



# Internet

## Modele arhitecturale de referință



# Cuprins

- Arhitectura retelelor de calculatoare
- Modelul de interconectare a sistemelor deschise - ISO-OSI
- Rolul ierarhiei de protocoale
- Formatul datelor – antet și continut
- Servicii și primitive de serviciu
- Modelul TCP/IP

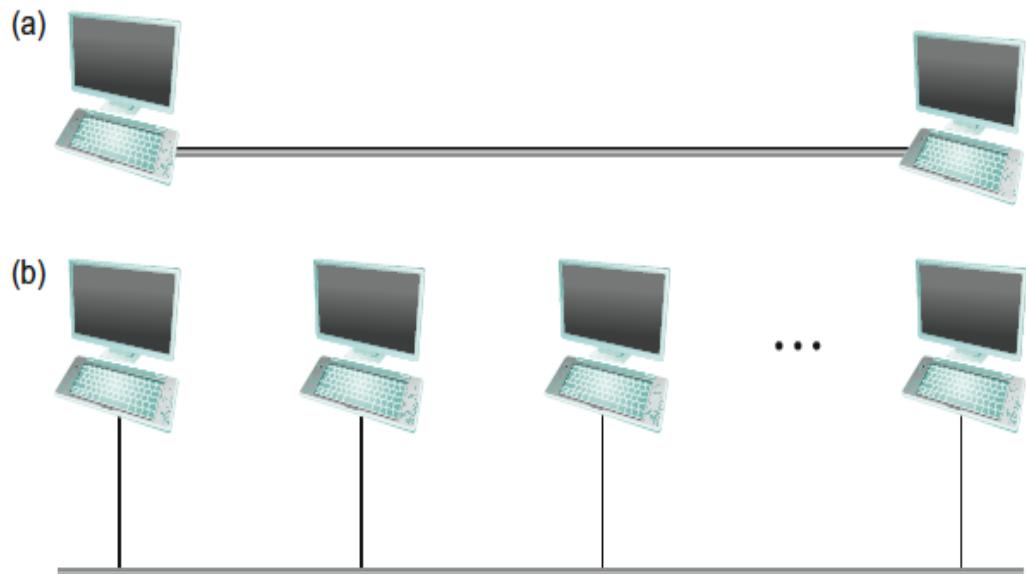
## Obiective:

- Notiuni fundamentale utilizate în restul cursului
- Imagine de ansamblu a protocoalelor ce vor fi studiate



# Comunicarea în aplicații Internet

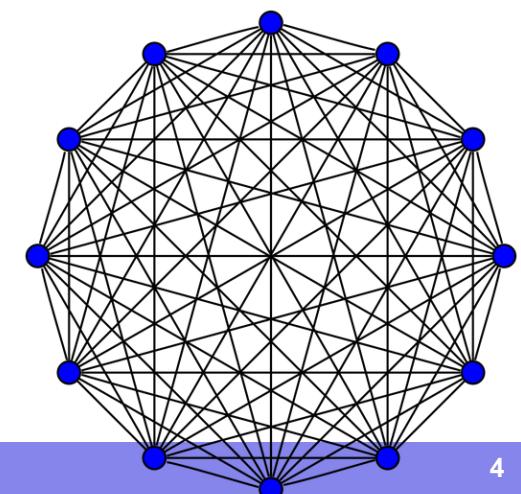
- Internetul este cunoscut de utilizatori prin aplicațiile sale
  - e-Mail
  - Transferul de fisiere
  - Web
  - și multe altele
- Bazate pe **comunicarea** prin legături între calculatoare
- **Legătura** poate fi directă, între două sau mai multe calculatoare (noduri)
  - punct la punct – prin fir
  - acces multiplu – de ex wireless (retele celulare sau Wi-Fi)





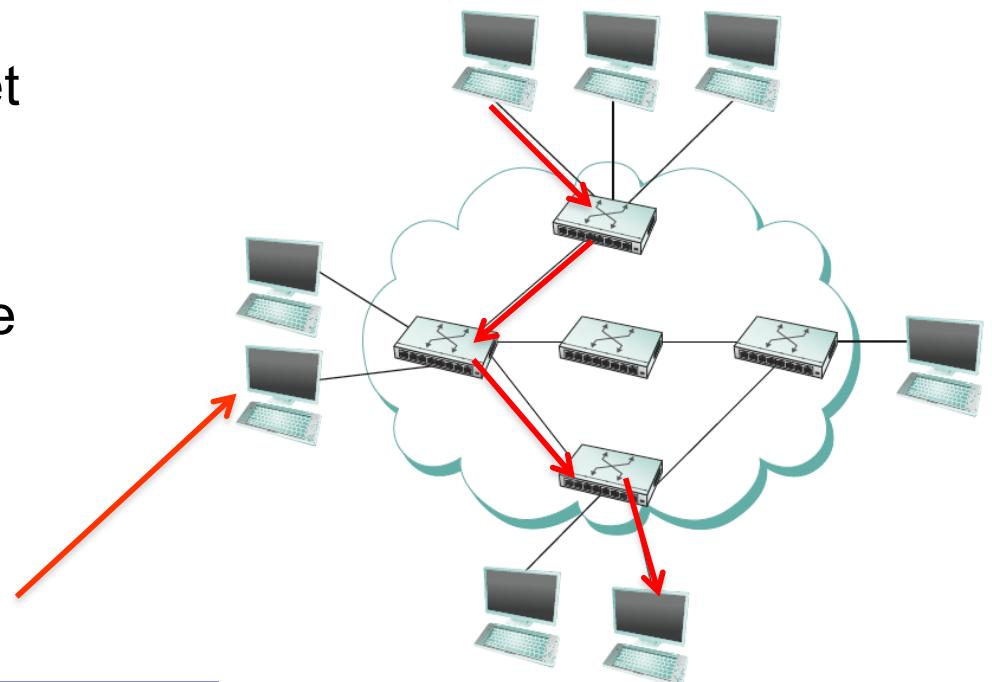
# Legatura de date

- Legaturile asigură funcționalități specifice **datelor** (diferite de cele pentru comunicări vocale)
  - codificarea bitilor pentru transmisie și înțelegerea codurilor la receptie
  - **detectia** erorilor de transmisie și **corectarea** lor
    - delimitarea sirurilor de biti (**incadrarea**) care constituie mesaje complete (**cadre**) și atașarea unor informații de **control**
  - controlul **fluxului** de date
  - controlul accesului la media (pentru acces multiplu)
- Legaturile directe între noduri  
**nu oferă scalabilitate**



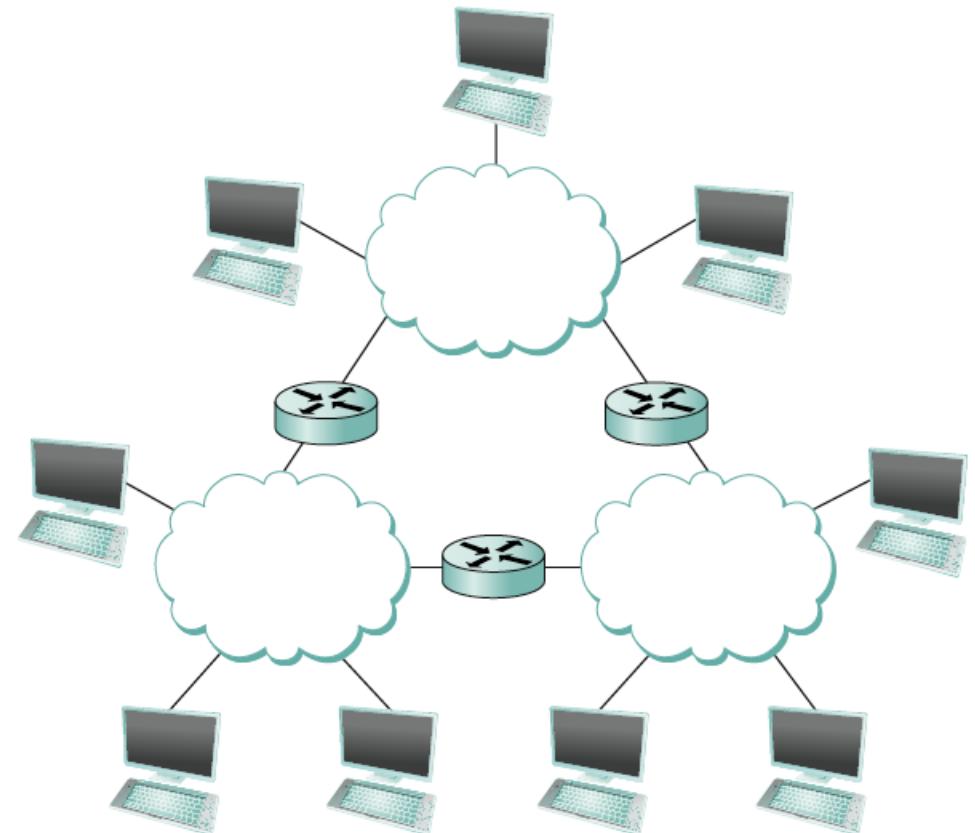
# Retele

- Conexiunea intre doua noduri se poate face si **indirect**, prin **noduri intermediare** (switch-uri)
  - atasate la mai multe legaturi
  - au functii speciale
  - formeaza o retea
- Functioneaza dupa principiul sistemului **postal**; un nod
  - primeste cate un bloc complet de date (**pachet**)
  - il **memoreaza** temporar
  - il **re-transmite (dirijeaza)** catre destinatie, printr-una din legaturile la nodurile vecine
- Nodurile din afara retelei sunt numite **gazde (ale aplicatiilor)**
- Pentru ca reteaua sa functioneze, fiecarui nod i se ataseaza o **adresa** de retea pentru identificare
- Nodul **sursa** include in pachet **adresa** nodului **de destinatie**



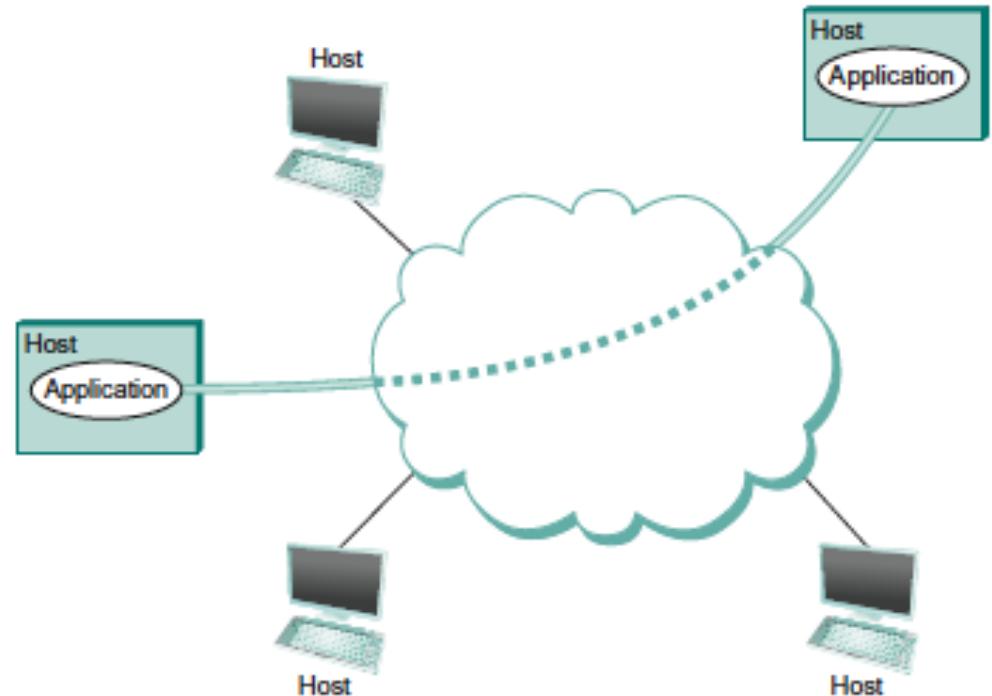
# Retele interconectate

- Nodurile intermediare pot fi organizate în mai multe rețele interconectate
- Un nod conectat la mai multe rețele este numit **ruter** (sau **gateway**)
  - are funcții similare cu switch-ul
- Rețelele se bazează pe buna funcționare a **legăturilor** dintre noduri
- Rețelele transmit pachete între oricare două noduri neconectate direct
- Nu asigură **corectitudinea** transmisiei
  - e.g. pachete pierdute



# Comunicarea între aplicații

- Reteaua trebuie să asigure comunicarea **între aplicații** din calculatoare diferite, adică să ofere **canale “logice”** prin care **procesele** de aplicatie să comunice între ele
- Aceasta presupune facilități suplimentare peste cele de retea
- **Identificarea** unică a fiecarui **capăt de canal logic** printr-o pereche (adresa de retea, **port**)
- mecanisme de **trimitere / primire** de mesaje de către procese
- garantarea **recepției corecte** a mesajelor
- livrarea mesajelor în **ordinea** în care au fost transmise
- pastrarea **confidențialității**
- etc.





# Cerintele aplicatiilor

- Modul în care este folosit un canal logic difera de la o aplicație la alta
- În **Web**, comunicarea se face între două procese
  - la comanda unui utilizator, un proces **client** (browser) trimite o **cerere** către un **server** Web – un mesaj care include identificatorul paginii dorite
  - procesul server trimite un răspuns către client – un mesaj care conține pagina Web solicitată, pe care clientul o afișază pe ecran
- În aplicații de livrare de continut **audio/video**
  - transmiterea cererii este similară cu Web
  - livrarea este diferită
    - se trimite ca răspuns o serie de mesaje
    - continutul este livrat utilizatorului pe măsură ce mesajele sunt primite
    - impune respectarea unor constrângeri de timp



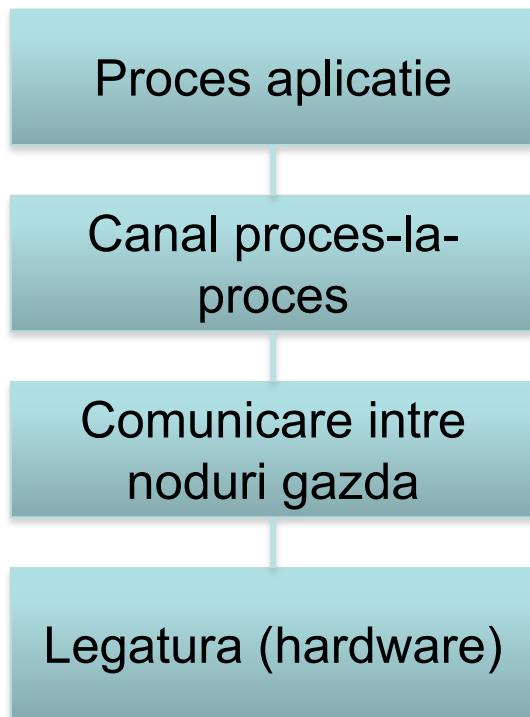
# Arhitectura Retelelor de Calculatoare

- Arhitectura descrie
  - modul în care componentele sunt organizate și
  - felul în care ele interacționează
- Retelele de calculatoare sunt sisteme complexe
- Abordarea lor ca ansambluri de componente logice simplifică înțelegerea și realizarea
- Pentru retele de calculatoare avem componente pentru
  - comunicarea pe o legătură de date
  - comunicarea între două noduri gazda
  - comunicarea pe canale logice



# Arhitectura ierarhica

- Componentele care asigura comunicarea intre doua noduri gazda folosesc **serviciile** oferite de componentele pentru o legatura de date si adauga noi functionalitati (dirijarea, adresarea...)
- Arhitectura este o **ierarhie** de **nivele** functionale



