazute in versiunea curenta
    - Securitate
    - Dirijare stricta de la sursa - indica o anumtia cale
    - Dirijare aproximativa de la sursa
    - Inregistreaza calea
    - Amprenta de timp

16. Adrese IP

Fiecare gazdă și ruter din Internet are o adresă IP, care codifică a**dresa sa de rețea** și de **gazdă**. Toate adresele IP sunt de 32 de biți lungime și sunt folosite în câmpurile Adresă sursă și Adresă destinație ale pachetelor IP. Adresele IP erau împărțite în cinci categorii:

Valoarea **0** înseamnă rețeaua curentă sau gazda curentă. Valoarea **-1 (127)** este folosită ca o adresă de difuzare pentru a desemna toate gazdele din rețeaua indicată.

CIDR (Classless InterDomain Routing - Dirijarea fără clase între domenii)

- Rezolva problema **insuficientei adreselor IP.**
- aloca adresele IP rămase, în blocuri de dimensiune variabilă, **fără a se ține cont de clase**
- Renunțarea la clase face rutarea mai complicată.
- Fiecare intrare din tabela de rutare este extinsa cu o **masca** de 32 de biti.
- Cand soseste un pachet, se extrage adresa si, folosind masca (SI logic), se compara cu intrarea din tabela; daca exista mai multe potriviri, se alege masca cea mai lunga

17. NAT – Translatarea adreselor de retea

- Legata tot de faptul caa adresele IP sunt insuficiente
- Reprezinta o rezolvare rapida pe termen scurt
- Ideea de bază a NAT-ului este de a aloca fiecărei companii o singură adresă IP
- În interiorul companiei, fiecare calculator primește o adresă IP unică, care este folosită pentru traficul intern.
- când un pachet părăsește compania, trece printr-o unitate NAT care convertește adresa IP internă la adresa IP reală a companiei.

- ❏ Se foloseste de campul Port in antetul IP-ului, pe care in inlocuieste cu un numar din tabela de intrari a NAT-ului
- ❏ Cand un pachet doreste sa ajunga in companie, NAT-ul transforma adresa companiei in adresa IP interna si reface portul initial
- ❏ Multi oameni sunt impotriva NAT-ului deoarece
    - ❏ Mai multe persoane folosesc aceeasi adresa IP
    - ❏ Transforma o retea fara conexiuni intr-o retea cu conexiuni
    - ❏ Nivelurile nu mai sunt independente
    - ❏ Procesele sunt constranse sa foloseasca TCP sau UDP (pentru a putea schimba portul cu o intrare din tabela NAT)

18. ICMP - Protocolul mesajelor de control din Internet

Atunci când se întâmplă ceva neobișnuit in internet, evenimentul este **raportat prin ICMP**, care este folosit și pentru **testarea** Internet-ului. Fiecare tip de mesaj ICMP este **încapsulat** într-un pachet IP.

Tipul mesajului	Descriere
Destinație inaccesibilă	Pachetul nu poate fi livrat
Timp depășit	Câmpul timp de viață a ajuns la 0
Problemă de parametru	Câmp invalid în antet
Oprire sursă	Pachet de șoc
Redirectare	Învață un ruter despre geografie
Cerere de ecou	Întreabă o mașină dacă este activă
Răspuns ecou	Da, sunt activă
Cerere de amprentă de timp	La fel ca cererea de ecou, dar cu amprentă de timp
Răspuns cu amprentă de timp	La fel ca răspunsul ecou, dar cu amprentă de timp

- ❏ Mesajul **DESTINAȚIE INACCESIBILĂ** (DESTINATION UNREACHABLE) este folosit atunci când subrețeaua sau un ruter **nu pot localiza destinația**, sau un pachet cu bitul DF nu poate fi livrat deoarece o rețea cu „pachete mici” îi stă în cale.
- ❏ Mesajul **TIMP DEPĂȘIT** (TIME EXCEEDED) este trimis când un pachet este eliminat datorită **ajungerii contorului său la zero**. Acest mesaj este un simptom al buclării

pachetelor, al unei enorme congestii sau al stabilirii unor valori prea mici pentru ceas.

- ❏ Mesajul **PROBLEMĂ DE PARAMETRU** (PARAMETER PROBLEM) indică detectarea unei **valori nepermise într-un câmp din antet.** Această problemă indică o eroare în programele IP ale gazdei emițătoare sau eventual în programele unui ruter tranzitat.
- ❏ Mesajul **OPRIRE SURSĂ** (SOURCE QUENCH) a fost folosit pe vremuri pentru a **limita traficul gazdelor care trimiteau prea multe pachete**. Când o gazdă primea acest mesaj, era de așteptat să încetinească ritmul de transmisie. Este folosit arareori, deoarece când apare congestie, aceste pachete au tendința de a turna mai mult gaz pe foc.
- ❏ Mesajul **REDIRECTARE** (REDIRECT) este folosit atunci când un ruter observă că un **pachet pare a fi dirijat greșit**. Este folosit de ruter pentru a spune gazdei emițătoare despre eroarea probabilă.
- ❏ Mesajele **CERERE ECOU** (ECHO REQUEST) și **RĂSPUNS ECOU** (ECHO REPLY) sunt folosite pentru a **vedea dacă o anumită destinație este accesibilă și activă**. Este de așteptat ca la recepția mesajului ECOU, destinația să răspundă printr-un mesaj RĂSPUNS ECOU.
- ❏ Mesajele **CERERE AMPRENTĂ DE TIMP** (TIMESTAMP REQUEST) și **RĂSPUNS AMPRENTĂ DE TIMP** (TIMESTAMP REPLY) sunt similare, cu excepția faptului că în răspuns sunt înregistrate **timpul de sosire a mesajului și de plecare a răspunsului.** Această facilitate este folosită pentru a măsura performanțele rețelei.

19. Protocolul de rezoluție a adresei: ARP

Deși fiecare mașină din Internet are una sau mai multe adrese IP, acestea nu pot fi folosite de fapt pentru trimiterea pachetelor deoarece hardware-ul **nivelului legăturii de date nu înțelege adresele Internet**. Fiecare placă Ethernet fabricată până acum vine cu o adresă Ethernet de **48 biți.** Nu există două plăci cu aceeași adresă.

In momentul in care un pachet ajunge in retea prin IP, reteaua are nevoie sa afle adresa Ethernet a destinatarului.

Gazda trimite un pachet de difuzare în rețeaua Ethernet întrebând: **„Cine este proprietarul adresei IP 192.31.65.5?”**.

Protocolul folosit pentru a pune astfel de întrebări și a primi răspunsul se numește **ARP.**

Este preferat ARP-ul in defavoarea fisierelor de configurare, deoarece acestea pot ajunge foarte mari.

Optimizari ale ARP-ului:

- ❏ la fiecare execuție a ARP, mașina păstrează rezultatul pentru cazul în care are nevoie să contacteze din nou aceeași mașină în scurt timp
- ❏ gazda 1 să includă în pachetul ARP corespondența dintre adresa sa IP și adresa Ethernet.
- ❏ fiecare mașină să difuzeze corespondența sa de adrese la pornirea mașinii. Difuzarea este realizata printr-un pachet ARP de cautare a propriei adrese IP

20. DHCP

Ca și RARP și BOOTP, DHCP este bazat pe ideea unui server special care atribuie adrese IP gazdelor care cer una. Acest server **nu** trebuie să se afle în **același LAN** cu gazda care face cererea. Deoarece serverul DHCP s-ar putea să nu fie accesibil prin difuzare, este nevoie ca **în fiecare LAN să existe un agent de legătură DHCP** (*DHCP relay agent*)

Pentru a-și **afla adresa IP**, o mașină tocmai pornită difuzează un pachet **DHCP DISCOVER**. Agentul de legătură DHCP din LAN interceptează toate difuzările DHCP.

Problema: cat de mult trebuie alocata o adresa IP?

- ❏ Dacă o gazdă părăsește rețeaua și nu returnează adresa sa IP serverului DHCP, acea adresă va fi pierdută permanent.
- ❏ atribuirea adresei IP va fi pentru o perioadă fixă de timp (închiriere)
- ❏ Inainte de expirare, gazda trebuie sa ceara innoire daca vrea sa continue sa foloseasca respectiva adresa IP

21. Protocolul de dirijare folosit de porțile interioare: OSPF

- Internet-ul este construit dintr-un număr mare de sisteme autonome(AS).
- Fiecare AS este administrat de o organizație diferită și poate folosi propriul **algoritm de dirijare în interior**.
- Chiar si asa, exista **standarde** pentru a face mai usoara comunicarea la **granitele** AS-urilor.
- OSPF suportă trei tipuri de conexiuni și rețele:
  - 1. Linii **punct-la-punct între exact două rutere**.
  - 2. Rețele **multiacces cu difuzare** (de exemplu, cele mai multe **LAN**-uri).
  - 3. Rețele **multiacces fără difuzare** (de exemplu, cele mai multe **WAN**-uri cu comutare de pachete).
- funcționează prin **abstractizarea** colecției de rețele, rutere și linii reale intr-un graf orientat
- OSPF permite AS-urilor să fie **divizate** în zone numerotate, unde o zonă este o rețea sau o mulțime de rețele învecinate.
- Orice AS are o zonă de **coloană vertebrală**, numită zona 0.
- pot fi necesare trei tipuri de căi:
  - Intrazonale
  - Interzonale
  - interASuri.
- OSPF funcționează prin schimb de informații între rutere **adiacente**

Tip mesaj	Descriere
Hello	Folosit pentru descoperirea vecinilor
Actualizare stare legatura	Emitatorul furnizeaza vecinilor costurile sale
Confirmare stare legatura	Confirma actualizarea starii
Descriere baza de date	Anunta ce actualizari are emitatorul
Cerere stare legatura	Cere informatii de la partener

22. Protocolul de dirijare pentru porți externe: BGP

În cadrul unui singur AS, protocolul de dirijare recomandat este OSPF (deși, desigur, nu este singurul folosit). **Între AS-uri** se folosește un protocol diferit, **BGP** (Border Gateway Protocol – Protocolul porților de graniță).

Ruterele ce folosesc protocolul de porți exterioare trebuie să țină cont întro mare măsură de **politică**.

Protocoalele pentru porți externe, în general și BGP în particular, au fost proiectate pentru a permite forțarea multor tipuri de politici de dirijare pentru traficul între AS-uri. Politicile tipice implică considerații **politice, de securitate sau economice**. Câteva exemple de constrângeri de dirijare sunt:

1. Nu se tranzitează traficul prin anumite AS-uri.
2. Nu se plasează Irak-ul pe o rută care pornește din Pentagon.
3. Nu se folosesc Statele Unite pentru a ajunge din Columbia Britanică în Ontario.
4. Albania este tranzitată numai dacă nu există altă alternativă către destinație.
5. Traficul care pleacă sau ajunge la IBM nu trebuie să tranziteze Microsoft.

Politicile sunt configurate manual în fiecare ruter BGP, nu fac parte din protocolul insusi.