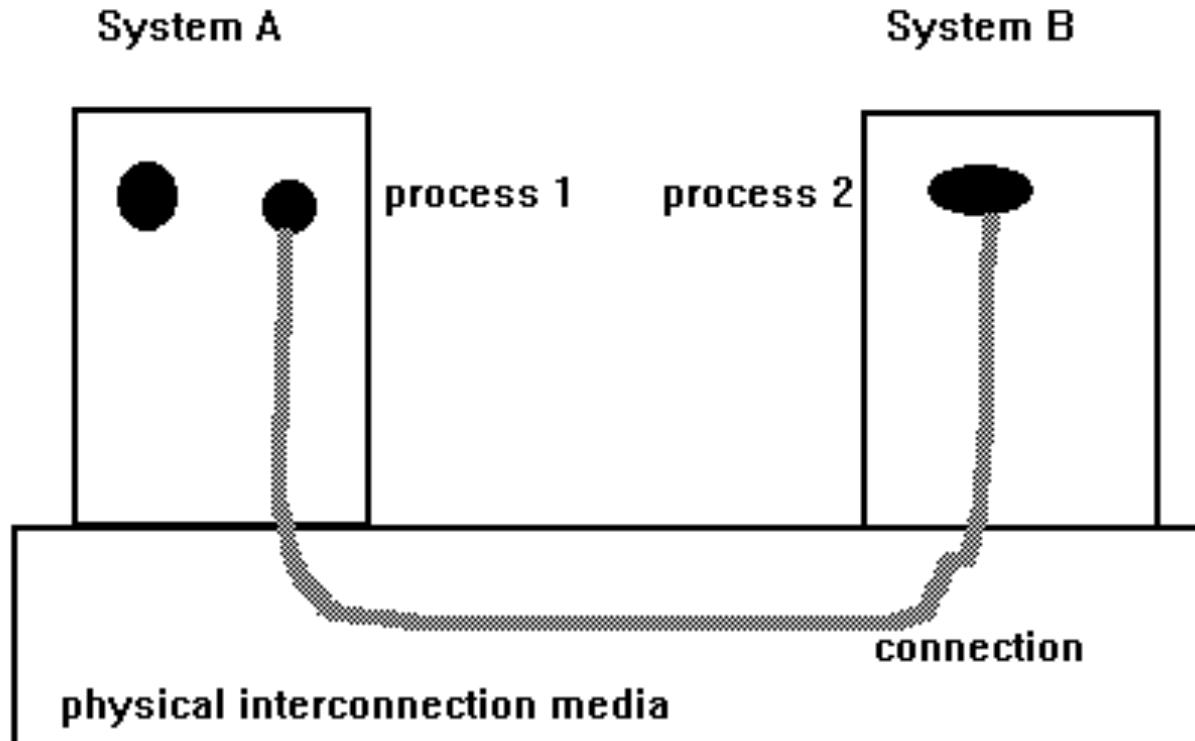
In figura se arata ierarhia componentelor dintr-un nod gazda

Fiecare **componentă** apartine unui **nivel** diferit

Fiecare nod gazda are o structura similară

# Componențe de bază

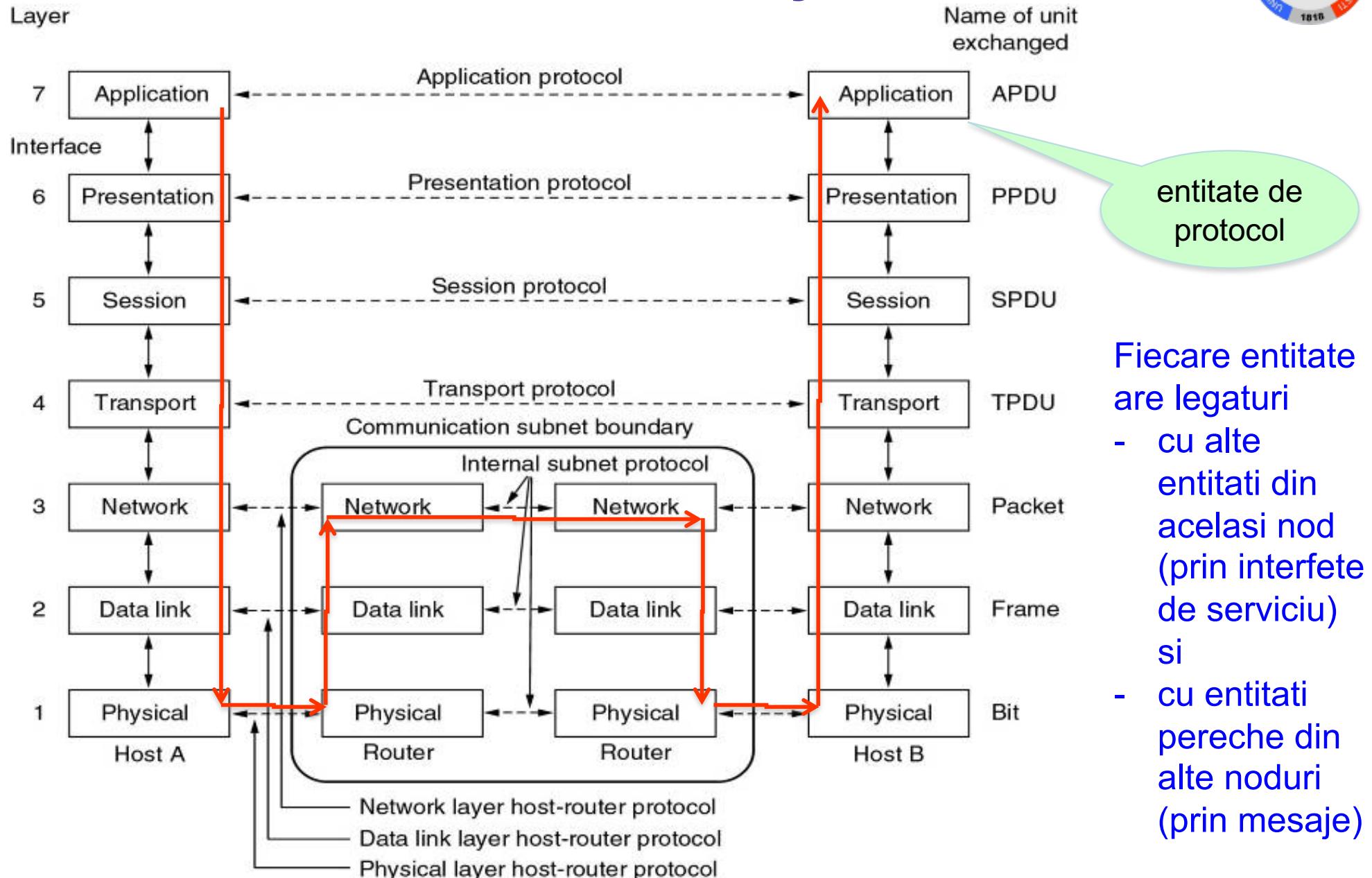
- Componențe de bază ale unui model de comunicare:



**Protocol** – set de reguli respectate de parti pentru a comunica între ele; se referă la

- **formatul** mesajelor comunicate: continut + meta-descriere (antet)
- **operatiile** executate
  - trimite cerere conectare, primește răspuns, confirma răspuns etc.

# Modelul de Referință ISO OSI





# Nivel fizic

- **Funcție** - transmitere a sirurilor de biți pe un canal de comunicație
- Principalele **probleme**
  - codificarea zerourilor și a unităților
  - stabilirea și desființarea conexiunilor fizice
  - modul de transmisie (semiduplex sau duplex) etc.
- **Exemplu**
  - 802.11 Wi-Fi



# Legătura de date

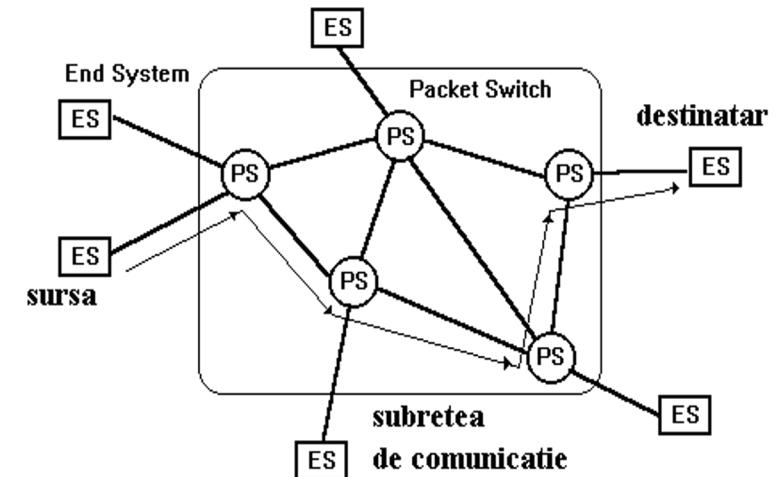
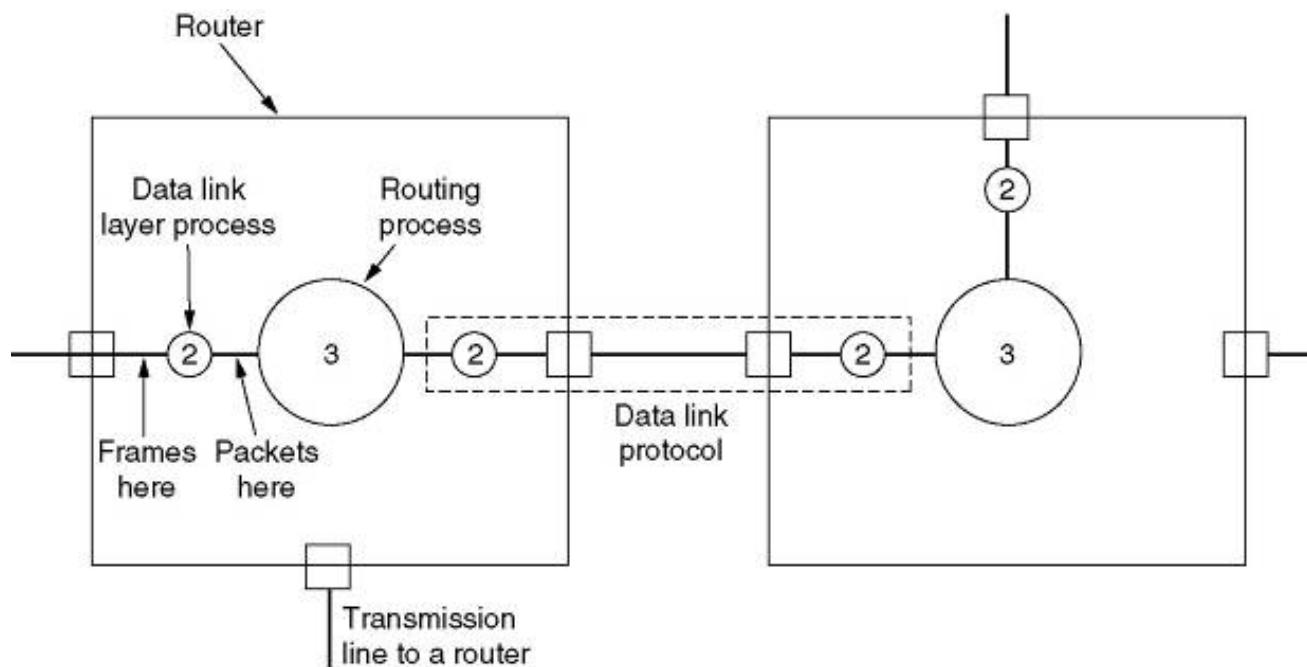
- **Funcție** - realizează **agregarea** unor siruri de biti în **cadre** și **comunicarea lor sigură și eficientă** între două noduri adiacente (conectate printr-un canal fizic de comunicație)
- **Probleme rezolvate**
  - Încadrare
  - Control erori
  - Control flux
  - Transmisie transparentă
  - Management legătură
- **Exemplu** de protocol: HDLC (High Level Data Link Control)

flag    address    command    data    FCS    flag

- **Implementare** prin
  - adaptoare de rețea
  - drivere din sistemul de operare al calculatorului

# Nivel rețea

- **Funcție** - transmiterea pachetelor între oricare două noduri din rețea
  - prin dirijarea lor de la un nod la altul
- **Probleme rezolvate**
  - alegerea legăturii următoare (dirijarea)
  - adresarea
  - calculul tabelelor de dirijare





# Nivel Transport

**Funcție** - asigura un transfer de date corect, eficient **între procese** din sistemul sursă și din sistemul destinatar

## Oferă

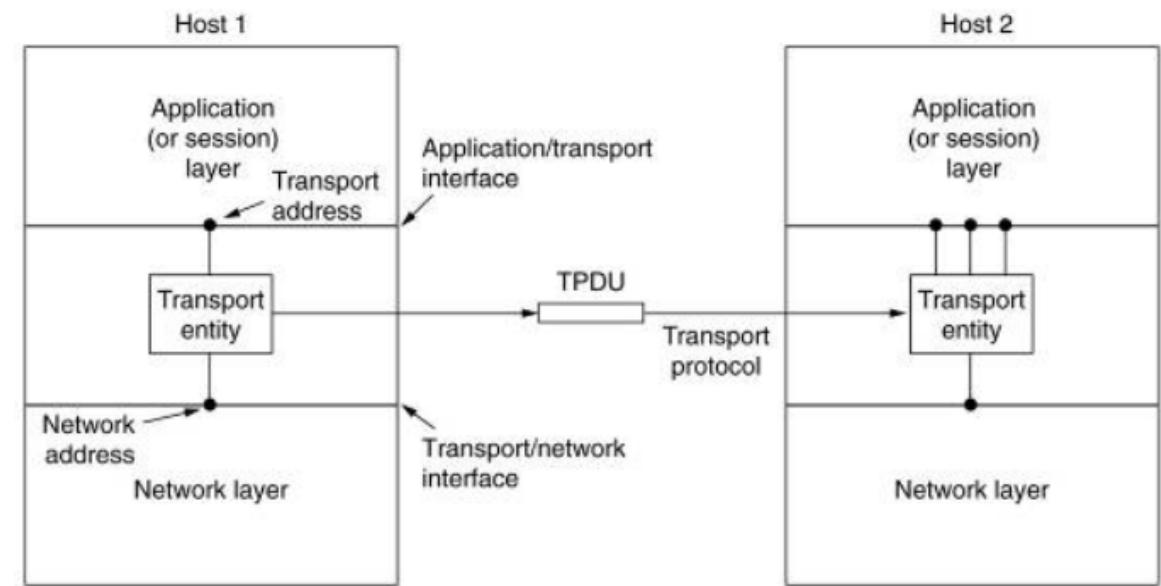
- un transfer sigur al datelor, chiar cu o rețea nesigură;
- o **interfață uniformă** pentru nivelul superior, independent de tipul rețelei utilizate.

## Separă două categorii de nivele

- furnizorul serviciilor de transport (nivele 1-4)
- utilizatorul serviciilor de transport (nivele 5-7)

## Probleme

- gestiunea conexiunilor
- transferul datelor
- controlul fluxului
- adresarea





# Nivele superioare

- **Nivel Sesiune**
  - Controlul dialogului între aplicații
  - Sincronizarea transferurilor
  - Stabilirea unor puncte de verificare și reluare a transferurilor
- **Nivel Prezentare**
  - Conversia formatului datelor între
    - sintaxa folosită de aplicații și
    - sintaxa de transfer



# Nivel Aplicație

- Servicii comune unor categorii de aplicații
  - Mesagerie
  - Transfer de Fișiere
  - Terminal Virtual
  - Serviciu de Directoare

# Ierarhii de protocoale

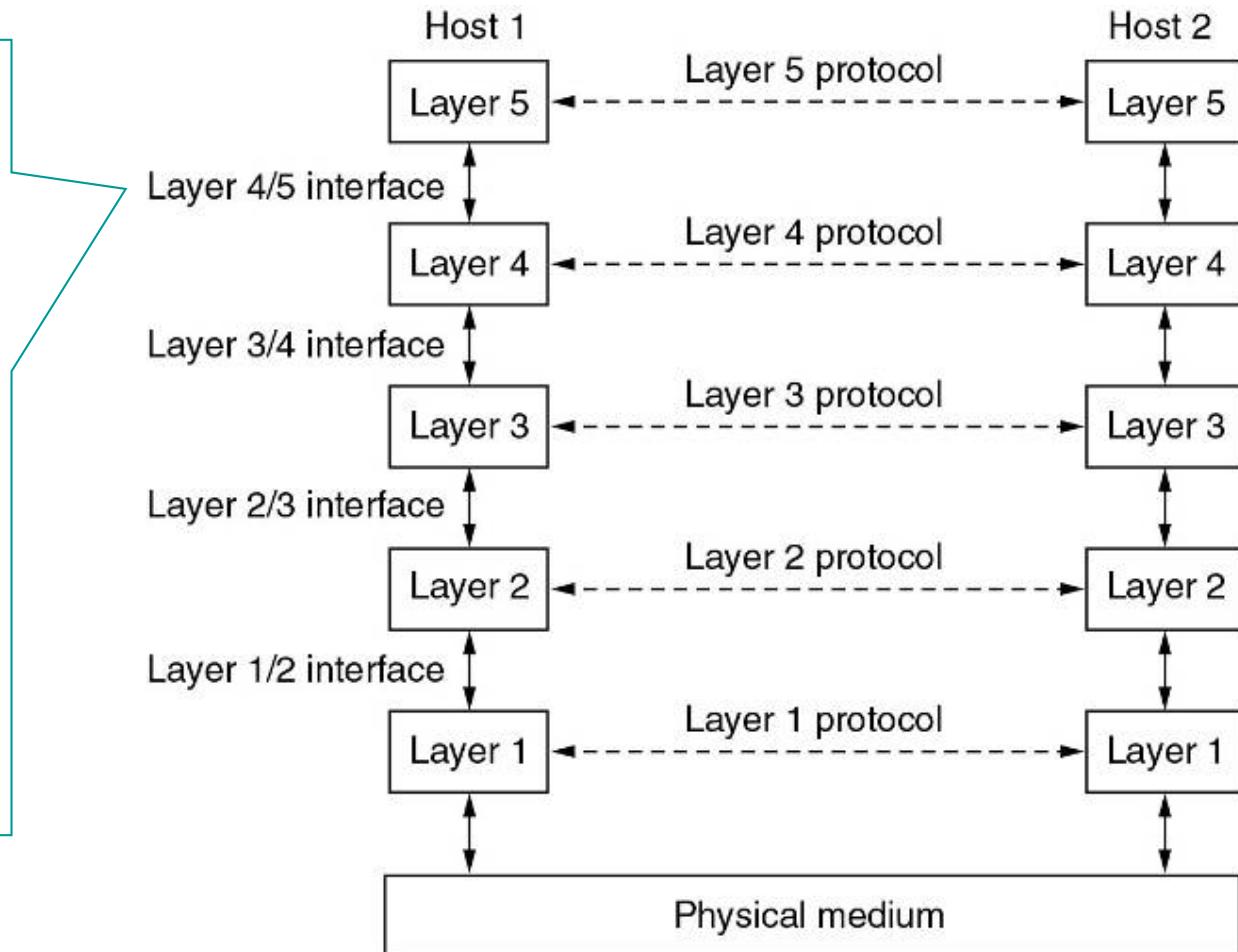
Complexitatea **conexiunii** → organizarea pe mai multe **nivele** cu funcții distincte

- **arhitectura retelei** = setul de nivale și protocoale
- **stiva de protocoale** = lista ordonată a protocoalelor folosite

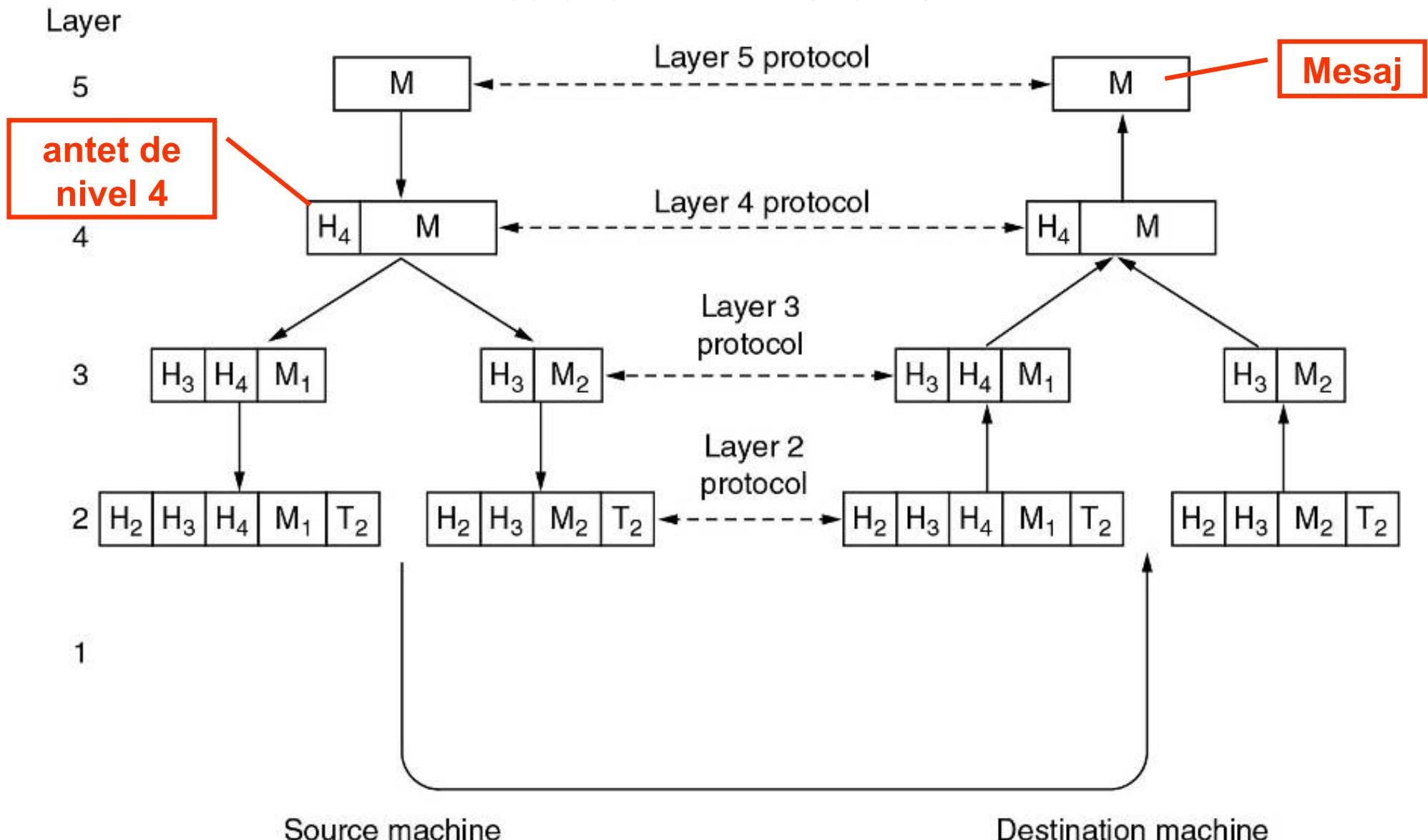
Interfața definește **serviciul (operatiile primitive)** pe care un nivel îl oferă nivelului de deasupra

Ex. de primitive:

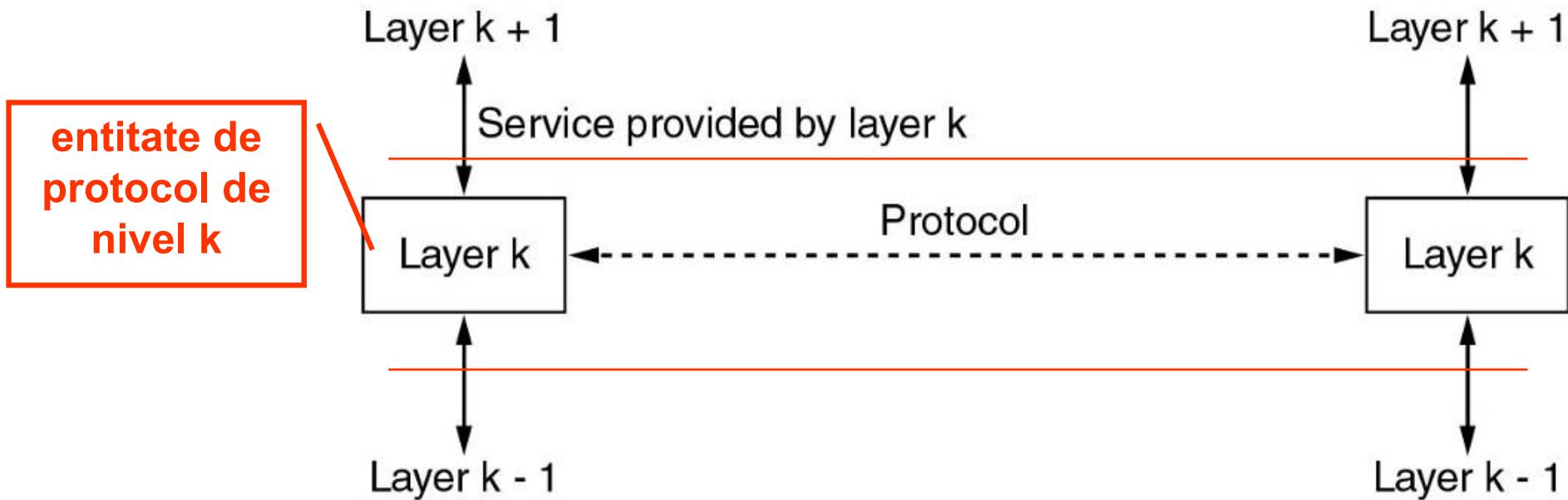
- connect
- accept
- send
- receive
- disconnect



# Flux de informație suportând o comunicare virtuală în nivelul 5



# Relația între servicii și protocoale



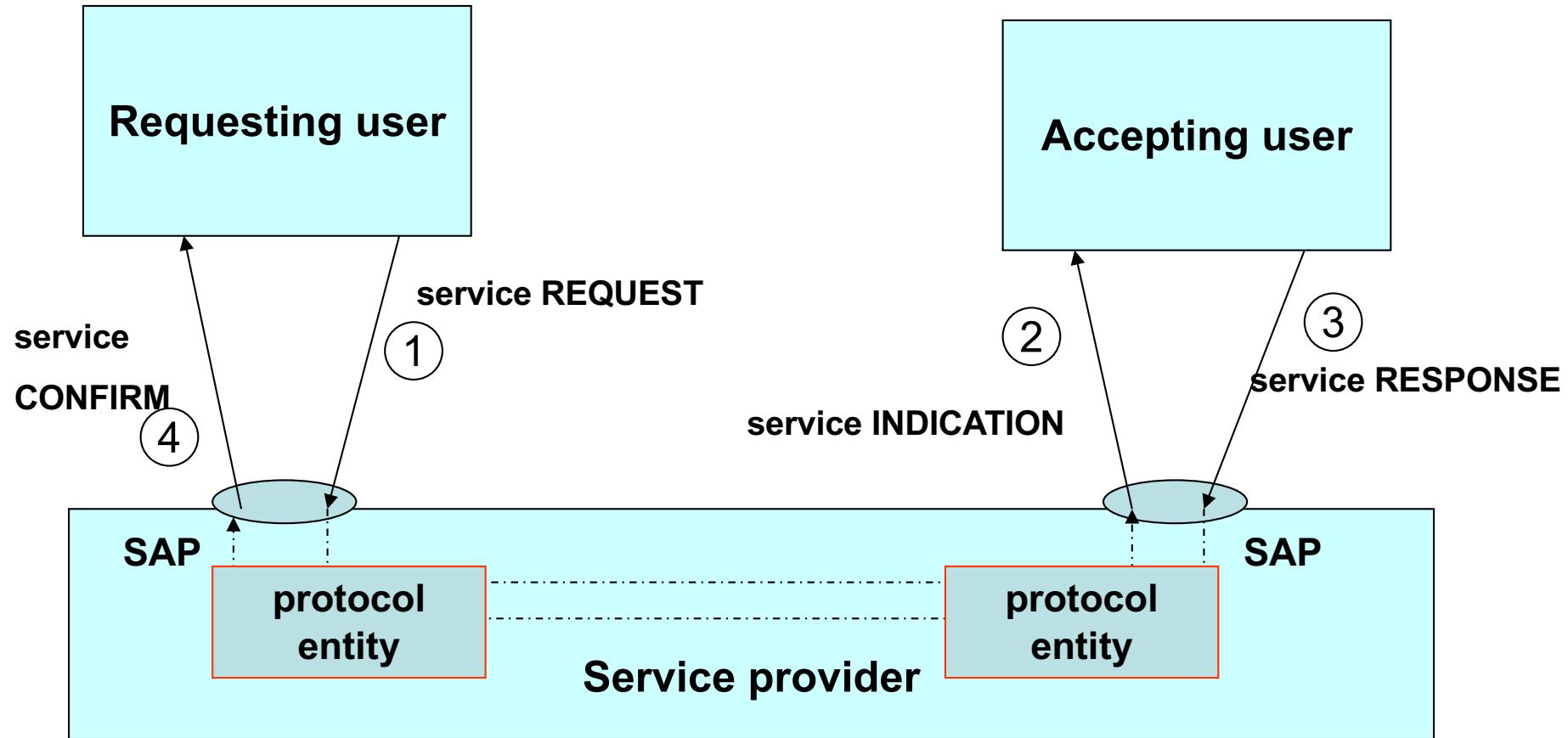
- entitatile de protocol din nivelul k
- colaboreaza folosind protocolul de nivel k
- folosind serviciul nivelului k-1
- pentru a furniza serviciul de nivel k
- entitatilor aflate pe nivelului k+1



# Primitive de serviciu

- Un serviciu este specificat de un set de **primitive** (operații accesibile utilizatorului serviciului)
- Patru clase de primitive
  - REQUEST cere un serviciu
  - INDICATION anunță primirea unei cereri
  - RESPONSE răspunde cererii
  - CONFIRM confirmă cererea

# Servicii confirmate

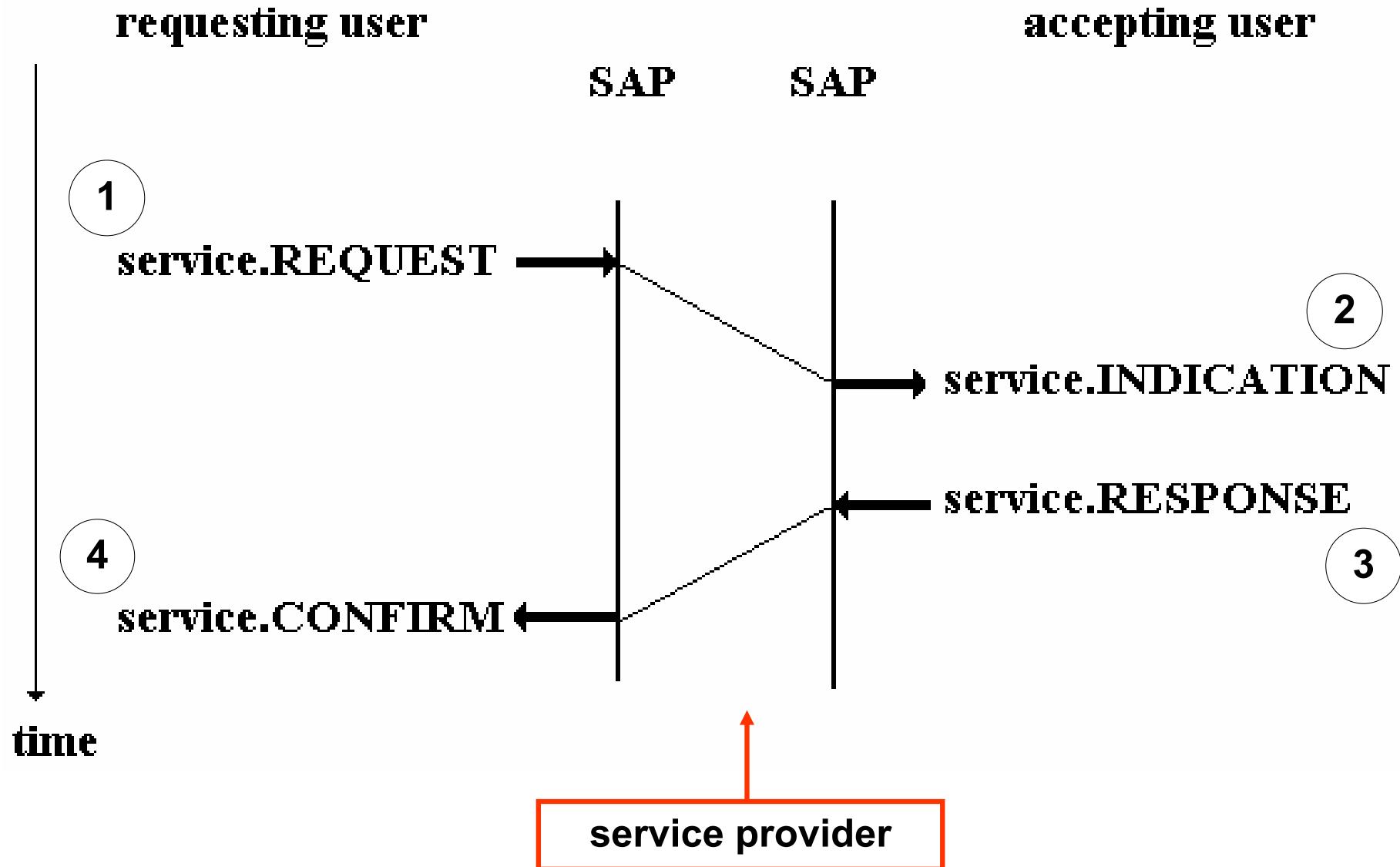


Entitatile de protocol pot inter-schimba mai multe mesaje pentru un singur serviciu confirmat