HDLC	High-level Data Link Control	
SDLC	Synchronous Data Link Control	
ADCCP	Advanced Data Communication Control Procedure	
LAP	Link Access Procedure	
CRC	Cyclic Redundancy Code	
PPP	Point to Point Protocol	
LCP	Link Control Protocol	
NCP	Network Control Protocol	
TCP	Transmission Control Protocol	
UDP	User Datagram Protocol	
ICMP	Internet Control Message Protocol	
IGMP	Internet Group Management Protocol	
OSPF	Open Shortest Path First	
BGP	Border Gateway Protocol	
DNS	Domain Name System	
ISP	Internet Service Provider	
SONET	Syncronous Optical NETwork	

IMP	Interface Message Processor
STDM	Synchronous Time Division Multiplexing
FDM	Frequency Division Multiplexing
MTU	Maximum Transmission Unit
API	Application Programming Interface
TPDU	Transaction Protocol Data Unit
RTP	Real-Time Transport Protocol
RIP	Routing Information Protocol
MANET	Mobile Ad-hoc NETwork
AODV	Ad-hoc On demand Distance Vector
CIDR	Classless InterDomain Routing
NAT	Network Address Translation
ARP	Address Resolution Protocol
URL	Uniform Resource Locator
DHCP	Dynamic Host Configuration Protocol
RARP	Reverse Address Resolution Protocol
OSPF	Open Shortest Path First
AS	Autonom System
BGP	Border Gateway Protocol
RR	Resource Record
MIME	Multi-Purpose Internet Mail Extensions

23. IPv6

- Ex: 105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255
- Ex: 69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF
- **Compresie zerouri :** FF0C:0:0:0:0:0:0:B1 = FF0C::B1
- Mai bun decat IPv4:
    - 32 biti -> 128 biti
    - Antet mai simplu (7 campuri in loc de 13) - ruterele lucreaza mai rapid

- Campurile necesare devin optionale - pachetele sunt prelucrate mai rapid
- Securitate

- Nu e compatibil cu IPv4, dar e compatibil cu celelalte protocoale Internet:
  - TCP, UDP, ICMP, IGMP, OSPF, BGP, DNS cu foarte mici modificari pentru a lucra cu adrese mai lungi

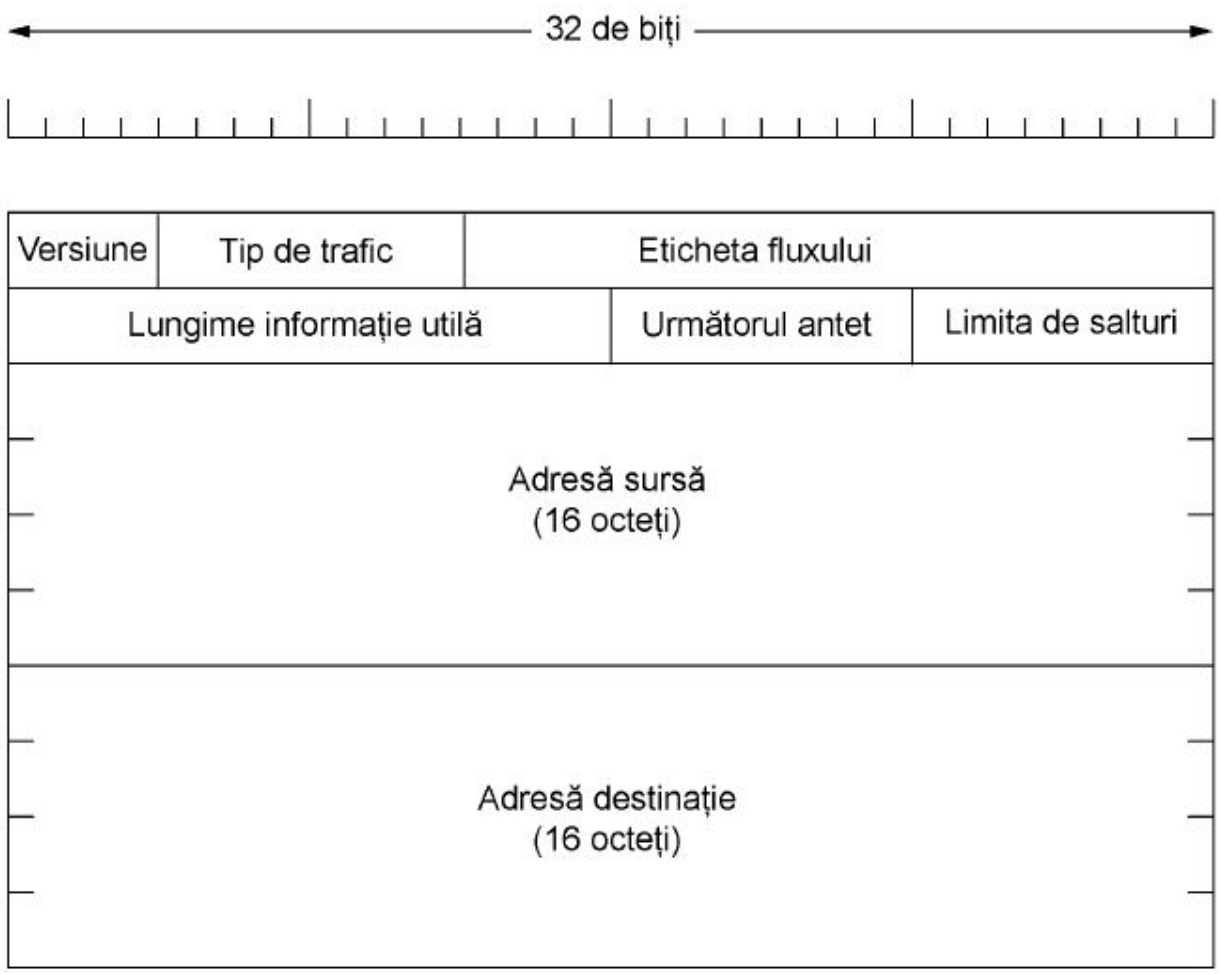


Fig. 5-68. Antetul fix IPv6 (obligatoriu).

- Campuri:
  - Versiune: 6 (4 pt IPv4)
  - Tip de trafic: distingerea pachetelor cu cerinte de livrare in timp real
  - Eticheta fluxului (experimental):
    - Pseudo-conexiune intre sursa si destinatie cu proprietati si cerinte particulare

- Asociaza datagramele unui flux
- Adresa sursa, destinatie, numar de flux => pot exista mai multe fluxuri active in acelasi timp (chiar si doua fluxuri venind de la gazde diferite cu acelasi numar de flux vor fi separate in ruter)
- Numerele de flux sunt alese aleator si nu secvential de la 1 deoarece vor fi folosite in tabele de dispersie

- Lungime informatie utila: cati octeti urmeaza **dupa antet**
- Antetul urmator:
  - Spune carui tip de protocol (ex TCP, UPD) i se va transmite pachetul
  - Defineste tipul antetului de extensie

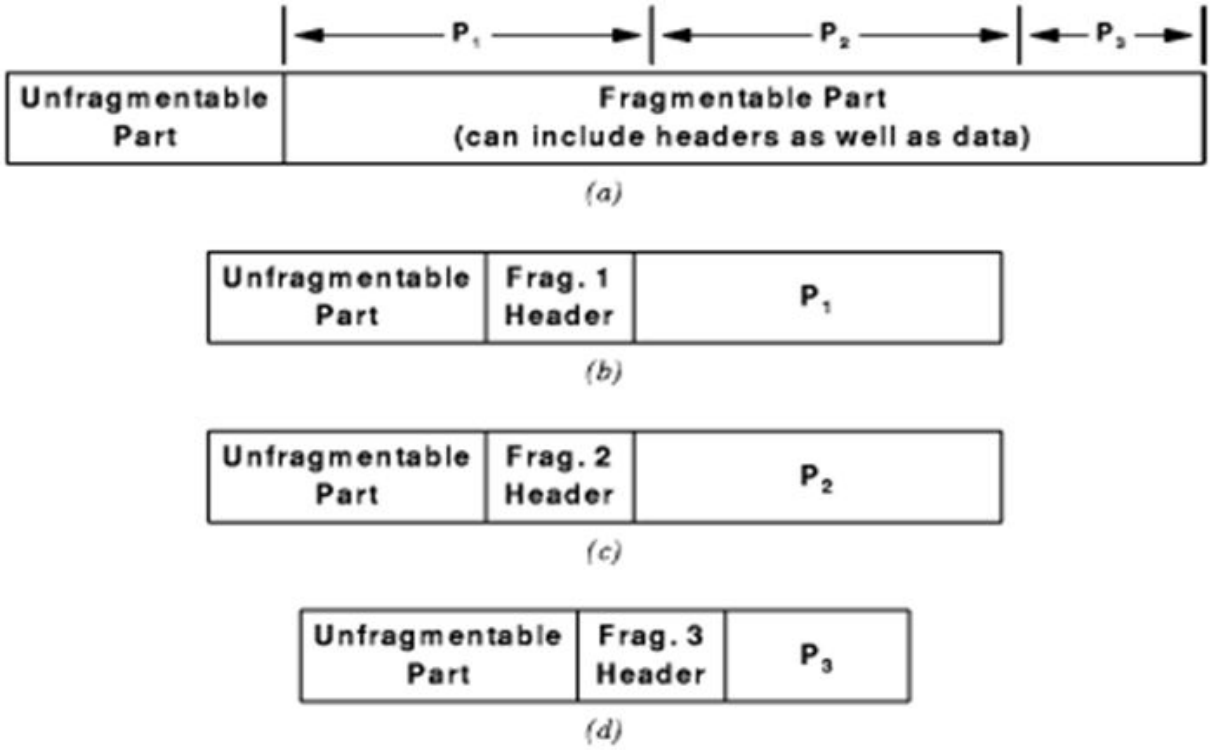
(a)

(b)

- Limita salturilor: impiedica pachetele sa “traiasca” vesnic (scade la fiecare salt dintr-o retea in alta)

- Antete extensie:
  - **Hop-by-hop** header:
    - Suport datagrame > 64K (jumbograme)
    - Specifica lungimea
    - Campul de lungime din antetul de baza = 0
  - **Destination** header:
    - Info aditionale pentru destinatie
  - **Routing**: lista rutere de vizitat
  - **Fragmentation:** identificare fragmente
  - **Authentication**: verificare identitate transmitator
  - **Encrypted security payload:** info despre continut criptat

- **Fragmentarea:**
  - Caracter dinamic: Calea se poate schimba
  - Eficienta: Antetul nu are spatiu pierdut
  - Flexibilitate: Noi antete pentru caracteristici
  - Dezvoltare incrementala: rutele care trateaza anumite antete coexista cu altele care le ignora
  - La sursa care:
    - Fragmenteaza pachetele
    - Descopera **path MTU**
  - Ruterele ignora datagramele mai lungi decat MTU

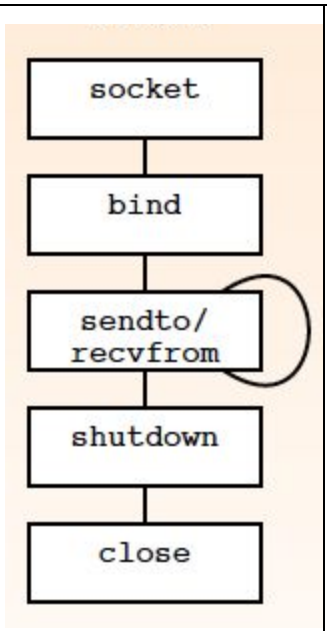


24. Comunicarea intre aplicatii

- **Identificare unica** a unui punct de acces prin <adresa_retea, port>
  - Transferul intre gazde foloseste adresa_retea
  - Comunicarea intre procese foloseste portul

- Servicii ale nivelului **Transport:**
  - **Transfer de date intre procese de aplicatie**, folosind retele de diverse tipuri
  - **Interfata uniforma** cu utilizatorii
  - Doua tipuri:
    - Orientate pe conexiune: TCP
    - Fara conexiune: UDP
  - Disponibile ca **API:**
    - **API** descrie cum sunt apelate functiile
    - Socket API:
      - Disponibil pe Windows/Solaris etc.
      - **Socket** = punctul in care procesul de aplicatie se atasaza la retea (identificat prin descriptor - **numar,** ca la fisiere)

- Creare socket
  - int socket(int family, int type, int protocol)
  - **socket_descr** = socket (protocol_family, comm_type, protocol)
    - Deschide un socket
    - Intoarce **socket_descr** folosit in apelurile urmatoare
  - **Protocol_family** selecteaza familia de protocoale
    - PF_INET - protocoale Internet
    - PF_APPLETALK - protocoale AppleTalk etc.
  - **Comm_type** selecteaza tipul de comunicare
    - SOCK_DGRAM - fara conexiune (datagrama)
    - SOCK_STREAM - orientat pe conexiune (flux de octeti)
  - **Protocol** specifica protocolul
    - IPPROTO_TCP - TCP
    - IPPROTO_UPD - UDP