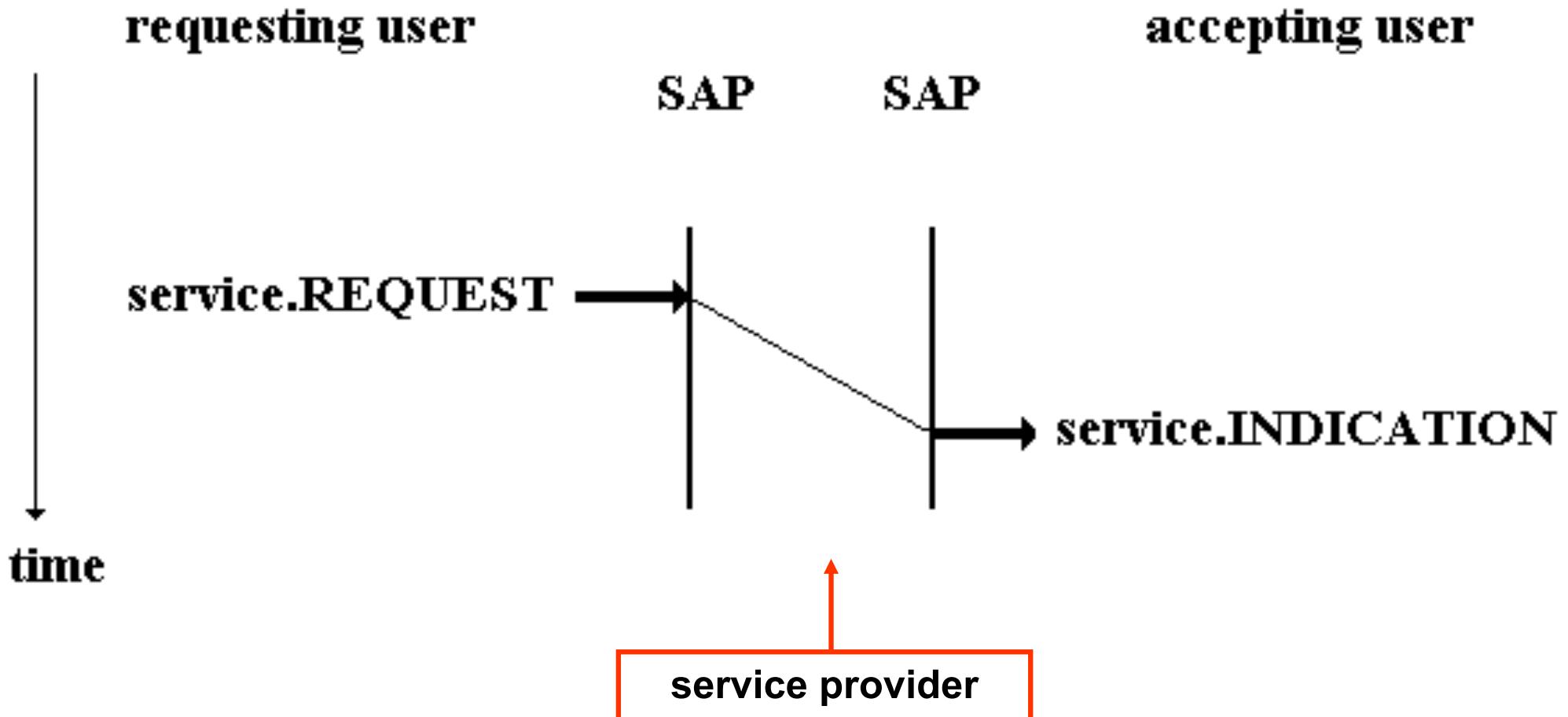
- ex.: mesaje de negociere a serviciului, de repetare la eroare etc.

**SAP = Service Access Point**

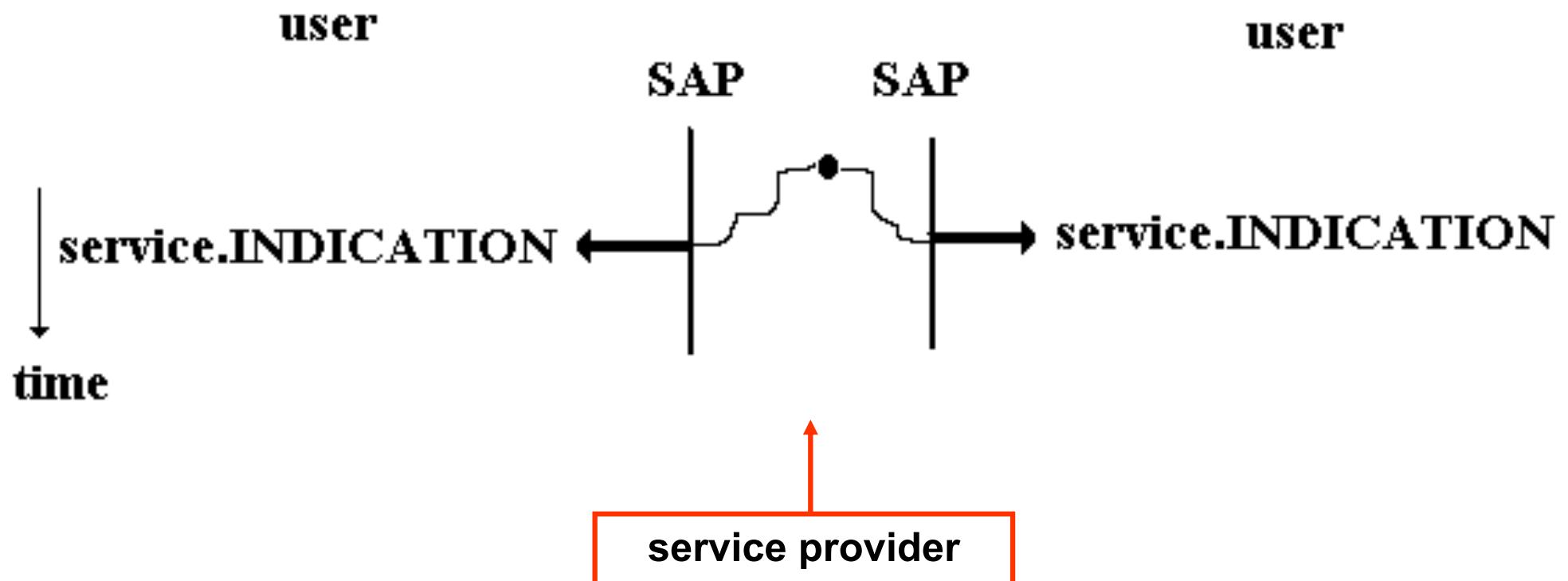
# Servicii confirmate (o alta reprezentare)



# Servicii neconfirmate



# Servicii inițiate de furnizor





# Mod orientat pe conexiune

utilizator solicitant

utilizator solicitat





# Ce conțin specificațiile ?

## Specificație Serviciu

- primitive (operații)
- parametri
- reguli asupra ordinii operațiilor (state machine)

## Specificație Protocol

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



# Protocole OSI

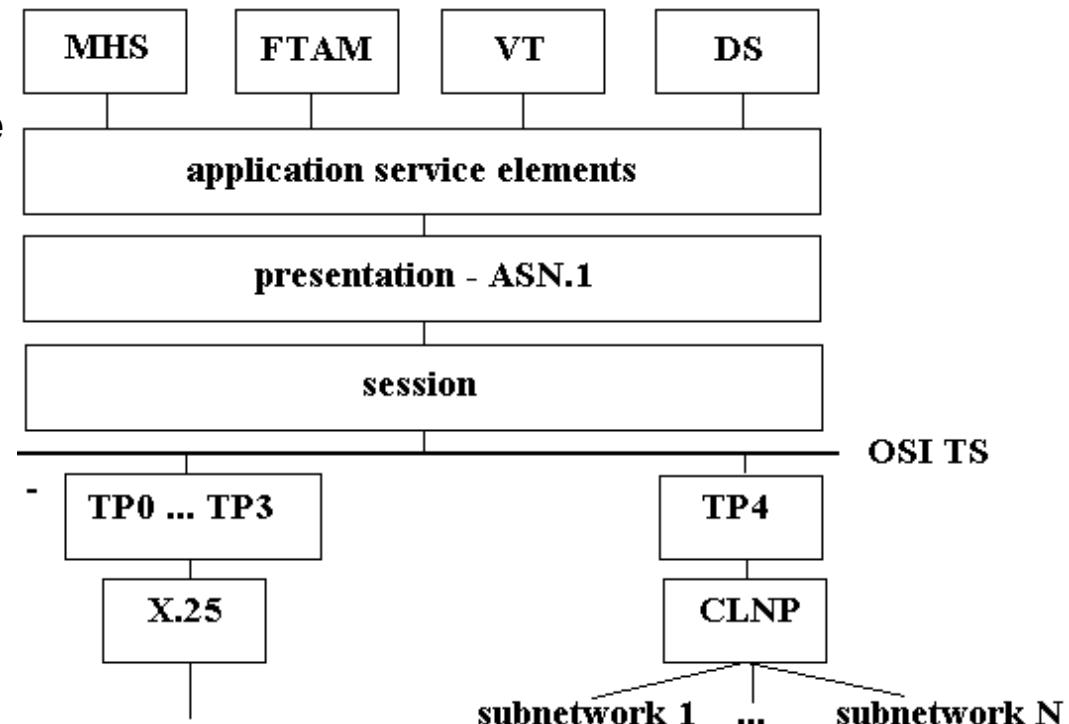
**Physical layer:** V10, V11, V24, V35  
X.21, EIA RS-232-D  
MAC for LANs  
ISDN physical interface

**Data Link Layer:** HDLC LAP B for X.25  
LLC for LAN  
LAP D for ISDN

**Network Layer:** X.25, X.3, X.28, X.29  
CLNP

**Transport Layer:** TP0,..., TP4

**Session Layer:** session protocol



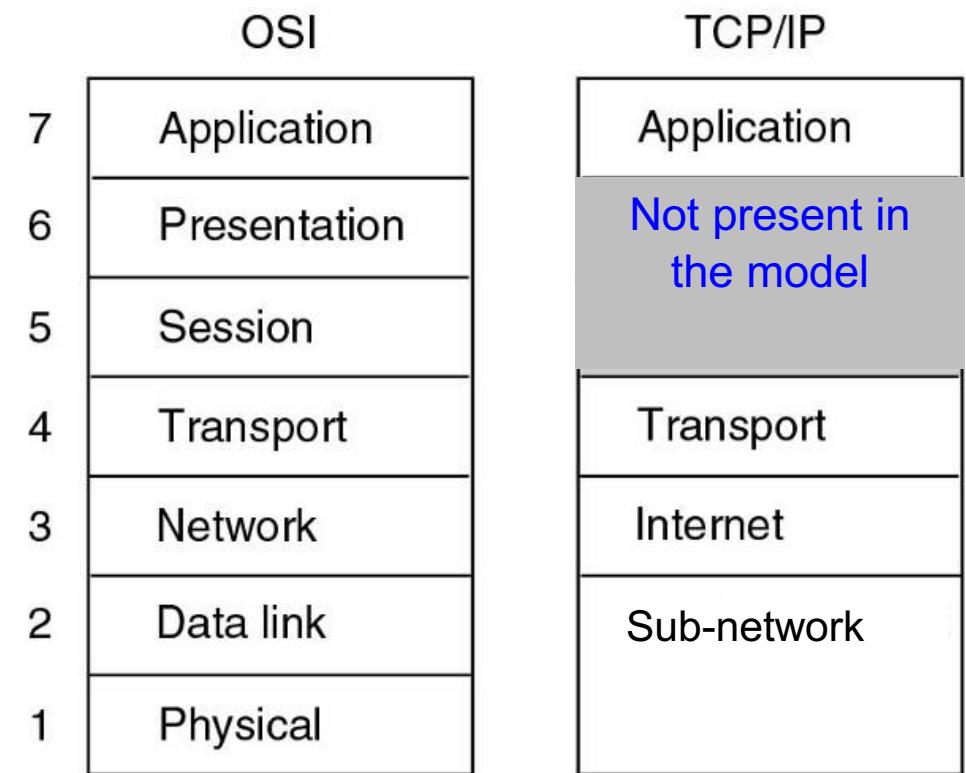
**Presentation Layer:** ASN.1 – Abstract Syntax Notation One

**Application Layer:** MHS - Message Handling System, X.400  
FTAM - File Transfer, Access, and Management  
VT - Virtual Terminal  
DS - Directory Services, X.500



# Modelul de referință TCP/IP

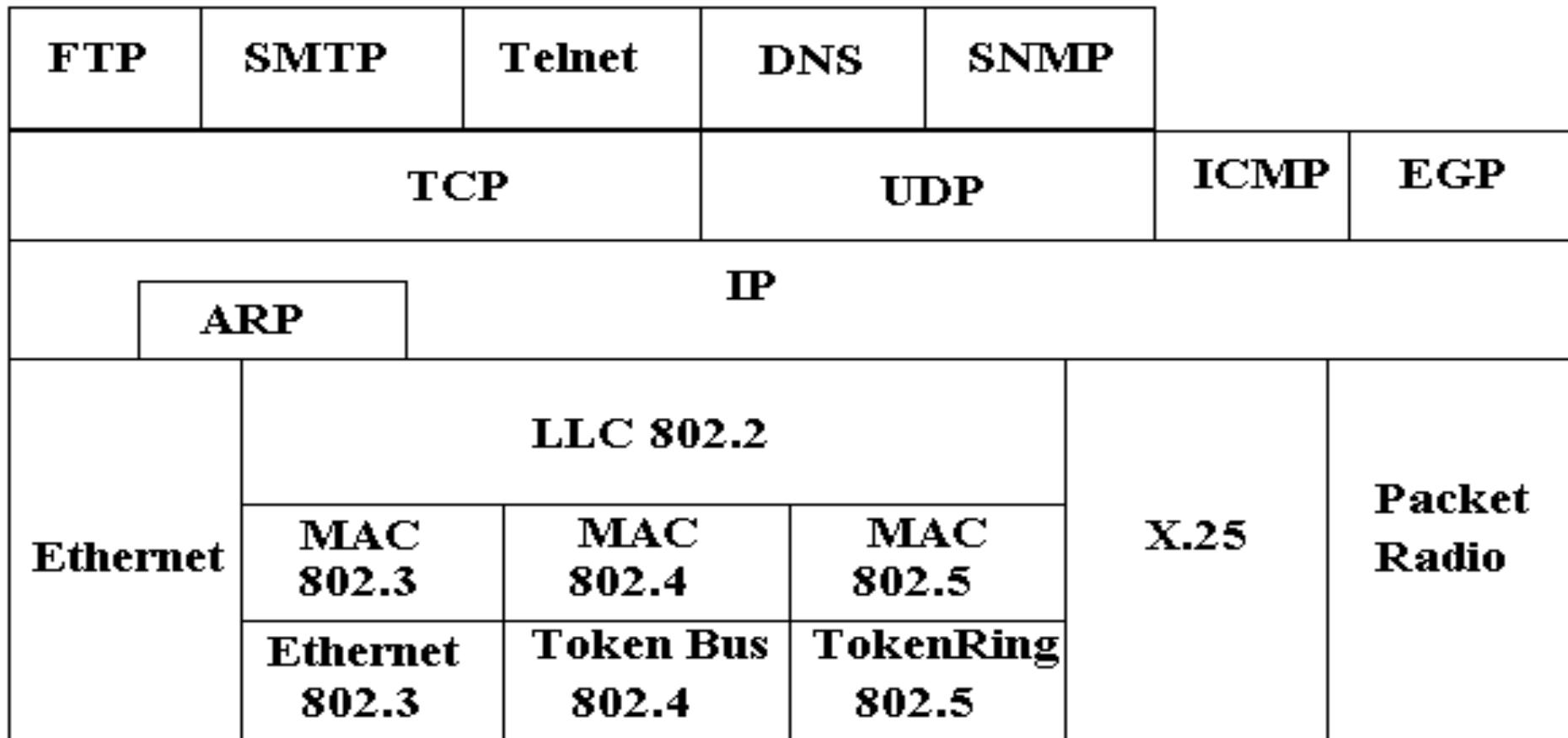
- Nivelul inferior este **sub-retea**
  - mare varietate de protocoale (ex. 802.11 Wi-Fi)
  - pot fi mai multe sub-nivele
  - modelul nu da detalii despre acest nivel
- Nivelul Internet
  - un singur protocol, IP
  - pentru retele interconectate
- Nivelul Transport
  - TCP – canal sigur pentru siruri de octeti
  - UDP – canal nesigur pentru livrarea datagramelor (user datagram – sinonim pentru mesaje)





# Protocole în modelul TCP/IP

- Nivel Aplicatie
  - Varietate de protocole pentru transferul fisierelor si postei, login la distanta, managementul retelei





# Alte protocole în modelul TCP

HTTP	HyperText Transfer Protocol
IIOP	Internet Inter-Orb Protocol
WAP	Wireless Application Protocol
SOAP	Simple Object Access Protocol
LDAP	Lightweight Directory Access Protocol
SSL	Secure Sockets Layer
VPN	Virtual Private Networks
IPSEC	IP Security
PKI	Public Key Infrastructure
HTML	HyperText Markup Language
XML	Extensible Markup Language
WSDL	Web Services Description Language
UDDI	Universal Description, Discovery, and Integration



# Comparație OSI și TCP/IP

## Contra OSI

- Moment nepotrivit
- Tehnologie proastă
- Implementări rele
- Politici proaste

## Contra TCP-IP

- Nu distinge între servicii, interfețe, protocoale
- Nu este un model general
- “Nivelul” gazdă-rețea nu este un nivel
- Nu menționează nivelele fizic și legătură de date
- Protocoale minore bine înrădăcinate - greu de înlocuit



# Test

Un sistem are o ierarhie de protocoale organizate pe 6 niveluri. Aplicațiile generează mesaje având dimensiunea de 45 de octeti. La fiecare nivel este adăugat un antet de 20 de octeti.

Ce fractiune din latimea benzii este ocupată de antete?

- (a) ~ 64 %
- (b) ~ 69 %
- (c) ~ 73 %

Arătați calculul facut.

Puteți evalua eficiența acestei aplicații?



# Studiu individual

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

- 1.4 MODELE DE REFERINȚĂ

- 1.4.1 Modelul de referință OSI
- 1.4.2 Modelul de referință TCP/IP
- 1.4.3 O comparație între modelele de referință OSI și TCP
- 1.4.4 O critică a modelului și protocolelor OSI
- 1.4.5 O critică a modelului de referință TCP/IP

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

- 1.4 REFERENCE MODELS

- 1.4.1 The OSI Reference Model
- 1.4.2 The TCP/IP Reference Model
- 1.4.3 A Comparison of the OSI and TCP/IP Reference Models
- 1.4.4 A Critique of the OSI Model and Protocols
- 1.4.5 A Critique of the TCP/IP Reference Model

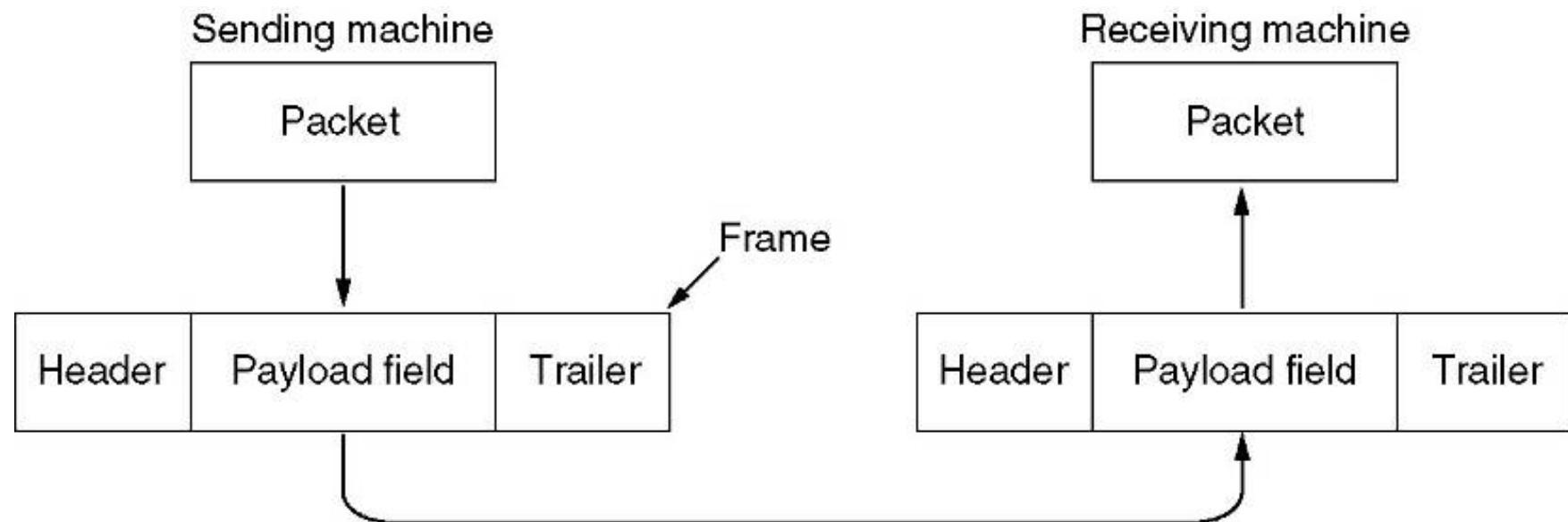


# Nivelul legăturii de date

# Funcțiile nivelului legăturii de date

## 1 – Încadrarea datelor

## 2 – Transmisia transparentă





### 3 – Controlul erorilor

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

### 4 – Controlul fluxului – protocol

- mesaje de permisiune pentru transmițător

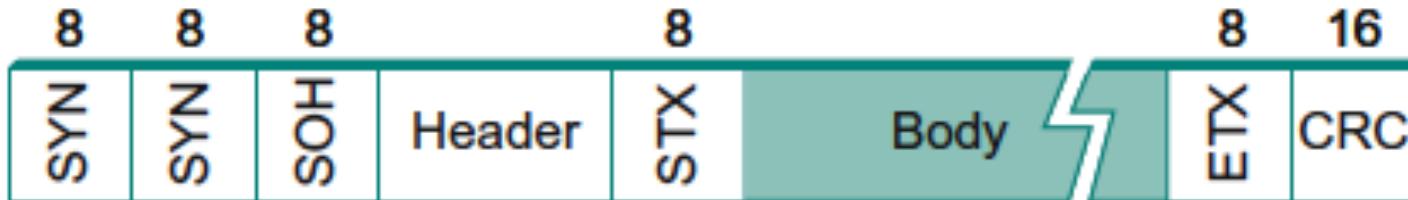
### 5 – Gestiunea legăturii – protocol

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



# Metode de încadrare

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



**SOH** - start of heading

**ETX** - end of text

**EOT** - end of transmission

**ACK** - acknowledge

**SYN** - synchronous idle

**CRC** - cyclic redundancy check

**STX** - start of text

**ETB** - end of transmission block

**ENQ** - enquiry

**NAK** - not acknowledge

**DLE** - data link escape



## Metode de încadrare

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

**SYN SYN SOH count flag resp seq address CRC data CRC**

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

**flag address command data FCS flag**



## 2 – Transmisie transparentă

STX **text** ETX

**text alfanumeric:** OK!

STX **text... ETX** ...**text** ETX

**text binar:** ETX fals ?

Solutie: **umplere cu caractere**

Dubleaza caracterele de control cu DLE

Defineste combinatii admise

DLE STX – start text transparent

DLE ETX – sfarsit text transparent

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

Dubleaza DLE la transmitere si elimina la receptie

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

Eroare: receptie DLE **x**

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



## Umplere cu biți

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

011011111101111111111010

01111110 011011111101111111111010 01111110

Solutia: adaugarea unui zero după 5 unități (în interiorul cadrului!)

01111110 011011111010111110111110010 01111110

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

Simplifică regula receptorului: **elimina zeroul aflat după 5 unități**

011011111010111110111110010

01101111101111111111010



# Detectia și corectarea erorilor

## Noțiuni preliminare

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

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

• ponderea Hamming a lui  $w$  = numărul de unități conținute de  $w$

• distanța Hamming,  $d(u,v)$  dintre  $u$  și  $v$  este

ponderea vectorului suma modulo 2  $u+v$



# Detectia și corectarea erorilor

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

Condiția – erorile să producă cuvinte din restul mulțimii  $W_n$

Multimea cuvintelor  $W_n$  de lungime n se imparte în 2 categorii:

$S_n$  este multimea cuvintelor cu sens

$F_n$  este multimea cuvintelor fără sens

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

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

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

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



# Exemplu

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

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

Astfel,

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

Nu se pot corecta 3 erori:

0000000111 poate proveni din 0000000**000** în cazul a 3 erori



# Metoda Hamming

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

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

Control paritate (pară sau impară)

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

Bit 1 controlat de biții 1

Bit 2 controlat de biții 2

Bit 3 controlat de biții 1, 2

Bit 4 controlat de biții 4

Bit 5 controlat de biții 1, 4

Bit 6 controlat de biții 2, 4

Bit 7 controlat de biții 1, 2, 4

Bit 8 controlat de biții 8

Bit 9 controlat de biții 1, 8

Bit 10 controlat de biții 2, 8

Bit 11 controlat de biții 1, 2, 8



# Metoda Hamming (2)

- Invers:
- Bit 1 controlează biți 1, 3, 5, 7, 9, 11
  - Bit 2 controlează biți 2, 3, 6, 7, 10, 11
  - Bit 4 controlează biți 4, 5, 6, 7
  - Bit 8 controlează biți 8, 9, 10, 11

Aceasta forma foloseste la calculul bitilor de control

Exemplu (paritate pară) pentru **1100001**

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



Prima forma foloseste la **corectarea** erorilor

Ex.1 În loc de:

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

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

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

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

**Bit 1** controlat de biții 1

Bit 3 controlat de biții 1, 2

Bit 5 controlat de biții 1, 4

Bit 7 controlat de biții 1, 2, 4

Bit 9 controlat de biții 1, 8

Bit 11 controlat de biții 1, 2, 8

**Bit 2** controlat de biții 2

**Bit 4** controlat de biții 4

Bit 6 controlat de biții 2, 4

**Bit 8** controlat de biții 8

**Bit 10 controlat de biții 2, 8**



Ex. 2 În loc de:

1100001	=>	<span style="color: blue;">1 2 3 4 5 6 7 8 9 10 11</span>
		<span style="color: black;">1 <u>0</u> 1 1 1 0 0 1 0 0 1</span>

Se primește eronat      1 1 1 1 1 0 0 1 0 0 1

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

1 1 1 1 1 0 0 1 0 0 1

celelalte sume sunt corecte → bit 2 este inversat

Obs.

**Codul Hamming corectează erorile de 1 bit**

# Corecția erorilor în rafală

Utilizarea unui cod Hamming pentru corecția erorilor în rafală

- matricea de biți este transmisă coloană cu coloană
- poate corecta erori în rafală dintr-o coloană dacă există un bit eronat pe fiecare linie