Creaza socket si aloca-i resurse de sistem:
int s = socket (AF_INET, SOCK_DGRAM, IPPROTO_UDP);
Asociaza un socket cu <port, adresa_IP>:
int bind (int socket_descriptor, struct sockaddr* local_address, int
address_length)
bind(s, &addr, sizeof(addr)); // addr locala - vezi slide urmator !
Primeste mesaje de la un socket aflat la distanţă:
int recvfrom (int socket_descriptor, char* buffer_address, int buffer_length, int flags, struct sockaddr* sender_address, unsigned Int sendaddress_length)
recvfrom(s, buf, BUFLEN, 0, NULL, NULL);
Opreşte trimitere sau/şi recepţie de date
shutdown (s, SHUT_RD / SHUT_RDWR / SHUT_WR)
Inchide socket - termina utilizarea socket si elibereaza resurse alocate
close (s)

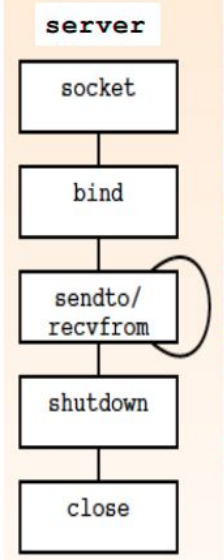
Setare adresa

Format TCP/IP:
struct sockaddr_in { u_char sin_len; /* total length of address */
u_char sin_family; /* family of the address */
u_short sin_port; /* protocol port number */
struct in_addr sin_addr; /* IP address */
char sin_zero[8] /* unused */ }

struct sockaddr_in serv_addr
memset ((char *) &serv_addr, 0, sizeof(serv_addr));

serv_addr.sin_family = AF_INET; // adrese pentru Internet
serv_addr.sin_addr.s_addr = INADDR_ANY; // foloseste adresa IP a masinii
serv_addr.sin_port = htons(portno); // converteste de la host la network byte order

Serviciu fara conexiune - server



Creaza socket si aloca-i resurse de sistem:
*int s = **socket** (AF_INET, SOCK_DGRAM, IPPROTO_UDP);*
Asociaza un socket cu <port, adresa_IP>:
int bind (int socket_descriptor, struct sockaddr* local_address, int address_length)
bind(s, &addr, sizeof(addr)); // addr locala - vezi slide urmator !
Primeste mesaje de la un socket aflat la distanță:
int recvfrom (int socket_descriptor, char* buffer_address, int buffer_length, int flags, struct sockaddr* sender_address, unsigned int sendaddress_length)
recvfrom(s, buf, BUFLEN, 0, NULL, NULL);
Oprește trimitere sau/și recepție de date
shutdown (s, SHUT_RD / SHUT_RDWR / SHUT_WR)
Inchide socket - termina utilizarea socket si elibereaza resurse alocate
close (s)

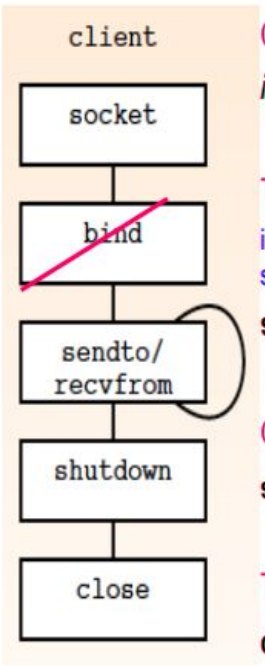
Setare adresa

Format TCP/IP:

```
struct sockaddr_in { u_char  sin_len;     /* total length of address */
    u_char  sin_family;                   /* family of the address   */
    u_short sin_port;                     /* protocol port number    */
    struct  in_addr sin_addr;             /* IP address              */
    char    sin_zero[8]                   /* unused                  */
}

struct sockaddr_in serv_addr
memset ((char *) &serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;                  // adrese pentru Internet
serv_addr.sin_addr.s_addr = INADDR_ANY;
                                      // foloseste adresa IP a masinii
serv_addr.sin_port = htons(portno);
                          // converteste de la host la network byte order
```

Serviciu fara conexiune - client



Creaza socket și aloca-i resurse de sistem:

*int s = **socket** (AF_INET, SOCK_DGRAM, PPROTO_UDP);*

Trimite mesaj la un socket aflat la distanță:

int sendto (int socket_descriptor, char* data_address, int data_length, int flags, struct sockaddr* dest_address, int destaddress_length)

sendto (s, buf, BUFLEN, 0, &addr, sizeof(addr));

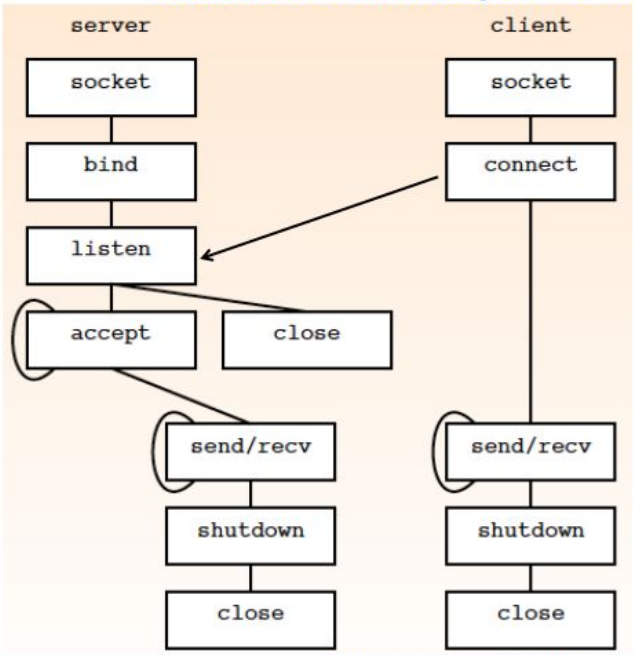
Opreste trimitere sau/și receptie de date

shutdown (s, SHUT_RD / SHUT_RDWR / SHUT_WR)

Termina utilizarea socket si elibereaza resursele alocate

close (s)

Serviciu orientat pe conexiune



Serviciu orientat pe conexiune - client

1.Creaza socket:

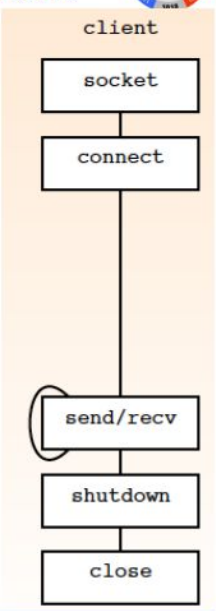
*int s = **socket** (AF_INET, SOCK_STREAM, IPPROTO_TCP);*

2. Conecteaza un socket client (specificat prin descriptor) cu un socket server (precizat prin adresa):

int connect (int client_socket_descriptor, struct sockaddr* server_socket_address, int server_sockaddress_length)

connect(**s**, &addr, sizeof(addr));

3. repetat - Trimite / primeste etc.



Transmisia de date cu TCP

send (s, buf, len)
int send(int socket, const char* buf, int len, int flags);
• Intoarce numarul de octeti trimisi
– Poate fi mai mic decat len !
recv (s, buf, max_len)
int recv(int socket, char* buf, int len, int flags);
• Intoarce numarul de octeti primiti
– Poate fi mai mic decat max_len!
flags indica optiuni speciale:
MSG_OOB – trimite/primeste date out-of-band
MSG_PEEK – livreaza date primite, dar trateaza ca necitite

Inchiderea conexiunii TCP

- Elibereaza resursele asociate conexiunii
- Informeaza capatul celalalt de inchiderea conexiunii
- API
  - shutdown (s,SHUT_RD/SHUT_RDWR/SHUT_WR)
    - int shutdown (int socket, int how)
    - opreste primirea – rejecteaza datele care sosesc
    - opreste transmiterea – ignora datele inca netrimise
    - Ambele
  - close (s)
    - int close (int socket);

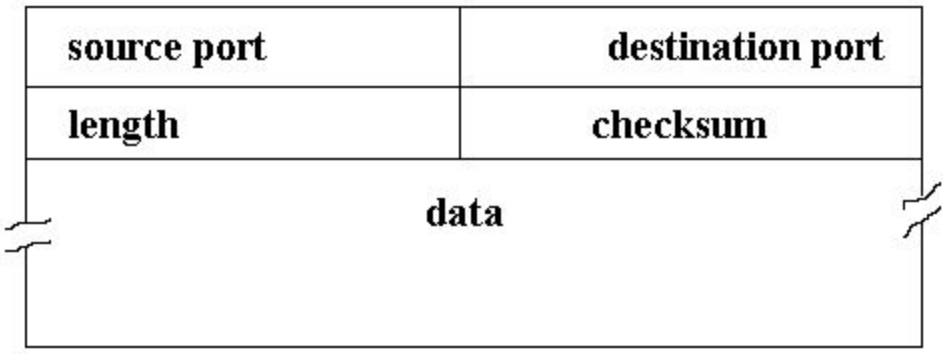
- inchide socket (elibereaza structurile de date din kernel)

UDP - User Datagram Protocol

UDP livreaza datagrame utilizator - user datagrams

- Livrare "Best effort" – datagramele pot fi pierdute, primite in alta ordine etc.
- Sume de control pentru integritate • Puncte de capat UDP = protocol ports sau ports • UDP identifica adresa Internet si numar port pentru sursa si destinatie • Destination port si source port pot diferi.
- Puncte de capat UDP = protocol ports sau ports
- UDP identifica adresa Internet si numar port pentru sursa si destinatie
- **Destination port** si **source port** pot diferi.

Antet UDP



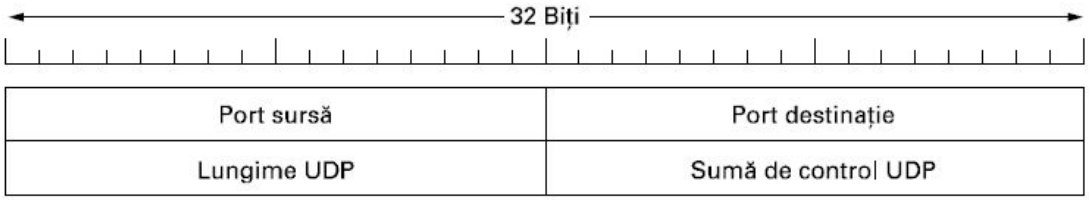


Fig. 6-23. Antetul UDP.

- checksum (calculat la fel ca la TCP) dar nefolosit
- Nu control flux
- Nu control erori
- Nu retransmisie
- Aplicatii care folosesc UDP:
  - DNS
  - Voice over IP
  - Online games
  - Se foloseste atunci cand:

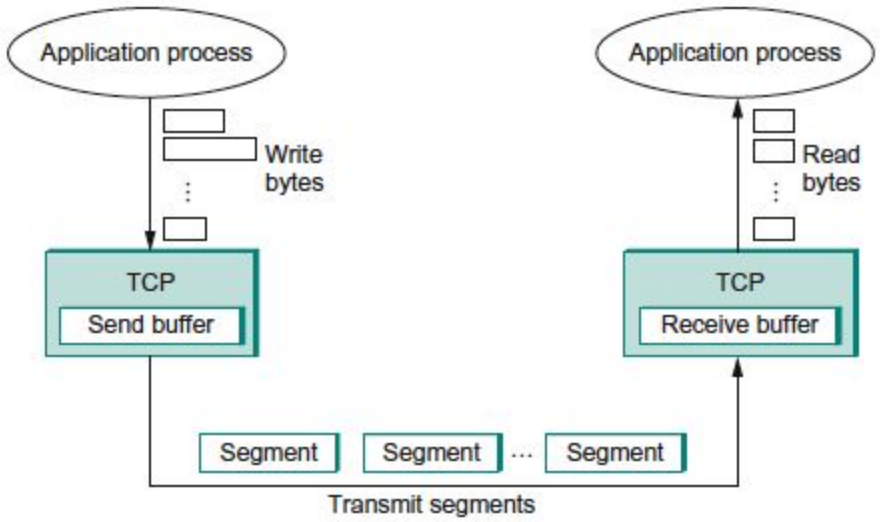
- Latenta este foarte importanta
- Livrarea tuturor datelor nu e necesara
- Retransmisiile sunt implementate de aplicatii cand e nevoie (DNS)