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

Order of bit transmission



# Coduri detectoare de erori

## Coduri polinomiale

$k$  biți de informație (date)       $i(X)$  polinomul corespunzător

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

$n-k$  biți de control       $r(X)$

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

$r(X)$  se alege astfel ca       $w(X) = X^{(n-k)}i(X) + r(X)$

să fie multiplu de  $g(X)$        $w(X) = g(X).q(X)$

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

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

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

# Coduri detectoare de erori

Frame : 1101011011

Generator: 10011

Message after 4 zero bits are appended: 11010110110000

Calculul sumei de control  
pentru un cod polinomial

10 biti informatie + 4 biti control

Imparte **11010110110000**

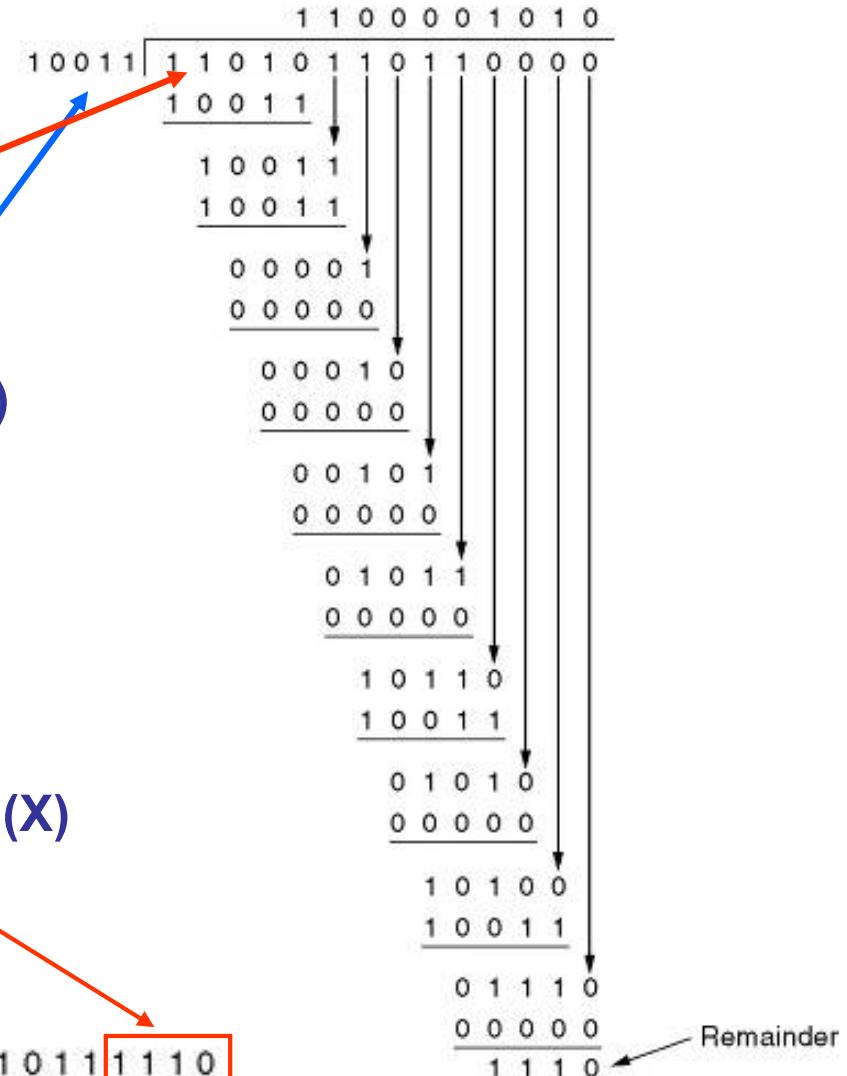
la **10011**

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

$g(X)$

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

Transmitted frame: 1101011011 **1110**



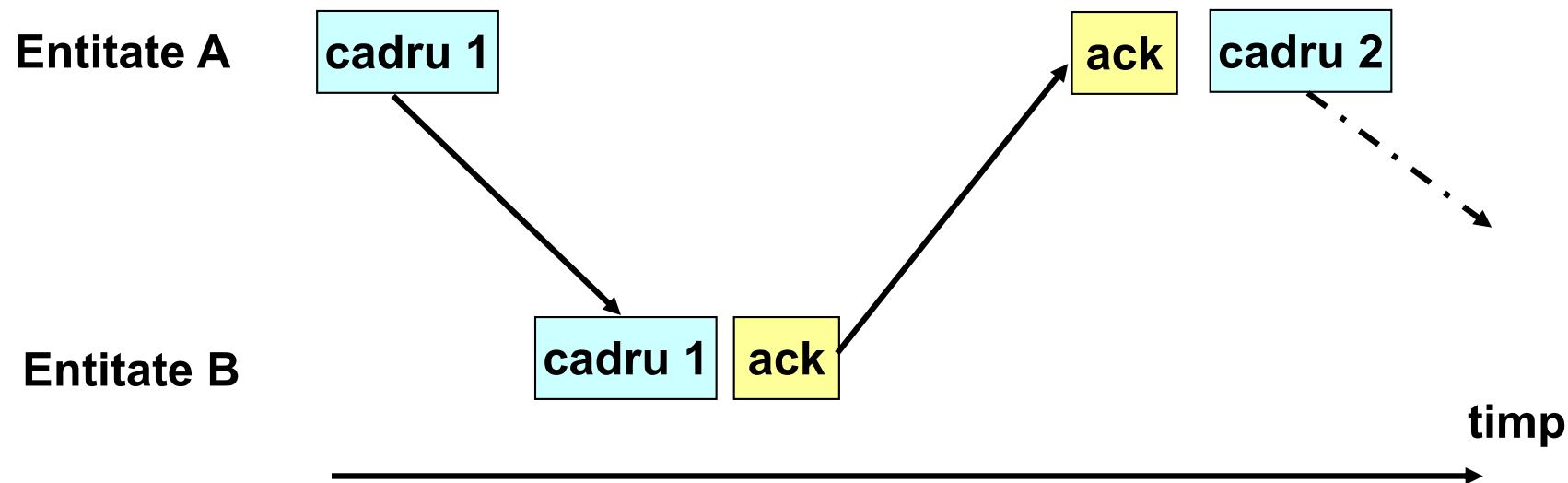


# Ce erori pot fi detectate?

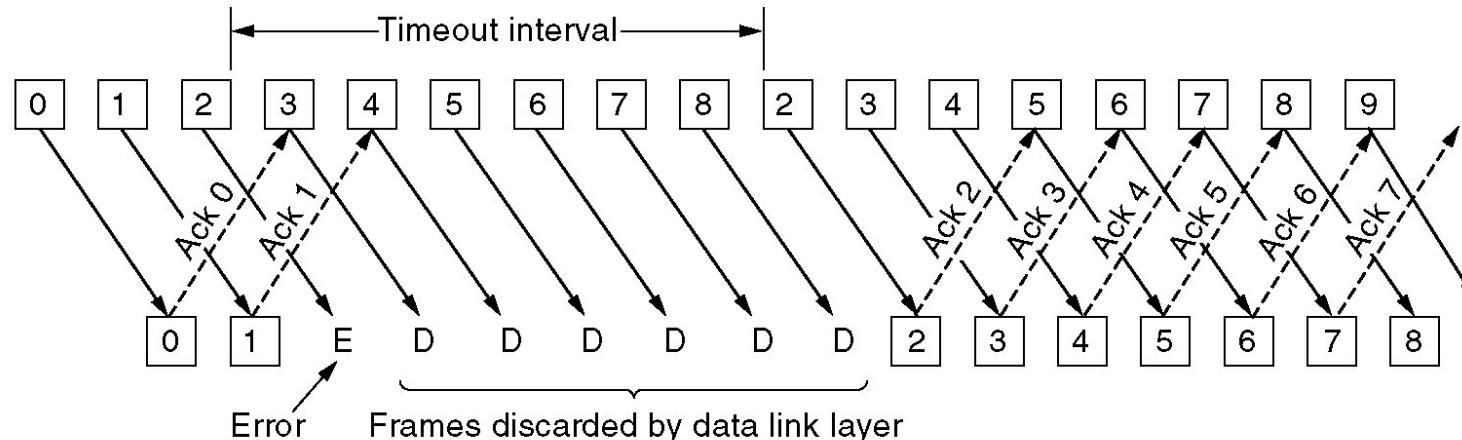
- Probabilitatea de detectie depinde de lungimea codului de control
- CRC si sume de control pe
  - 8 biti detecteaza 99.6094% din erori
    - $(x^8+x^2+x+1)$  CRC-8-CCITT)
  - 16 biti detecteaza 99.9985% din erori
    - $(x^{16}+x^{10}+x^8+x^7+x^3+1)$  CRC-16)
  - 32 biti detecteaza 99.9999% din erori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

# Protocole elementare pentru legătura de date

## Start-stop

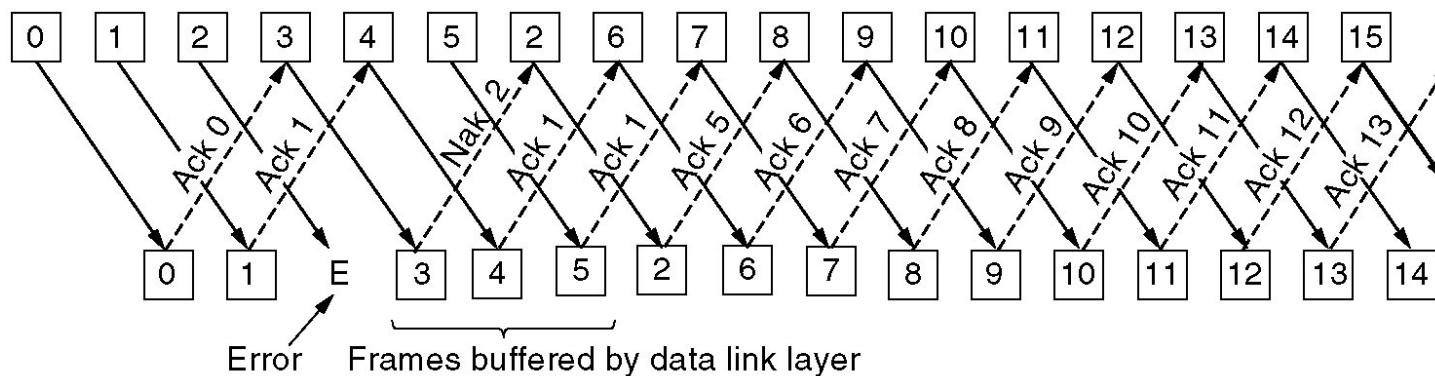


# Ferestre glisante



Time →

(a)



(b)



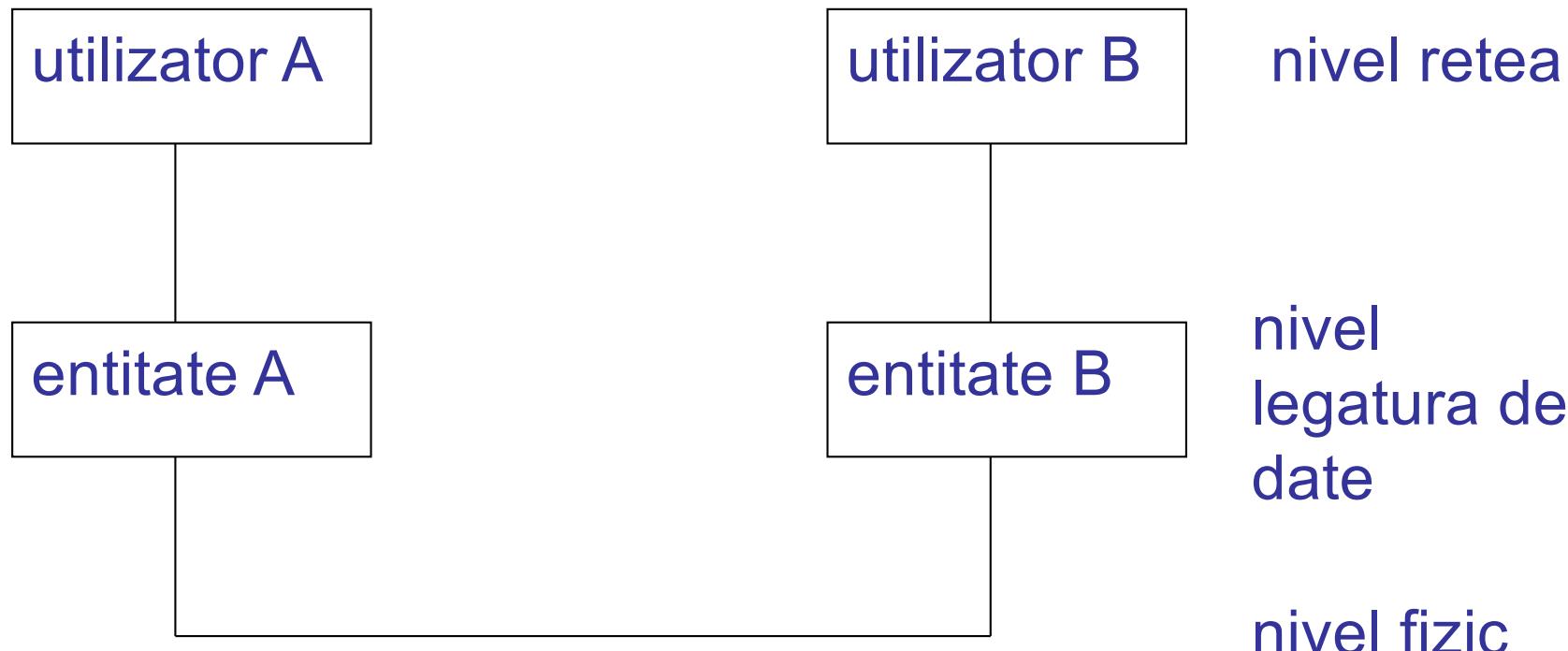
# Specificație Protocol

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



# Protocolele legăturii de date

Configurația entităților de protocol





# Datele

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

enum FelCadru {data, ack, nak};

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

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



# Primitivele de serviciu

La interfața cu nivelul superior (rețea)

- preluarea unui pachet de la rețea pentru transmitere pe canal  
**pachet DeLaRețea();**
- livrarea către rețea a unui pachet  
**void LaRețea (pachet);**

La interfața cu nivelul inferior

- trecerea unui cadru nivelului fizic pentru transmisie  
**void LaFizic (cadru);**
- preluarea unui cadru de la nivelul fizic  
**cadru DeLaFizic();**



## Tratarea evenimentelor

```
enum TipEven { SosireCadru,  
              EroareControl,  
              TimeOut,  
              ReteaPregatita};  
  
TipEven wait();
```



# Protocole start-stop

## Protocol simplex fără restrictii

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

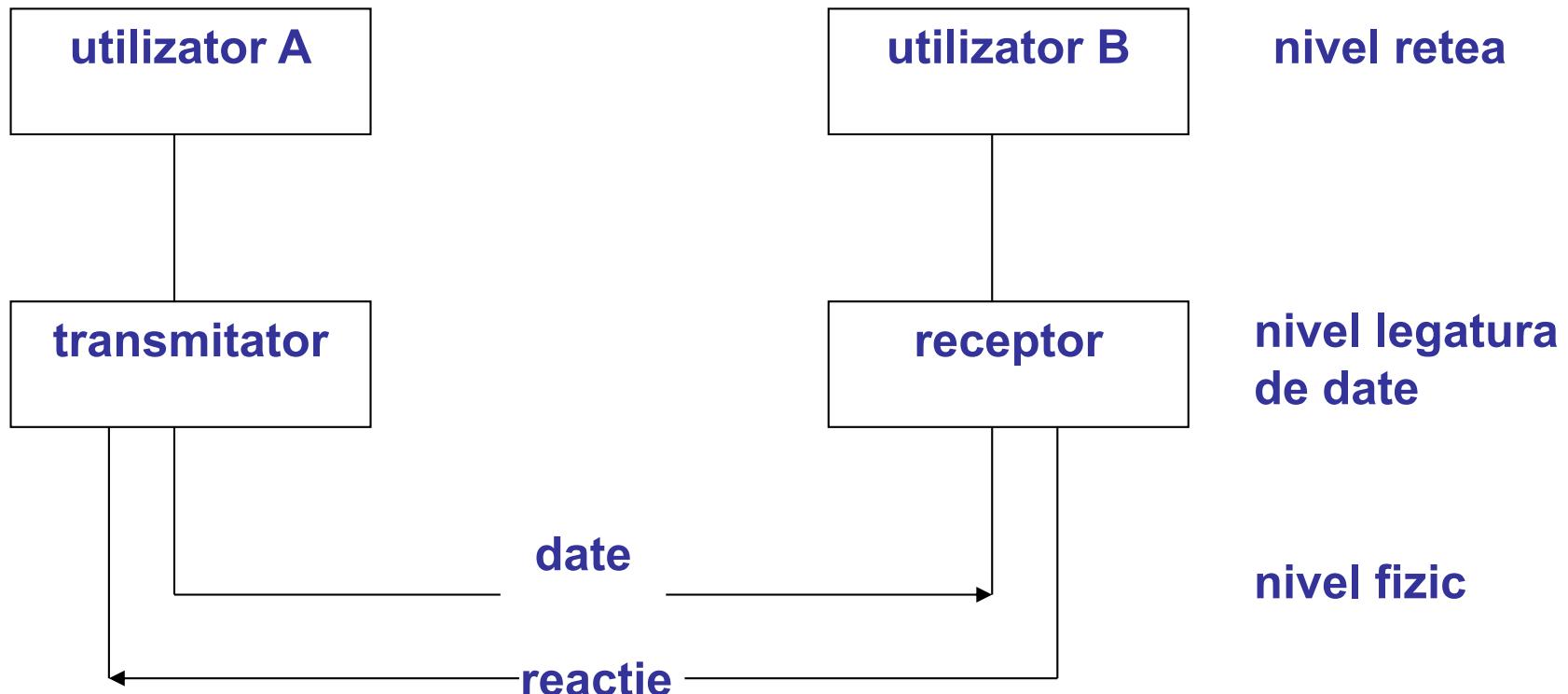


```
# define forever while(1)
// entitatea din sistemul transmitatorului
void transmit1() {
    cadru s;
    do{
        s.info = DeLaRetea(); //preia pachet
        LaFizic(s);          //transmite cadru
    } forever;
}

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

# Protocol simplex start-stop

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



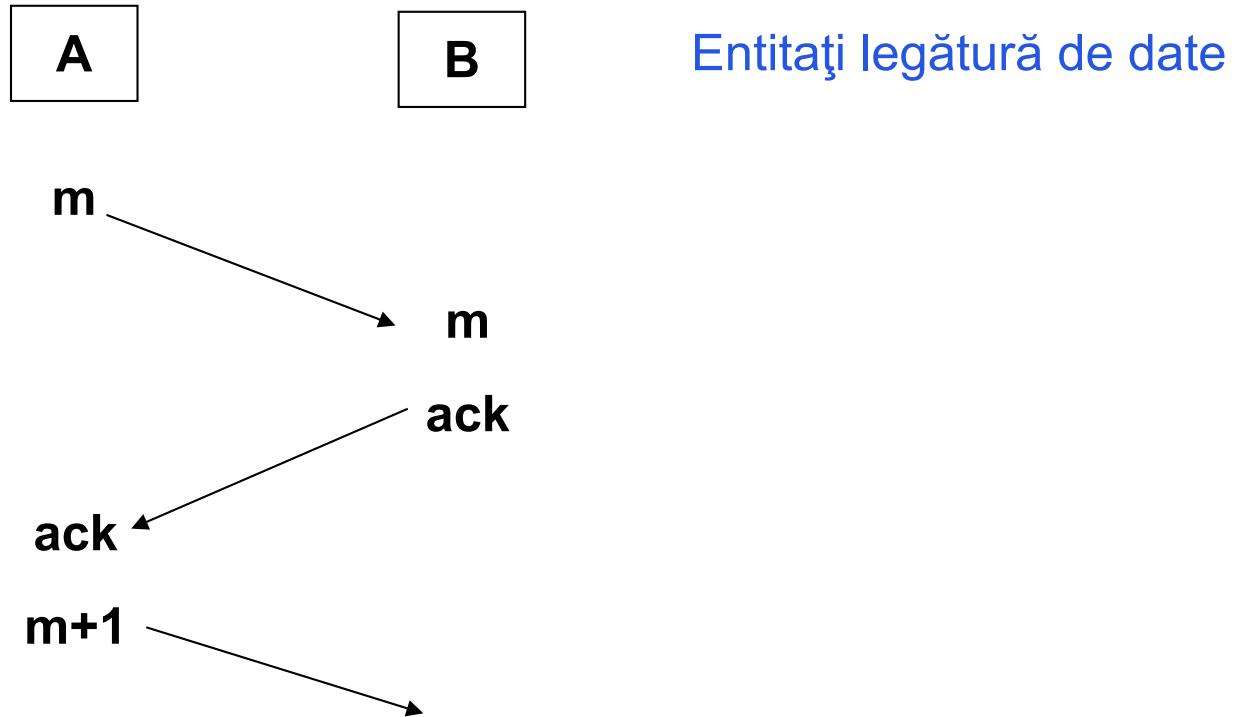


```
void transmit2(){
    cadru s;
    TipEven even;
    do{
        s.info=DeLaRetea();
        LaFizic(s);
        even=wait();           //asteapta permisiunea
    } forever;
}

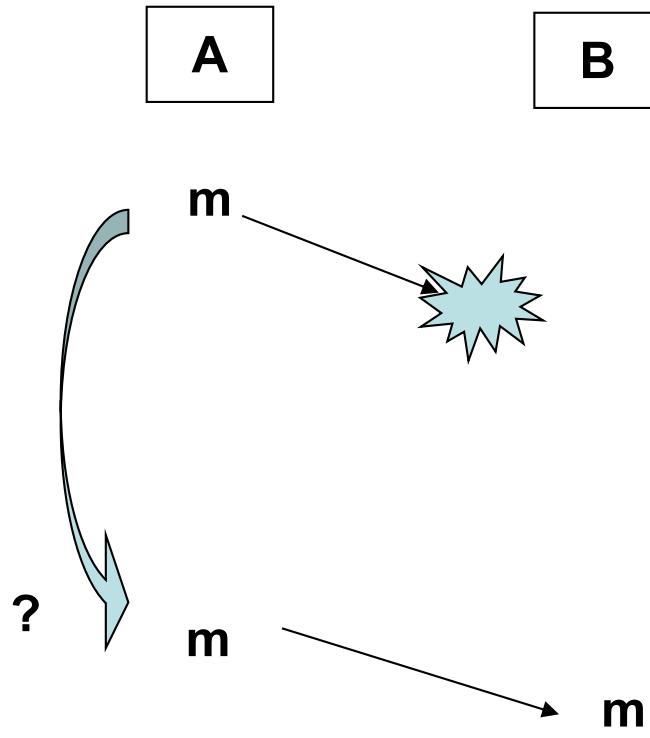
void recept2(){
    cadru s,r;
    TipEven even;
    do{
        even=wait();          //poate fi doar SosireCadru
        r=DeLaFizic();
        LaRetea(r.info);
        LaFizic(s);           //transmite permisiunea
    } forever;
}
```



# Protocol simplex pentru un canal cu erori



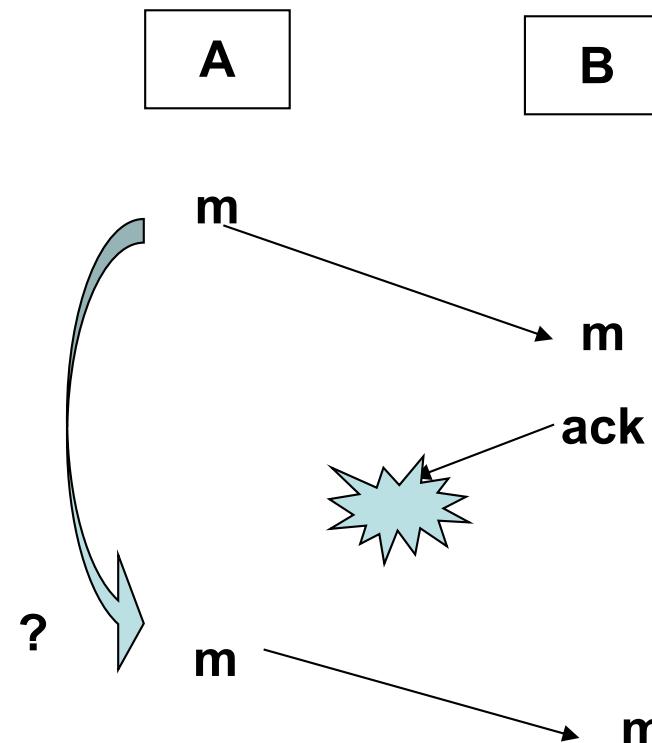
Transmisie corecta



**Pierdere m**

**La time-out A retransmite m**

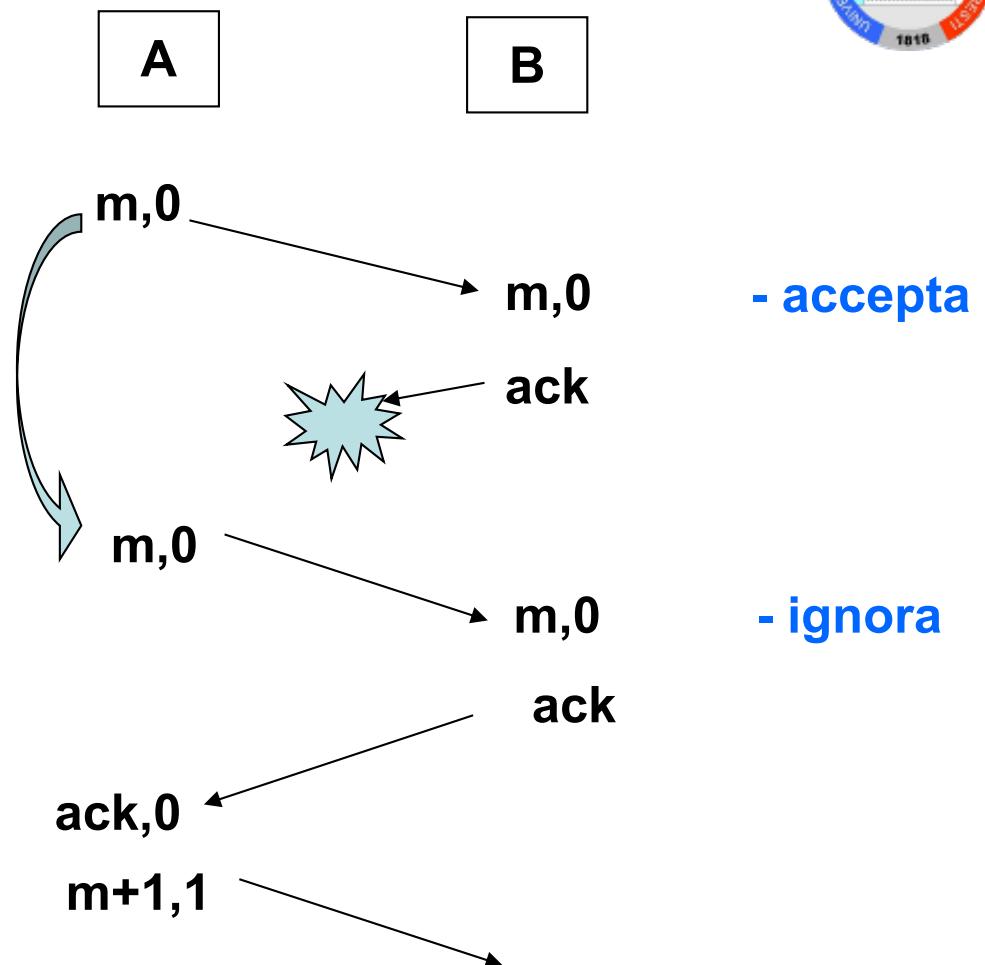
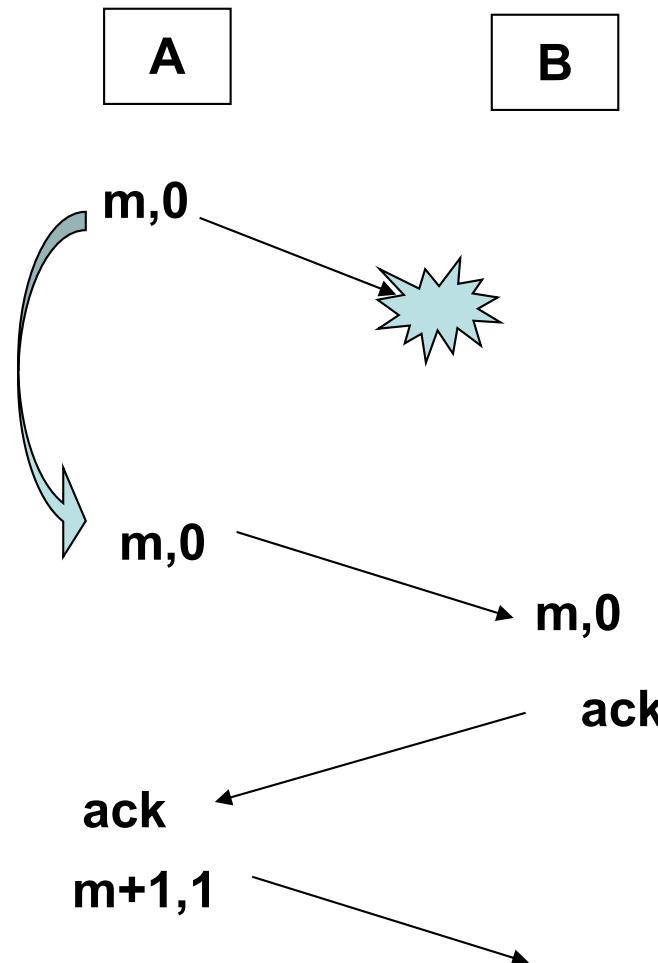
**Care este acceptat corect de B**



**Pierdere ack**

**La time-out se retrimit m**

**Care este acceptat incorrect, ca mesaj nou de B !**



**se adauga un numar de sevență**  
**la time-out se re-transmite ultimul**  
**cadru**  
**B acceptă dacă este corect**

**B ignoră dacă este dublura**



## Protocol simplex pentru un canal cu erori (2)

Este nevoie de un **ceas**

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

de eveniment **TimeOut**

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

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

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

```
#define MaxSecv 1
```

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



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

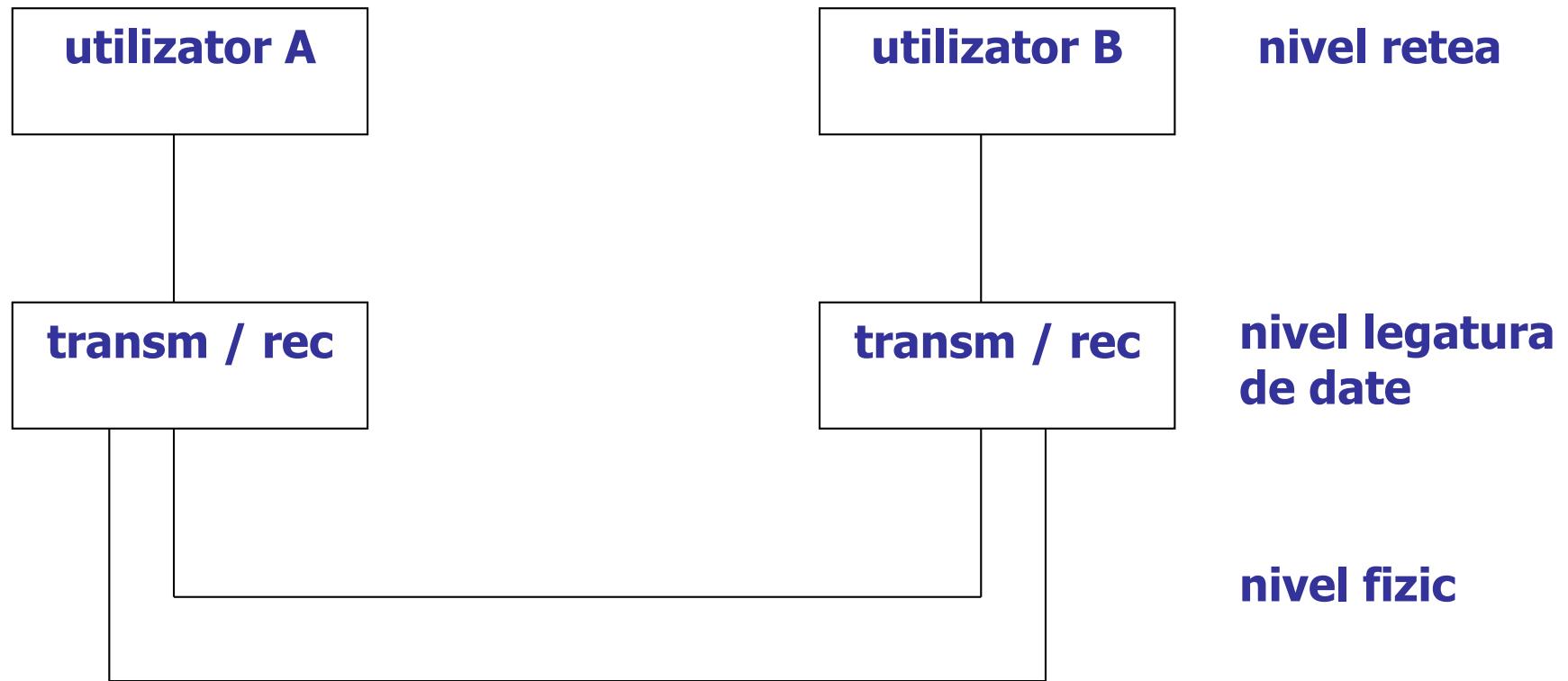