TCP - Transmission Control Protocol

Livrare sigura pe retea nesigura (datagrame)
Cel mai folosit protocol de transport

- Web
- Email
- SSH
- Chat
- Video Streaming
- Peer-to-peer

- TCP este **orientat pe flux de octeti (nu de mesaje):**

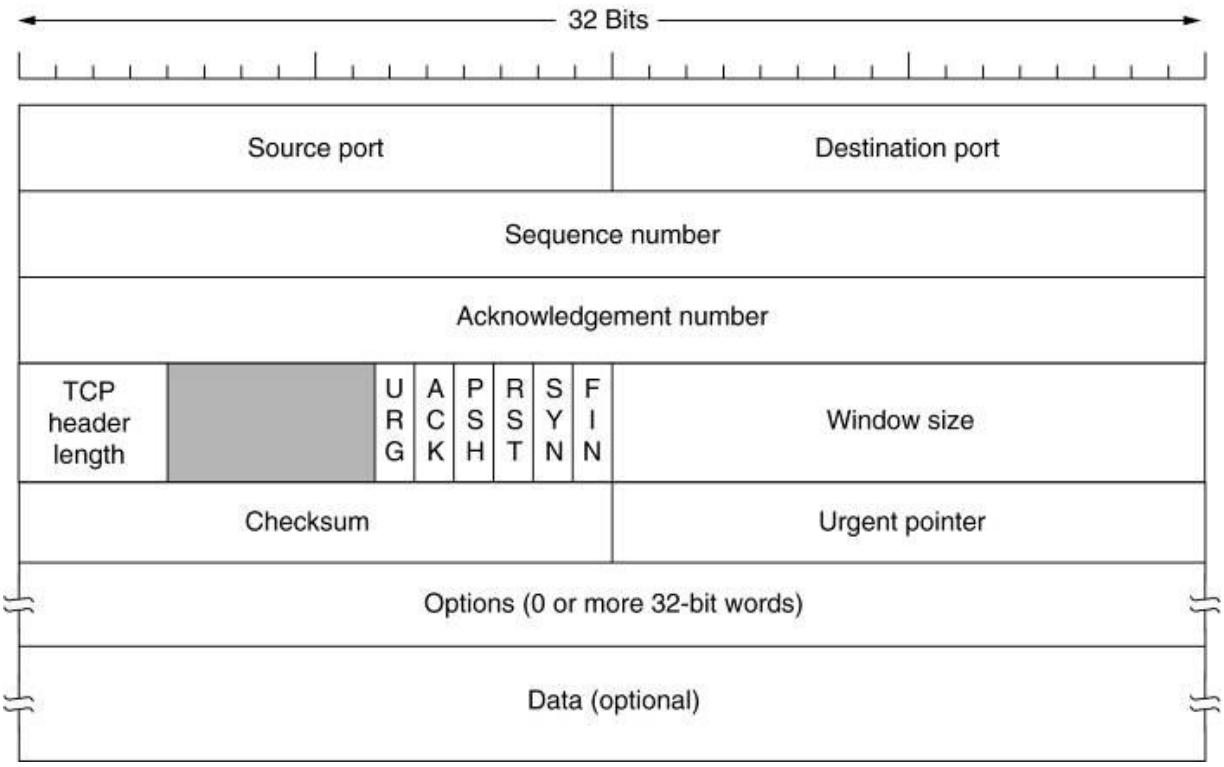


- Aplicația transmițătoare scrie octeți în conexiunea TCP
- TCP la sursă memorează octeții într-un buffer și transmite segmente
- TCP la destinație memorează segmentele într-un buffer
- Aplicația receptoare citește octeți (câți vrea!) din conexiunea TCP

- Caracteristici:
  - Orientat pe conexiune
  - Interfata flux (stream)
    - Transmisie si receptie pe siruri de octeti
    - Datele pot fi expediate imediat sau pot fi retinute intr-un tampon (pentru a colecta multa informatie trimisa in acelasi timp)
  - Face **controlul congestiei** adaptand viteza de transmisie la conditiile retelei
  - Garanteaza **transmisie in ordine si sigura** a datelor pe o conexiune

- **Full duplex**
- Stabilirea sigura a conexiunii: **three-way handshake**
  - Server:
    - LISTEN
    - ACCEPT
  - Client:
    - CONNECT - adresa IP si port
- Eliberare lina a conexiunii: **fara pierdere de date**

Antet TCP



Sequence number – numarul primului octet din segment
Acknowledgement number – numarul urmatorului octet asteptat
Window size - numărul de octeți care pot fi trimiși, începând cu octetul confirmat Urgent Pointer – deplasamentul, față de Sequence number, ptr. info. Urgentă
Port sursa si port destinatie - punctele finale ale conexiunii (end-point de 48 de biti = adresa IP a masinii + port)
Camp de 6 biti neutilizat - dovada ca TCP a fost bine proiectat (altfel ar fi avut nevoie de acei biti pentru erori)
Checksum - aduna toate cuvintele de 16 biti (XOR) si apoi XOR pe cei 16 biti finali. Receptorul face suma de control pe tot, inclusiv Checksum si trebuie sa dea 0.
Options
- Adaugarea unor facilitati suplimentare neacoperite de antetul obisnuit
- Cea mai importanta e aceea care permite fiecarei masini sa specifice incarcarea

maxima de informatie utila TCP pe care e dispusa sa o accepte (segmente mari => eficienta (mai putini octeti folositi pe antete))
    - ❏ Dimensiunea maxima nu trebuie sa fie aceeasi in ambele directii

- ❏ Indicatori:
    - ❏ P(U)SH - forteaza expedierea (de fapt ii spune sa nu intarzie din ratiuni de eficienta)
    - ❏ URG(ENT) - intreruperea acumularii si transmisia imediata a intregii informatii disponibile
    - ❏ ACK - numarul de confirmare este valid / invalid
    - ❏ RST - desfiinteaza o conexiune devenita inutilizabila; refuza un segment invalid sau o incercare de deschidere a unei conexiuni
    - ❏ SYN - stabilirea unei conexiuni (SYN = 1. ACK = 0 => cerere de conexiune, ACK = 1 => acceptare de conexiune)
    - ❏ FIN - pentru a incheia o conexiune; emitatorul nu mai are informatii de transmis (contine nr de secventa, la fel ca SYN, pentru a putea fi preluate in ordinea corecta)
- ❏ Protocol default folosit: **fereastra glisanta** de dimensiuni variabile

Pseudo-antet

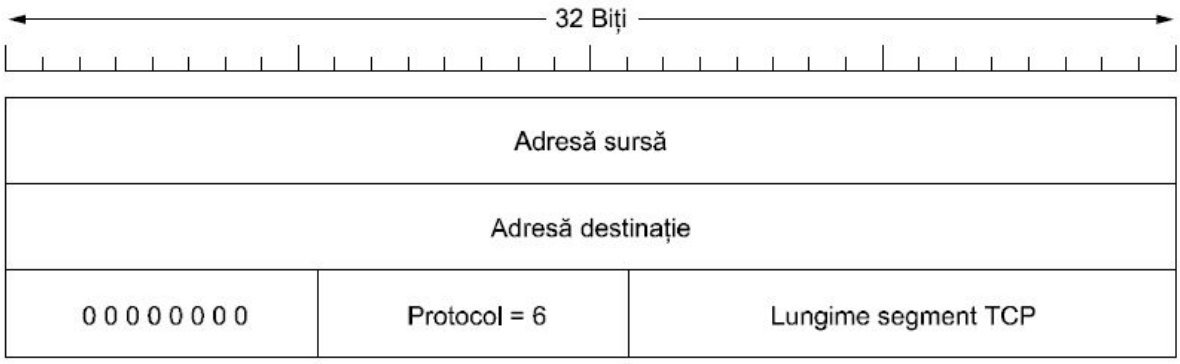


Fig. 6-30. Pseudo-antetul inclus în suma de control TCP.

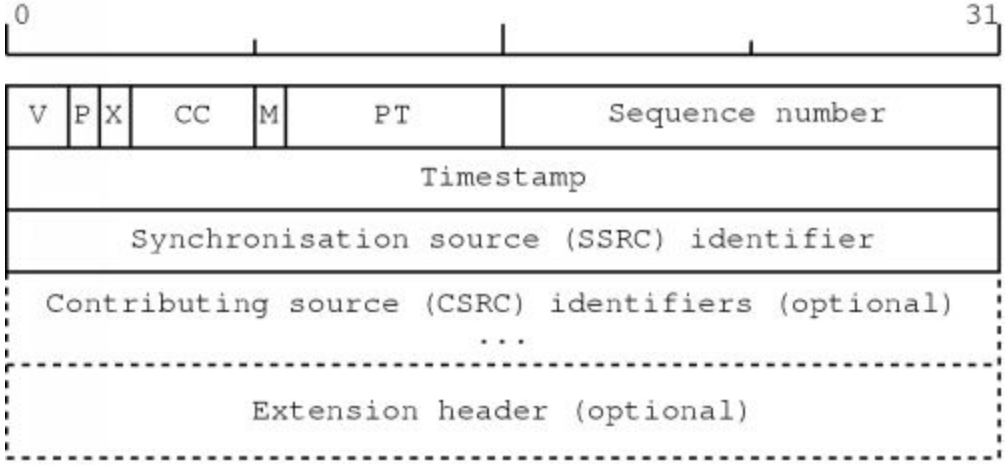
- ❏

RTP - Real-Time Transport Protocol

- ❏ Protocol de transport implementat la nivelul aplicatie (Ciudat, dar asa e)
- ❏ Folosit pentru:
    - ❏ Aplicatii multimedia in timp real
    - ❏ Radio pe internet
    - ❏ Telefonie pe internet

- Muzica la cerere
- Video-conferintele
- Video la cerere

- Ruleaza peste UDP
- Se afla in spatiul utilizator impreuna cu aplicatia
- Multiplexeaza mai multe fluxuri de date intr-un singur flux UDP
- Fiecare pachet are un numar cu 1 mai mare decat cel precedent (se poate afla daca dispar anumite pachete => se aproximeaza valoarea prin interpolare; nu se retransmite pentru ca fiind **REAL-TIME** va ajunge probabil prea tarziu)



- **X** indica prezenta unui antet extins
- **CC** arata cate surse contribuabile sunt prezente de la 0 la 15
- **M** este un bit de marcare specific aplicatiei (inceputul unui cadru video sau al unui canal audio de exemplu)
- **Sequence number** este un contor incrementat pe fiecare pachet RTP trimis; detecteaza pachete pierdute
- **Timestamp** ajuta la reducerea bruiajului din receptor; ne arata carui flux ii apartine pachetul; se foloseste la multiplexare/demultiplexare intr-un singur flux UDP

25. TCP Tranzactional

- Exista vreo posibilitate de a combina eficienta RPC-ului folosind UDP (doar doua mesaje) cu fiabilitatea TCP? Aproape ca da!
- T/TCP = Transactional TCP
- Transferul de date se realizeaza in timpul initializarii

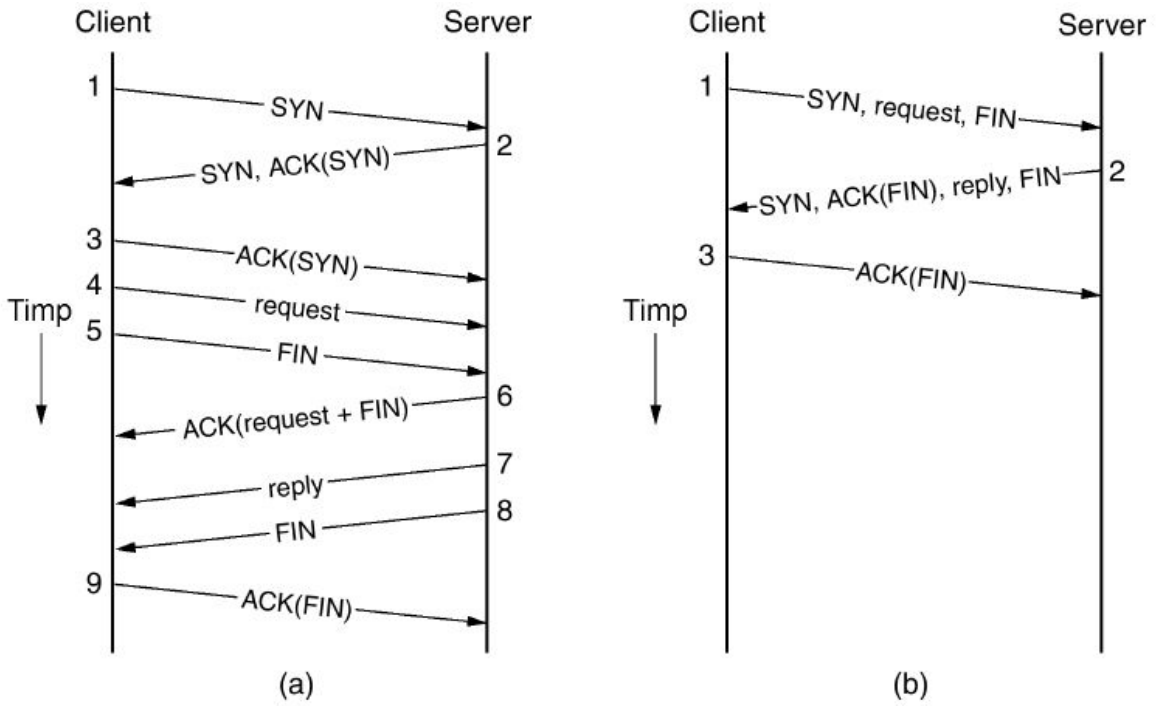
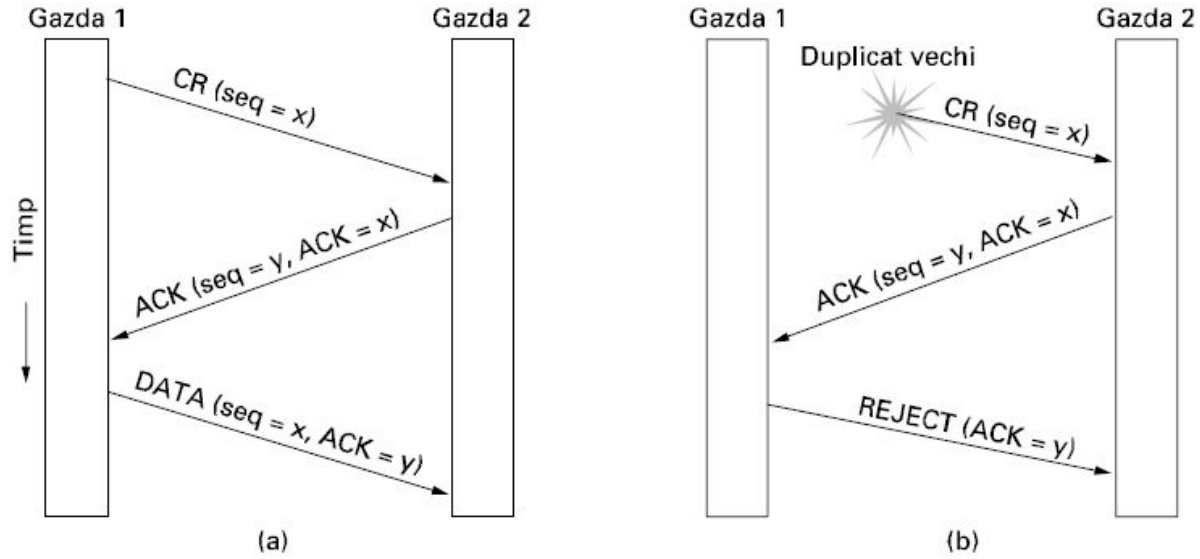


Fig. 6-40. (a) RPC folosind TCP clasic ; (b) RPC folosind T/TCP

26. Three-way handshake



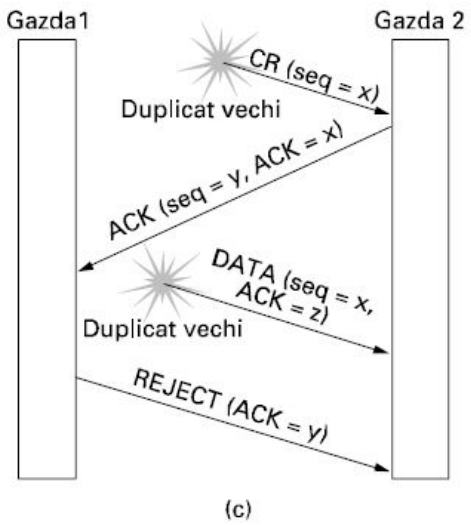


Fig. 6-11. Trei scenarii posibile de stabilire a conexiunii pentru un protocol cu înțelegere în trei pași. CR reprezintă CONNECTION REQUEST. (a) Cazul normal, (b) Un duplicat vechi al unui mesaj CONNECTION REQUEST apare când nu trebuie, (c) Sunt duplicate atât CONNECTION REQUEST cât și CONNECTION ACCEPTED.

- **CR** = Connection Request
- Acest protocol nu necesita ca ambele parti sa inceapa sa trimita acelasi numar de secventa, deci poate fi utilizat si impreuna cu alte metode de sincronizare decat ceasul global. (figura (a) reprezinta conectarea)
- Eliberarea unei conexiuni e mai usoara decat conectarea:
  - Eliberare asimetrica:
    - Brusca si poate genera pierderi

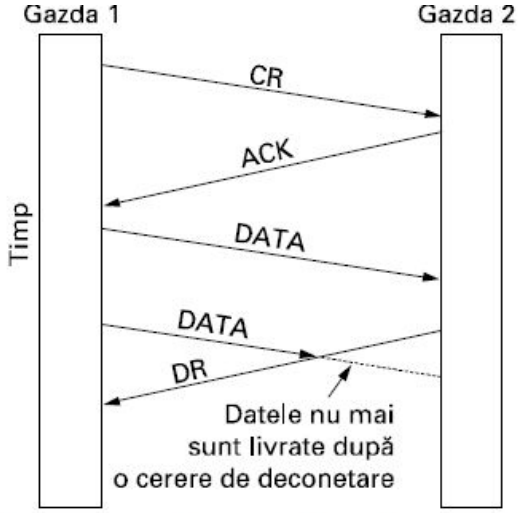


Fig. 6-12. Deconectare bruscă cu pierdere de date. CR= CONNECTION REQUE ACK=CONNECTION ACCEPTED , DR=DISCONNECT REQUEST.

-
- Se foloseste la apelurile telefonice

- Eliberare simetrica:
  - Fiecare directie este eliberata independent de cealalta
  - Un calculator gazda poate continua sa primeasca date chiar si dupa ce a trimis TPDU de eliberare a conexiunii
  - Utila atunci cand fiecare proces are o cantitate fixa de date de trimis si stie bine cand trebuie sa transmita si cand a terminat

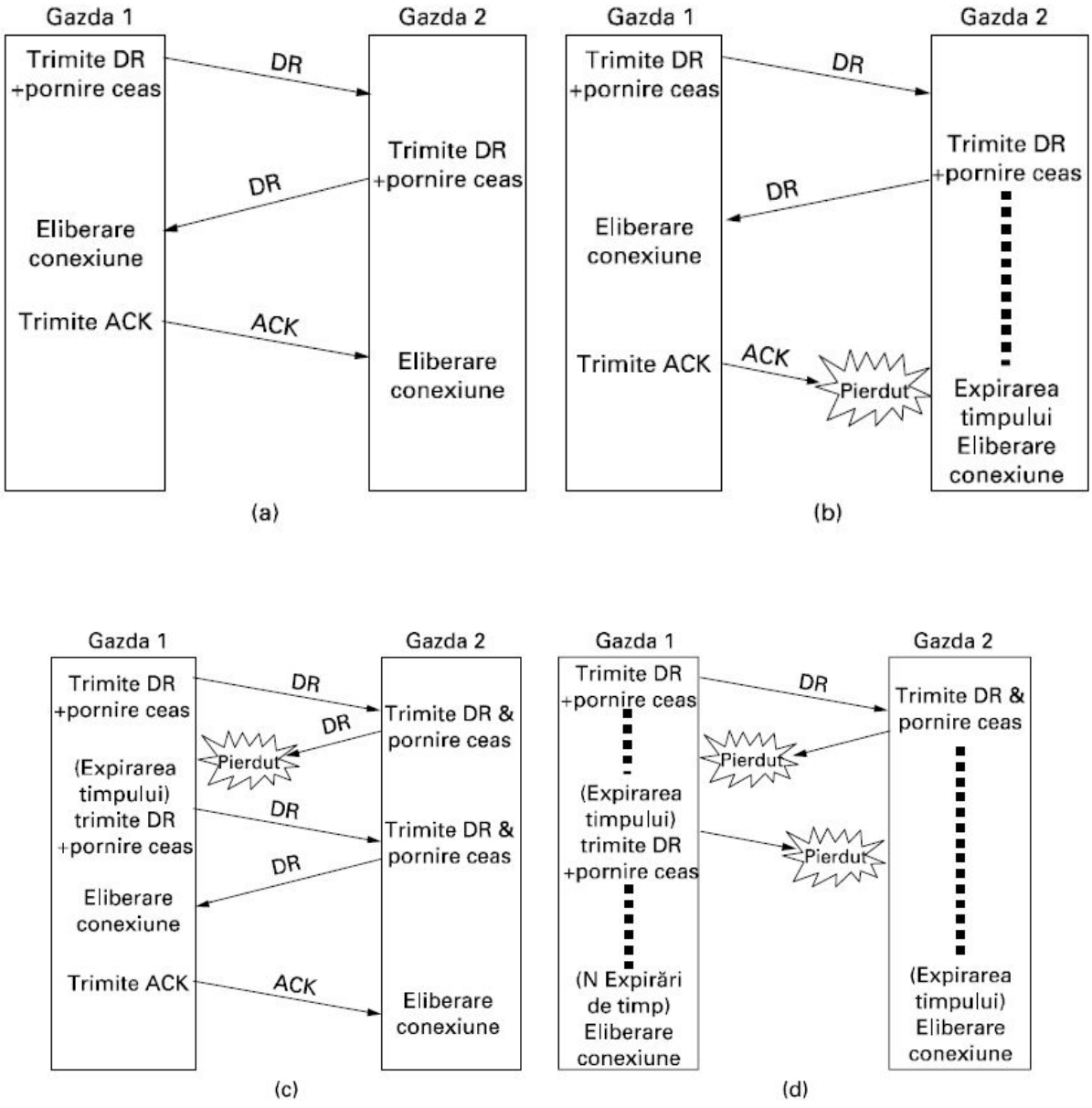


Fig. 6-14. Patru cazuri posibile la eliberarea conexiunii: (a)Cazul normal cu confirmare în trei timpi. (b)Ultima confirmare este pierdută. (c) Răspunsul este pierdut. (d) Răspunsul și următoarele cereri de deconectare sunt pierdute. (DR=DISCONNECT REQUEST).

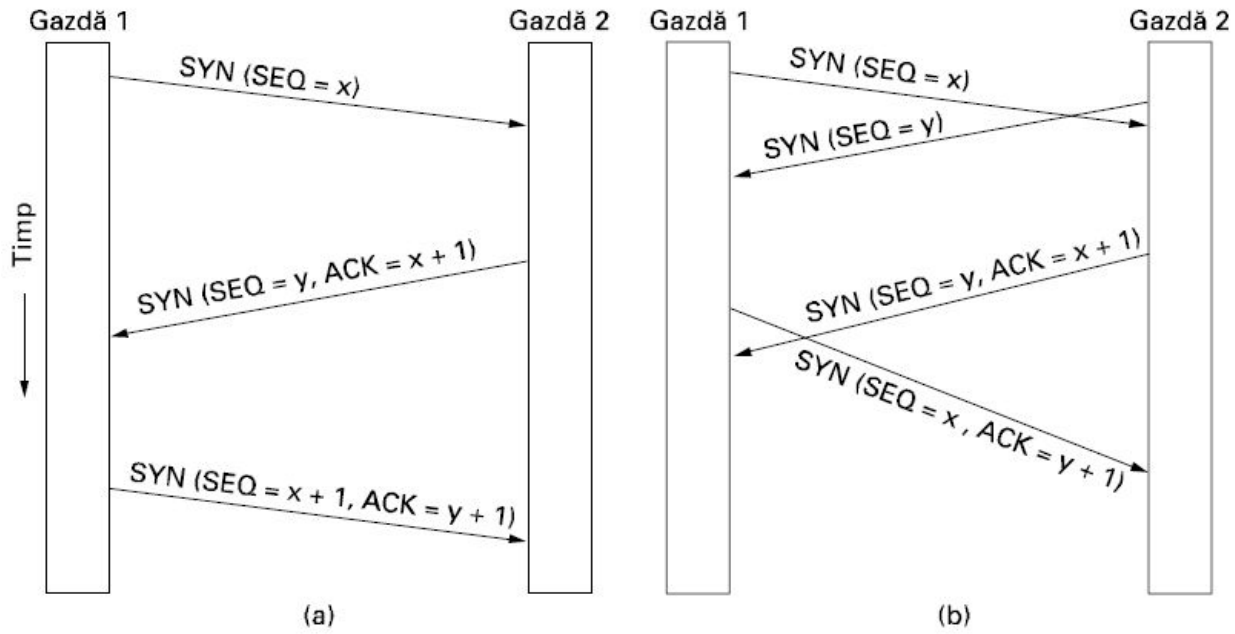


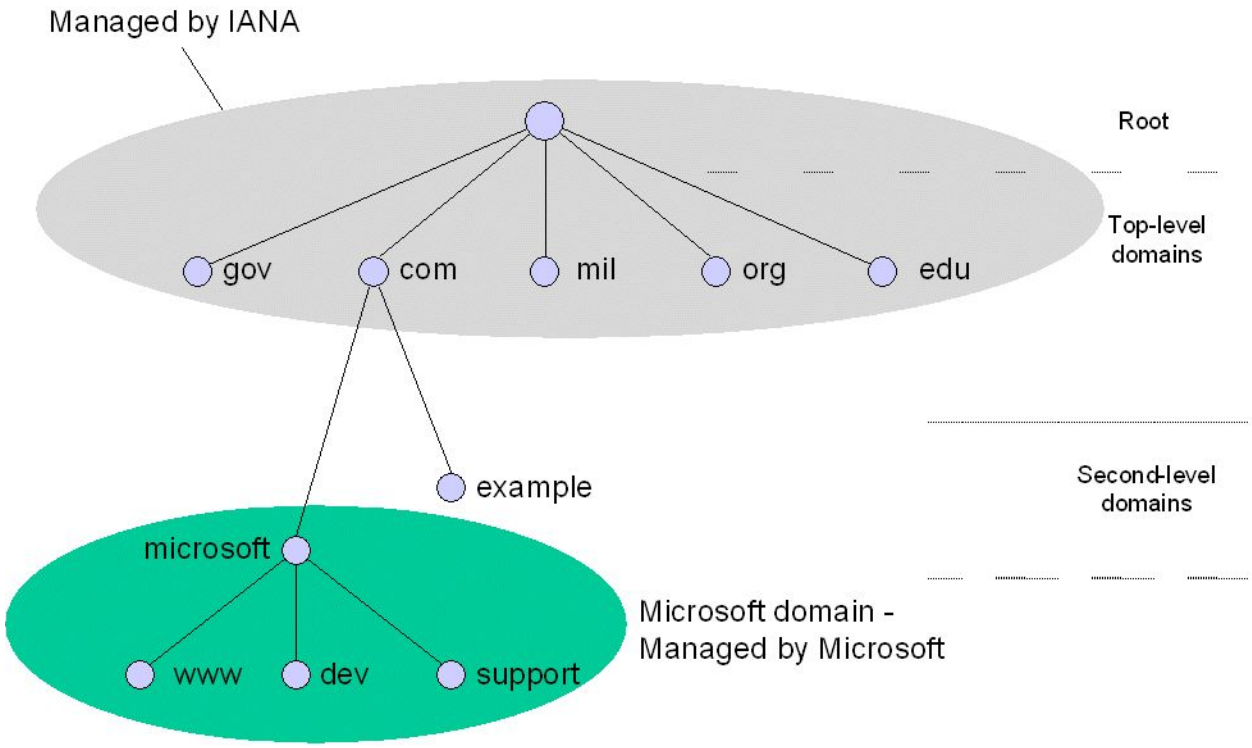
Fig. 6-31. (a) Stabilirea unei conexiuni TCP în cazul normal. (b) Coliziunea apelurilor.

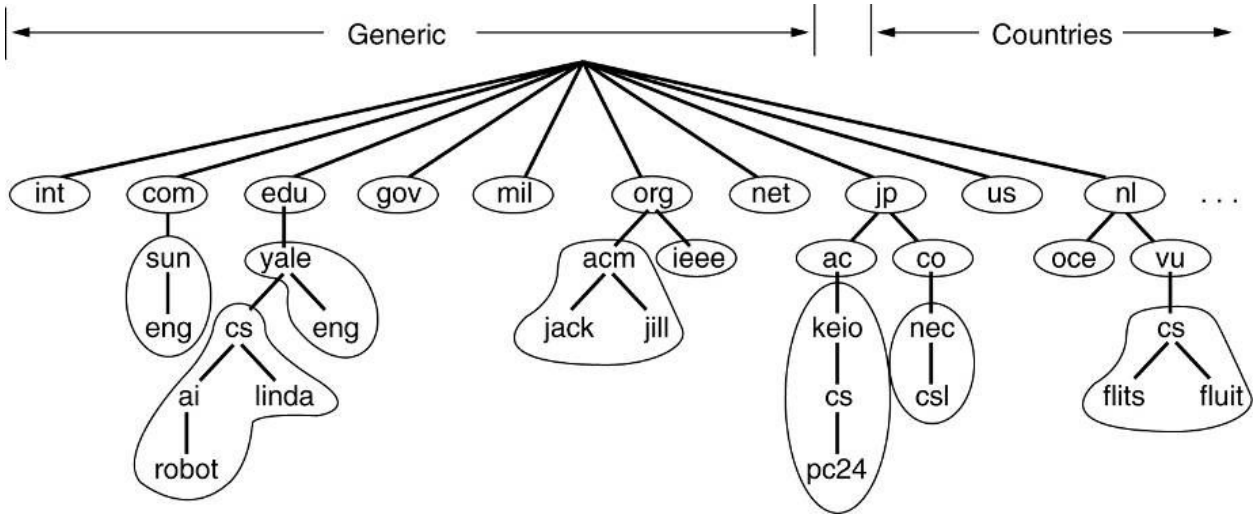
27. DNS

- **Motivatie**: Adresele IP sunt mai greu de tinut minte!
- Se folosesc **adrese simbolice (nume de domeniu)** a caror translatare in adrese IP se face prin **DNS**
- Continute in URL: (adresele)
  - Schema = protocol (http, ftp etc.)
  - Host = nume / adresa IP a serverului Web
  - port# = numar port server Web (80 pt HTTP)
  - Path = calea de la radacina serverului la resursa

Schema	**Utilizat pentru**	**Exemple**
http	Hipertext (HTML)	http://www.cs.vu.nl/~ast
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
mailto	Trimitere de poș ta electronică	mailto:JohnUser@acm.org
telnet	Conectare la distanță	telnet://www.w3.org:80

- Descarcarea unei pagini Web
  - Browser
    - Determina URL http://www.w3.org/TheProject.html
    - Cere unui server DNS adresa IP pentru http://www.w3.org/
      - Server DNS raspunde cu: 18.23.0.23
    - Deschide o conexiune TCP la port 80 pe 18.23.0.23
    - Trimite o comanda la server Web:
      - GET TheProject.html HTTP/1.1 Host: www.w3.org
      - Server Web www.w3.org trimite fisierul TheProject.html
    - Conexiunea TCP este inchisa
    - Afiseaza continutul TheProject.html
- Peste 200 de domenii de nivel superior
  - Impartite in subdomenii s.a.m.d => **arbore**
  - Fiecare are mai multe sisteme gazda (**frunze**)
  - Tipuri:
    - Generice (de nivel inalt):
      - Com (comercial)
      - Edu (educational)
      - Gov (guvernul federal SUA)
      - Int (org internationale)
      - Mil (fortele militare SUA)
      - Net (furnizori Internet)
      - Org (org nonprofit)
    - De tari:
      - O intrare pentru fiecare tara
    - Sunt separate prin ‘dot’ (.)
    - Nu se face distinctie intre litere mici si mari
    - Componentele au cel mult 63 de caractere, iar in total nu depaseste 255 de caracatere
    - “example.com” e diferit de “example.mil”



- 

- Servere **DNS**
  - Administreaza zone **DNS**
  - Pastreaza BD cu informatii necesare clientilor
  - In **inregistrari de resurse**
- **Inregistrari de resurse**
  - Contine:
    - Nume_domeniu (srv1.dev.microsoft.com. - ultimul punct este RADACINA)
    - Timp_de_viata (ex 3600 - in secunde)
      - O zi - stabila
      - Un minut - instabila
    - Clasa (IN - pentru Internet)

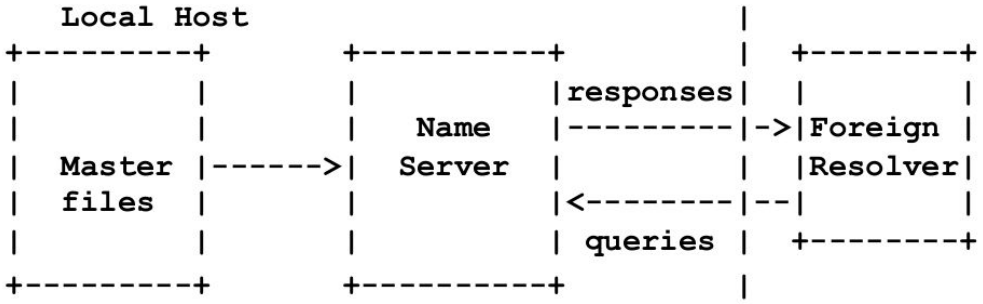
- Tip (A - adresa)

Tip	Semnificație	Valoare
SOA	Start autoritate	Parametrii pentru această zonă
A	Adresa IP a unui sistem gazdă	Întreg pe 32 de biți
MX	Schimb de poștă	Prioritate, domeniu dispus să accepte poștă electronică
NS	Server de Nume	Numele serverului pentru acest domeniu
CNAME	Nume canonic	Numele domeniului
PTR	Pointer	Pseudonim pentru adresa IP
HINFO	Descriere sistem gazdă	Unitate centrală și sistem de operare în ASCII
TXT	Text	Text ASCII neinterpretat

Fig. 7-2. Principalele tipuri de înregistrări de resurse DNS.

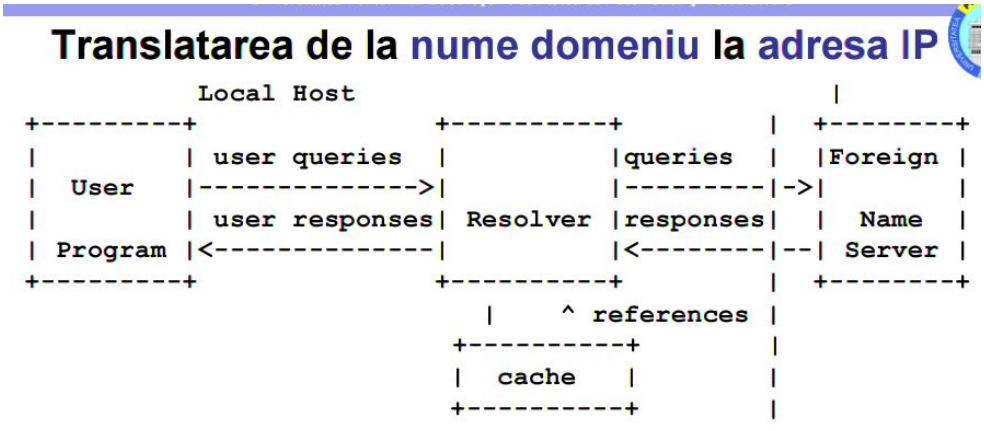
- Valoare (157.60.221.205)

- **Server de nume**



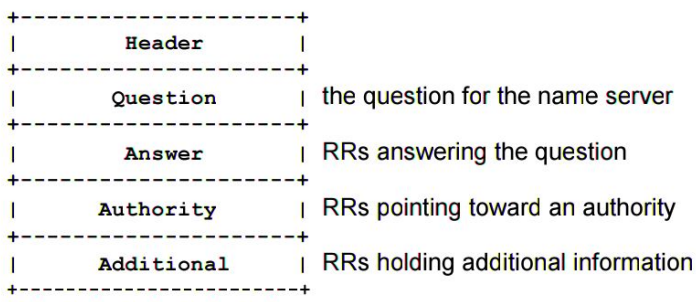
- Informatia **primara** este pastrata in fisiere **master** aflate in sistemul de fisiere local al **DNS**
- Server de nume **primar:**
  - Foloseste **master files** pentru a actualiza BD
  - Raspunde intrebarilor resolverilor

- Translatarea de la **nume domeniu** la **adresa IP**



- Se apeleaza un **rezolver** local
  - Formatul mesajelor **user ⇔ resolver** este specific sistemului gazda (ex UNIX: gethostbyname)
  - Apeleaza la randul sau un **server DNS** local (ii cunoaste **adresa IP**) cu un mesaj denumit **DNS request** => **DNS reply** din partea serverului
  - Formatul mesajelor **resolver ⇔ name server** este standard (protocol DNS)
  - Poate pastra in **cache** numele si adresele IP recent rezolvate
    - Perioada este data de **time-to-live** din **inregistrari DNS**

- **Format mesaje DNS**

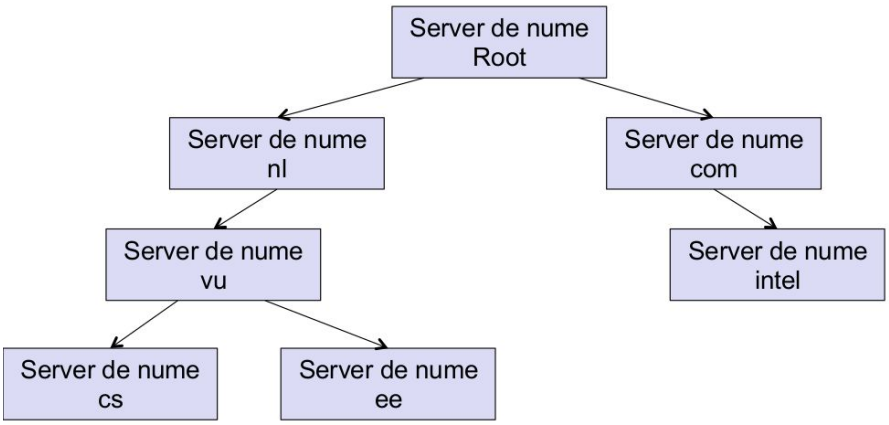


RR = Resource Record

- Header contine info despre:
  - Ce sectiuni sunt prezente in mesaj
  - Mesajul este **intrebare** sau **raspuns**
  - Alta operatie
- Question - intrebarea
  - Tuplu **Nume_domeniu, Tip, Clasa**
  - Singurul camp inclus in intrebare
- Answer - raspunsul
  - Include RRs care corespund intrebarii
- Restul - colectie de RRs reprezentand autoritatea si info aditionale

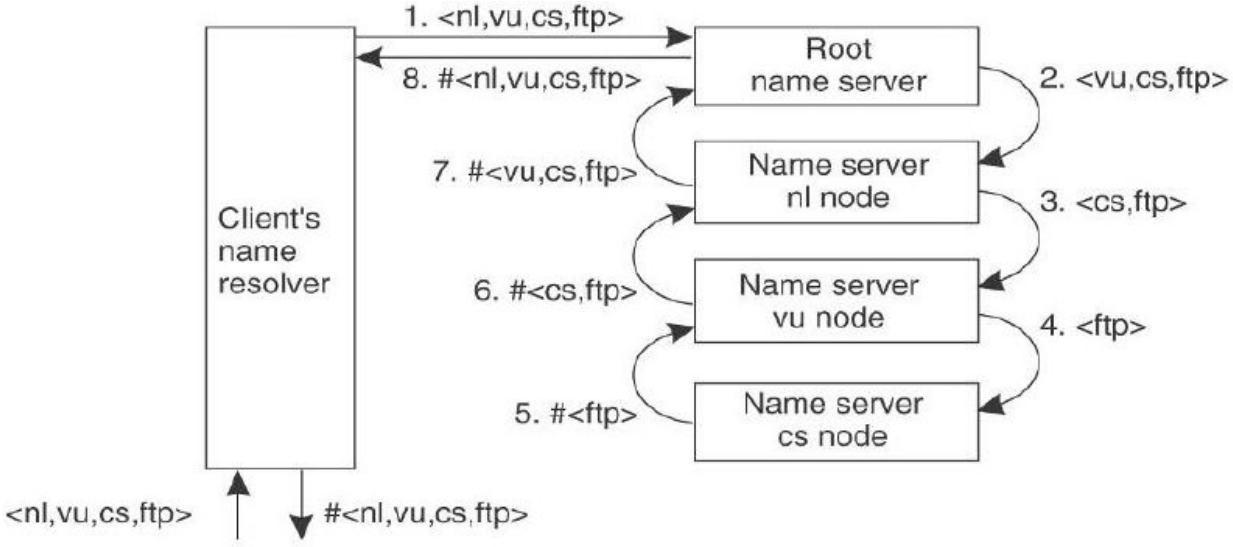
- Un **server DNS** este **server autoritate** pentru numele gestionate
  - Daca cererea conține un nume gestionat de serverul apelat, acesta răspunde direct (**ai.cs.yale.edu** este sub **cs.yale.edu**)
  - Altfel, cererea trebuie sa ajunga la serverul autoritate pentru acel nume

- O posibila ierarhie de **servere DNS**

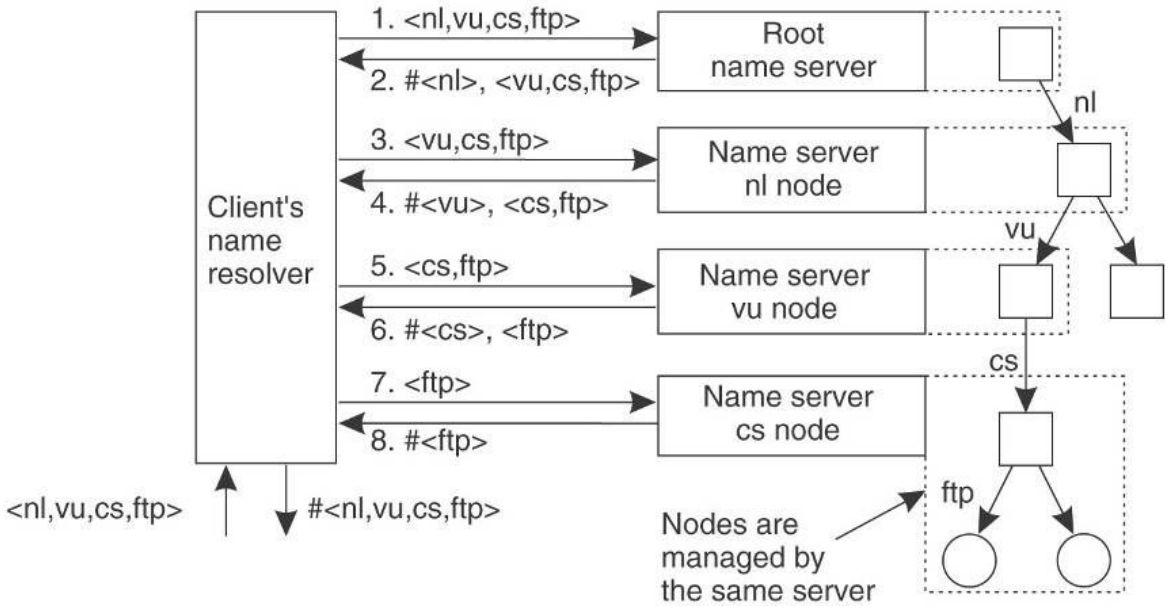


- adresele ptr nume de **top** (nl, com) sunt stiute de **root**
- exista mai multe servere **root**, adresele lor IP fiind copiate, din fisiere de **config** in **cache** DNS, la pornirea serverului DNS

- **Rezolvare recursiva**
  - Cererea este pasata de la un server DNS la altul pana ajunge la serverul DNS care rezolva numele din cerere
  - Raspunsul este trimis pe calea inversa
  - Cand resursele ajung inapoi ele vor fi puse in **memoria ascunsa** pentru a putea fi folosite ulterior (neautorizat deoarece orice schimbare facuta la un domeniu se va propaga la toate serverele care l-au folosit) => campul Timp_de_viata din RRs



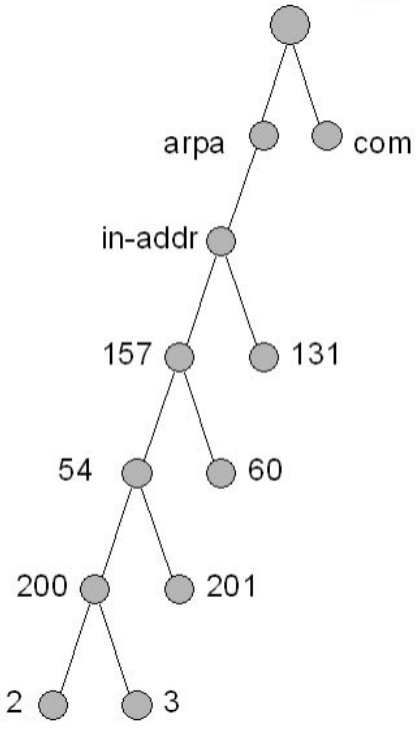
- **Rezolvare iterativa**
  - daca serverul DNS nu poate rezolva intregul nume, el trimite clientului partea nerezolvata si adresa serverului DNS care o poate rezolva
  - clientul trimite o noua cerere acestui server DNS



❑

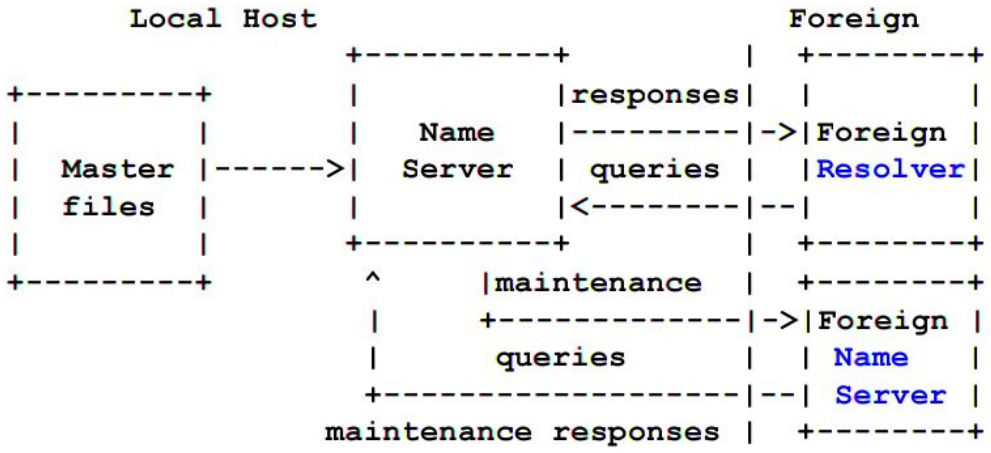
Cereri inverse

- Cauta nume pentru adresa IP 157.54.200.2
- Organizare - un domeniu special
  in-addr.arpa
  in care nodurile sunt numite dupa numerele din adresa IP
- In in-addr.arpa se creaza inregistrari PTR, in care numele sunt adrese IP
- Clientul face o cerere PTR pentru numele 2.200.54.157.in-addr.arpa
- Cautarea se face in inregistrari PTR si intoarce numele resursei care corespunde adresei IP 157.54.200.2, de ex. mail.alfa.com.
- Aplicatie: in tracert – pentru afisare nume rutere



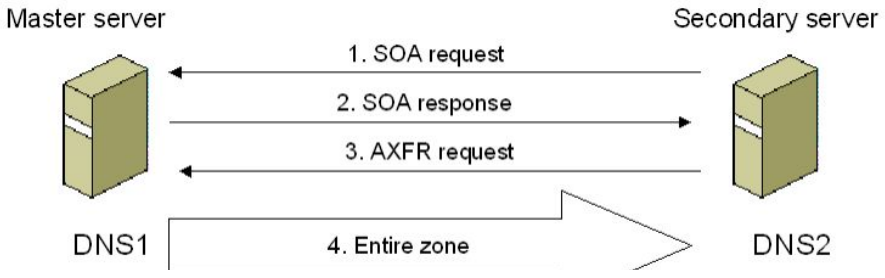
Replicarea serverelor DNS

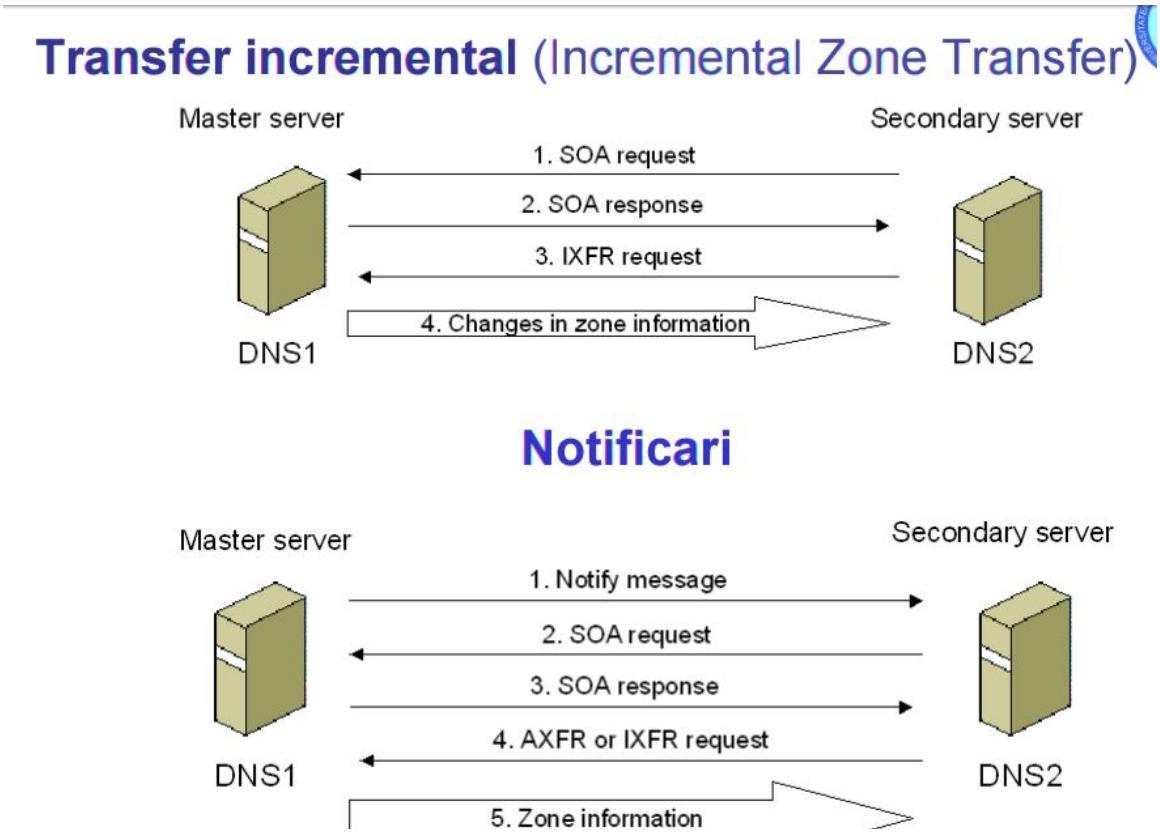
- Fiecare zona trebuie sa aiba mai multe servere DNS
- Server Primar – pe el se fac toate modificarile inregistrarilor, folosind Master files
- Secundar – preia info de la servere primare
  - pentru asta, foloseste acelasi format de mesaje DNS



Facilitati – transfer toata zona

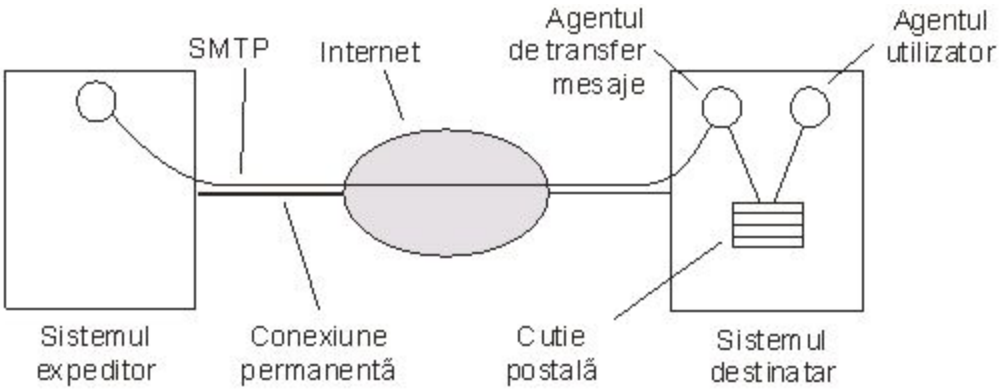
- Server secundar
  - (periodic) Cerere SOA (Start Of Authority)
  - Primeste raspuns si verifica daca "serial number" este mai mare decat cel local
  - Daca da, cere toata zona (cerere AXFR – Authoritative transfer)
  - Primeste info toata zona





28. Posta electronica (Mail)

- Arhitectura sistemului de e-mail

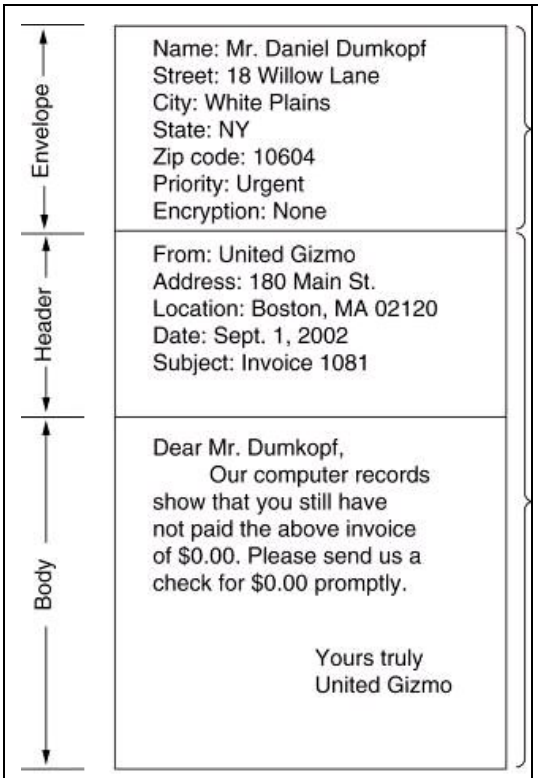


- **Agent utilizator**
  - permite citirea si scrierea mesajelor
  - interfata utilizatorului cu sistemul de e-mail
- **Agentul de transfer mesaje**
  - suporta transmiterea mesajelor de la sursa la destinatie

- ❏ agentul client preia un mesaj, stabileste o conexiune cu agentul server si ii transmite mesajul
- ❏ agentul server primeste mesajul si il plaseaza in cutia postala
- ❏ agenti = demoni de sistem care ruleaza in fundal

❏ Mesaje si plicuri:

❏



plic = informatia necesara pentru transportul mesajului, folosita in protocolul SMTP
- ❏ Adresa destinatar
- ❏ Prioritatea
- ❏ Nivel securitate

antet = informatia de control pentru agentul utilizator
- ❏ perechi nume-valoare referitoare la utilizatori si la continutul mesajului

corp = informatia destinata utilizatorului

❏ **Adrese e-mail:**
- ❏ nume_utilizator@nume_server_mail
- ❏ Nume_server_mail
  - ❏ este numele de domeniu
  - ❏ folosit de clientul de e-mail care:
    - ❏ rezolva numele destinatarului folosind DNS (MX, daca se poate)
    - ❏ contacteaza serverul de e-mail de la destinatie
    - ❏ transmite mesajul la server
- ❏ Nume_utilizator
  - ❏ are un specific local; ex: droms, Ralph_E._Droms, 578.4309
  - ❏ folosit de serverul de mail care:
    - ❏ primeste mesajul de la client
    - ❏ interpreteaza nume_utilizator conform cu adresele locale
    - ❏ plaseaza mesajul in cutia postala corespunzatoare

❏ Continutul unei cutii postale:

#	Flags	Bytes	Sender	Subject
1	K	1030	asw	Changes to MINIX
2	KA	6348	trudy	Not all Trudys are nasty
3	K F	4519	Amy N. Wong	Request for information
4		1236	bal	Bioinformatics
5		104110	kaashoek	Material on peer-to-peer
6		1223	Frank	Re: Will you review a grant proposal
7		3110	guido	Our paper has been accepted
8		1204	dmr	Re: My student's visit

- **K** – Kept – mesaj pastrat in cutia postala (mesajul nu este nou)
- **A** – Answered – mesaj la care s-a raspuns
- **F** – Forwarded – mesaj retransmis altui utilizator

- Formatul mesajelor:

Antet	Conţinut
To:	Adresa(ele) de e-mail a(le) receptorului(ilor) primar(i)
Cc:	Adresa(ele) de e-mail a(le) receptorului(ilor) secundar(i)
Bcc:	Adresa(ele) de e-mail pentru „blind carbon copy"
From:	Persoana sau persoanele care au creat mesajul
Sender:	Adresa de e-mail a transmiţătorului curent
Received:	Linie adăugată de fiecare agent de transfer de-a lungul traseului
Return-Path:	Poate fi folosită pentru a identifica o cale de întoarcere la transmiţător

Campuri din antet care se refera la transportul mesajului.

Unele campuri sunt folosite de agentul de transfer pentru a alcatui plicul.

- Formatul mesajelor este descries in RFC 5322

- Campuri folosite de **agentul utilizator** sau de **utilizator**

Antet	Conţinut
Date:	Data şi momentul de timp la care a fost trimis mesajul
Reply-To:	Adresa de e-mail la care ar trebui trimise răspunsurile
Message-Id:	Număr unic, utilizat ulterior ca referinţă pentru acest mesaj (identificator)
In-Reply-To:	Identificatorul mesajului al cărui răspuns este mesajul curent
References:	Alţi identificatori de mesaje ale caror raspunsuri sunt mesajul current, mesajul precedent etc.
Keywords:	Cuvinte cheie alese de utilizator
Subject:	Scurt cuprins al mesajului, afişabil pe o singură linie

- ❏
- ❏ Antete adaugate de MIME - Multi-Purpose Internet Mail Extensions

Antet	Conţinut
MIME-Version:	Identifică versiunea de MIME
Content-Description:	Descrierea a ce este în mesaj (similar subiectului)
Content-Id:	Identificator unic al continutului
Content-Transfer-Encoding:	Cum este codificat cotinutul pentru transmisie
Content-Type:	Tipul datelor continute in mesaj

- ❏

MIME

Mesaj cu mai multe componente

```
From: elinor@abcd.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@abcd.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/enriched

Happy birthday to you
Happy birthday to you
Happy birthday dear <bold> Carolyn </bold>
Happy birthday to you

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
        access-type="anon-ftp";
        site="bicycle.abcd.com";
        directory="pub";
        name="birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

- ❏
- ❏