```
void recept3() {
    NrSecv CadruAsteptat=0;
    cadru r,s;
    TipEven even;
    do{
        even=wait();      //SosireCadru sau EroareControl
        if(even==SosireCadru) {
            r=DeLaFizic();
            if(r.secvt==CadruAsteptat) {
                LaRetea(r.info);      //cadru în secventa
                inc(CadruAsteptat);
            }
            LaFizic(s);      //transmite oricum confirmarea
        }
    }forever;
}
```



# Protocole cu fereastră glisantă

## Configurația



**2 legaturi pentru date+confirmare**



## Protocol cu numar de secventa de un bit

**Fiecare stație**

are o secvență de **initializare** în care trimite un prim cadru

**și realizează ciclic următoarele operații:**

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

Fiecare cadru are campurile: info, secv, conf



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



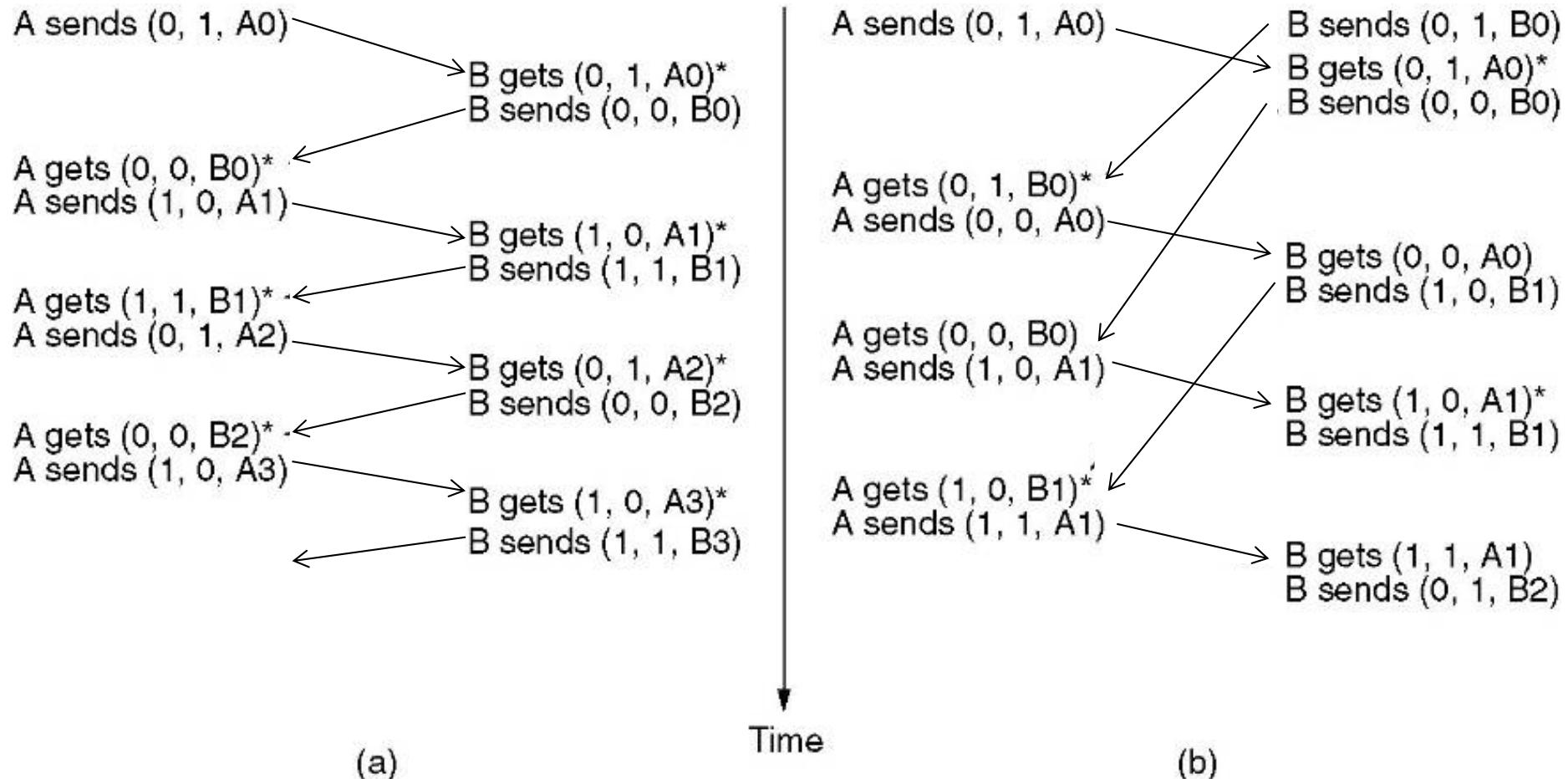
```
do {
    even=wait();
    if (even==SosireCadru) {
        r=DeLaFizic();
        //când este cadrul așteptat, livrează-l entității retea
        if (r.secv==CadruAșteptat) {
            LaRetea(r.info);
            inc(CadruAșteptat);
        }
        //când cadrul transmis este confirmat, pregătește
        // urmatorul cadrus
        if (r.conf==CadruUrmator) {
            StopCeas(r.conf);
            s.info=DeLaRetea();
            inc(CadruUrmator);
        }
    }
}
```



//construitește și transmite un nou cadru

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

# Funcționare protocol cu număr de secvență de un bit

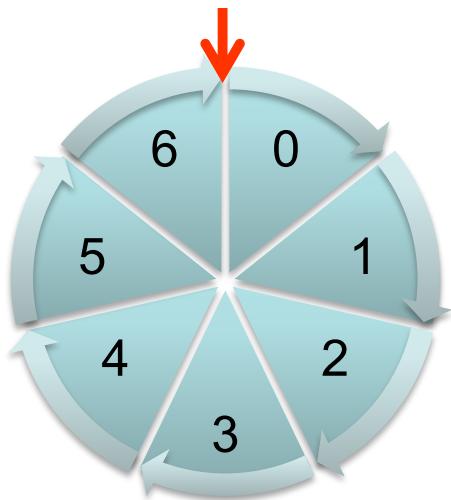


Două scenarii pentru protocolul 4. (a) Cazul normal. (b) Caz anormal.

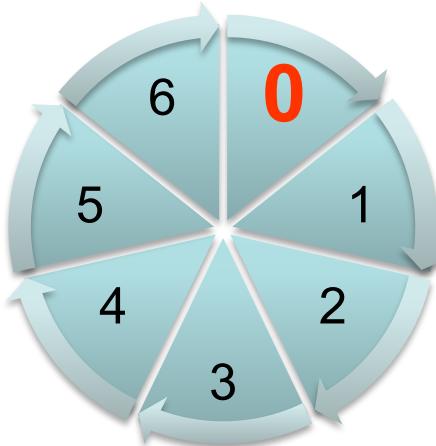
Notăția este (seq, ack, packet number).

Un asterisc arată că nivelul rețea acceptă pachetul.

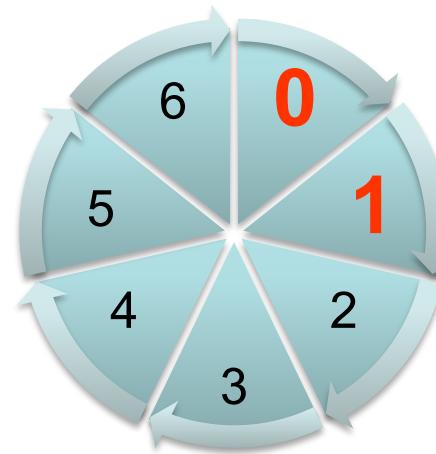
# Fereastra transmitatorului



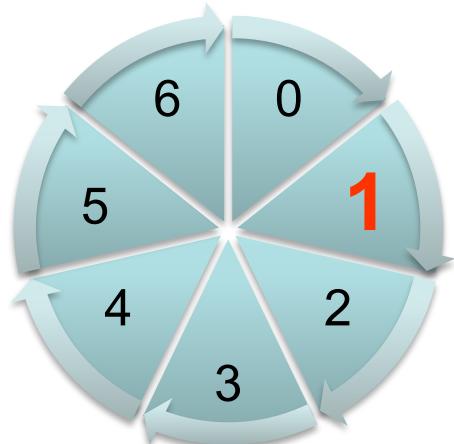
Initial  
Fereastra  $\Phi$



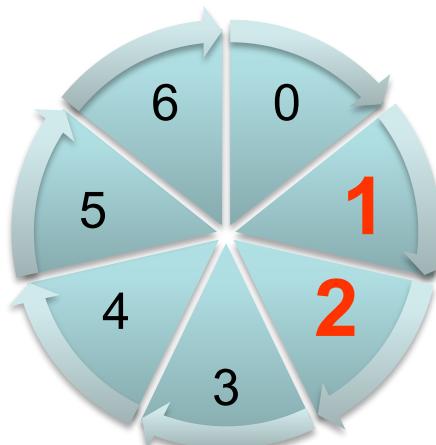
Trimis cadru 0  
Fereastra 0



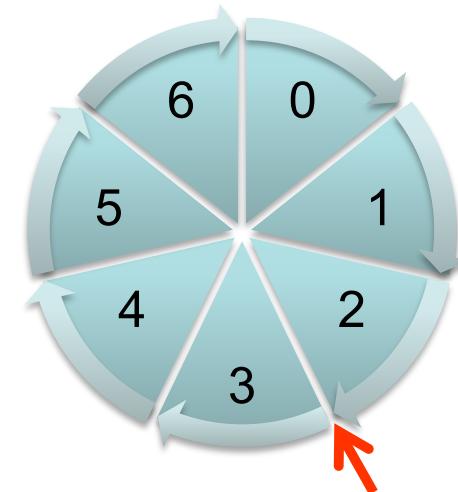
Trimis cadru 1  
Fereastra 0,1



Primit ack 0  
Fereastra 1

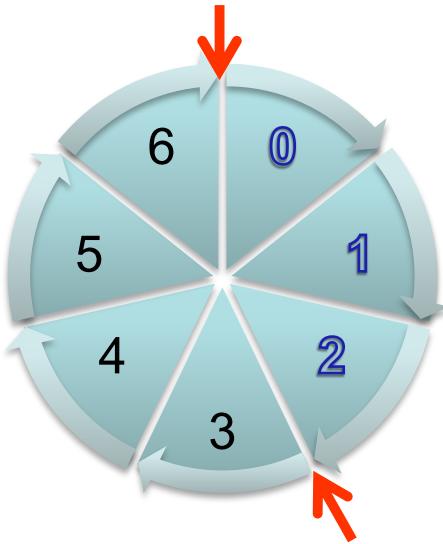


Trimis cadru 2  
Fereastra 1,2

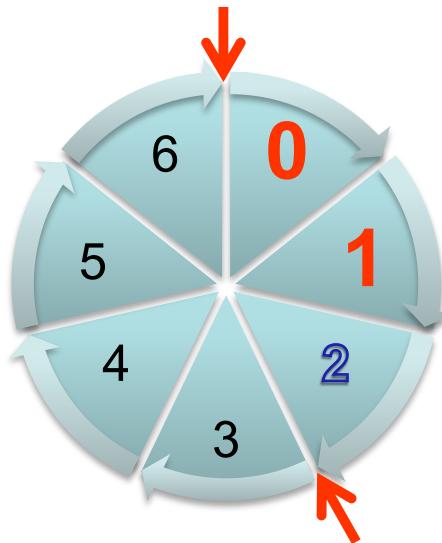


Primit ack 2  
Fereastra  $\Phi$

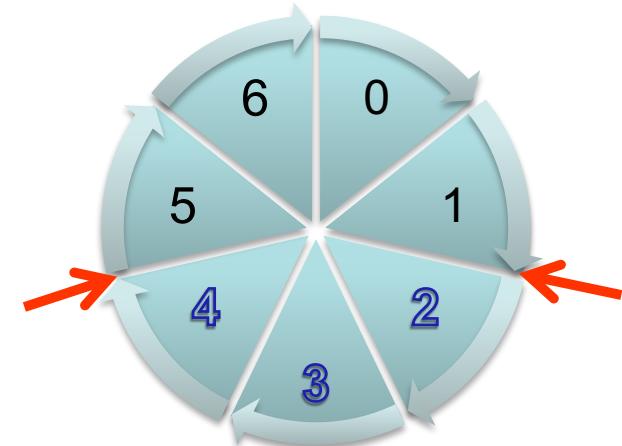
# Fereastra receptorului



Loc ptr 3 cadre  
Fereastra 0,1,2



Primit cadru 1 apoi 0



Livrat cadre 0 si 1  
Fereastra 2,3,4



# Fereastra glisantă

- fereastra este un sub-șir de numere de secvență  
0, 1, 2, **3, 4, 5, 6**, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...
- pe parcursul transmiterii cadrelor, fereastra **glisează**  
0, 1, 2, 3, 4, 5, **6, 7, 0, 1**, 2, 3, 4, 5, 6, 7, 0, 1 ...
- la **transmitator**, fereastra contine numerele cadrelor transmise și ne-confirmate
- dimensiunea ferestrei transmitatorului este **variabilă**
  - creste cand se trimite un nou cadru; ex. 7  
0, 1, 2, **3, 4, 5, 6, 7**, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...
  - scade cand se primește o confirmare; ex. pentru 3 și 4  
0, 1, 2, 3, 4, **5, 6, 7**, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 ...

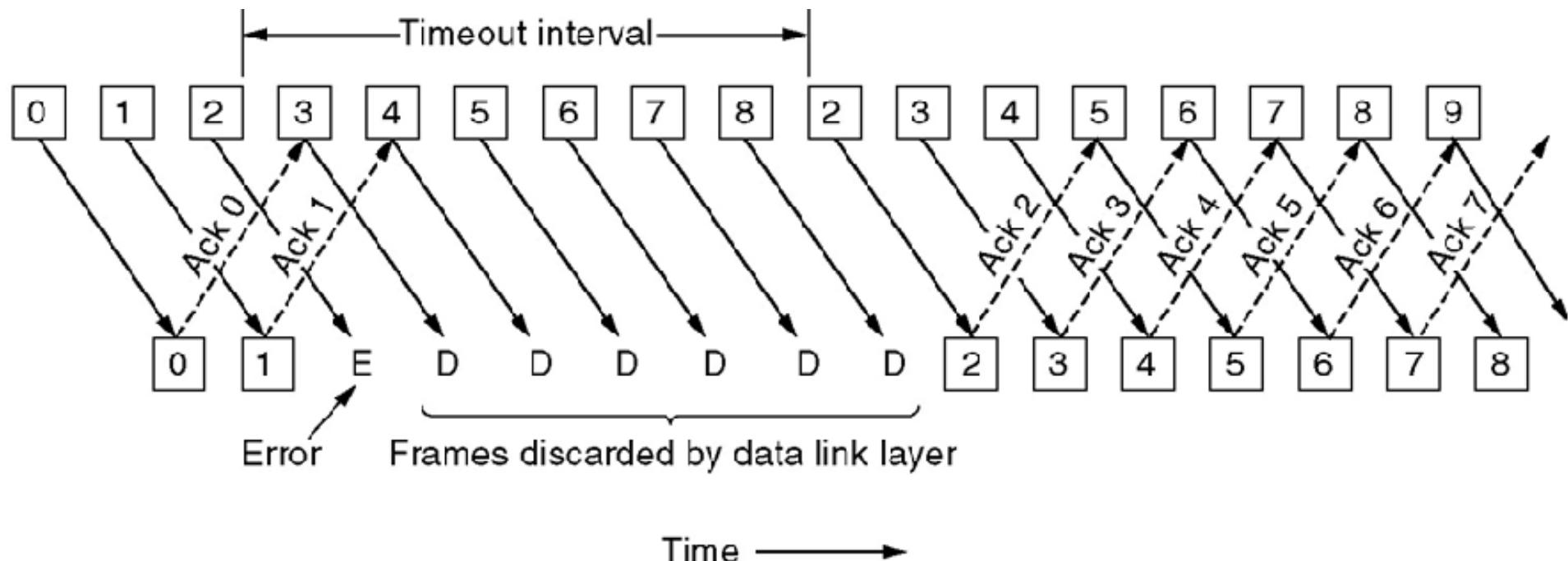


# Fereastra receptorului

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

# Un protocol cu retransmitere neselectivă “Go Back N”

Efectul erorii cand fereastra receptorului este 1.





## Protocole cu fereastră supraunitară de transmisie

### Protocol cu retransmitere neselectivă

Sunt **MaxSecv** + 1 numere de secvență diferite

Fereastra maxima a transmitatorului poate fi de **MaxSecv** cadre

Demonstratie pe scenariu cu **MaxSecv** = 7 și fereastra de 8

1. Transmitatorul trimite cadrele 0..7;
2. Toate cadrele sunt receptionate și confirmate;
3. Toate confirmările sunt pierdute;
4. Transmitatorul retrimită la **time-out** toate cadrele;
5. Receptorul **acceptă** duplicatele.



```
#define MaxSecv 7

void ActivRetea();
void DezactivRetea();
NrSecv CadruUrmator, //urmatorul cadru de transmis
        ConfAsteptata, //cel mai vechi cadru neconfirmat
                    // impreuna desemneaza fereastra transmisie
        CadruAsteptat; //urmatorul cadru asteptat

cadru r,s;
pachet tampon[MaxSecv+1];
NrSecv ntampon,i;
TipEven even;
```



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

ex.1

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

a        b        c

ex.2

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

a        b        c

ex.3

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

a        b        c



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



```
void protocol5() {
    ActivRetea();           // permite even. "ReteaPregatita"
    CadruAsteptat=0;        // fereastra de receptie 1 cadru
    CadruUrmator=0;         // margine sup fereastra transmisie
    ConfAsteptata=0;         // margine inf fereastra transmisie
    ntampon=0;               // initial nici un pachet in tampon
    do{
        even=wait();          // functionare total dirijata
                               // de evenimente
        switch(even) {
            case ReteaPregatita: // reteaua are de transmis
                //memoreaza pachet in tampon (ptr retransmisie)
                //si trimite-l pe legatura de date
                tampon [CadruUrmator]=DeLaRetea();
                ntampon++;
                transmite(CadruUrmator);
                inc(CadruUrmator);
                break;
        }
    }
}
```

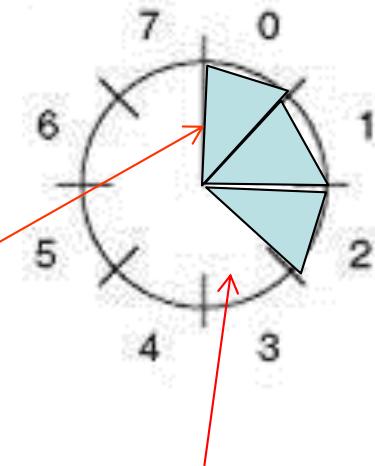


```

case SosireCadru:
    r=DeLaFizic();
    if(r.secv==CadruAsteptat) {
        LaRetea(r.info);
        inc(CadruAsteptat);
    }
    while(intre(ConfAsteptata, r.conf, CadruUrmator))
    {
        //confirma cadre intre ConfAsteptata si r.conf
        ntampon--;
        StopCeas(ConfAsteptata);
        inc(ConfAsteptata);
    }
    //la iesire din bucla ConfAsteptata = r.conf+1
break;

case EroareControl: break;      // ignora cadre eronate

```



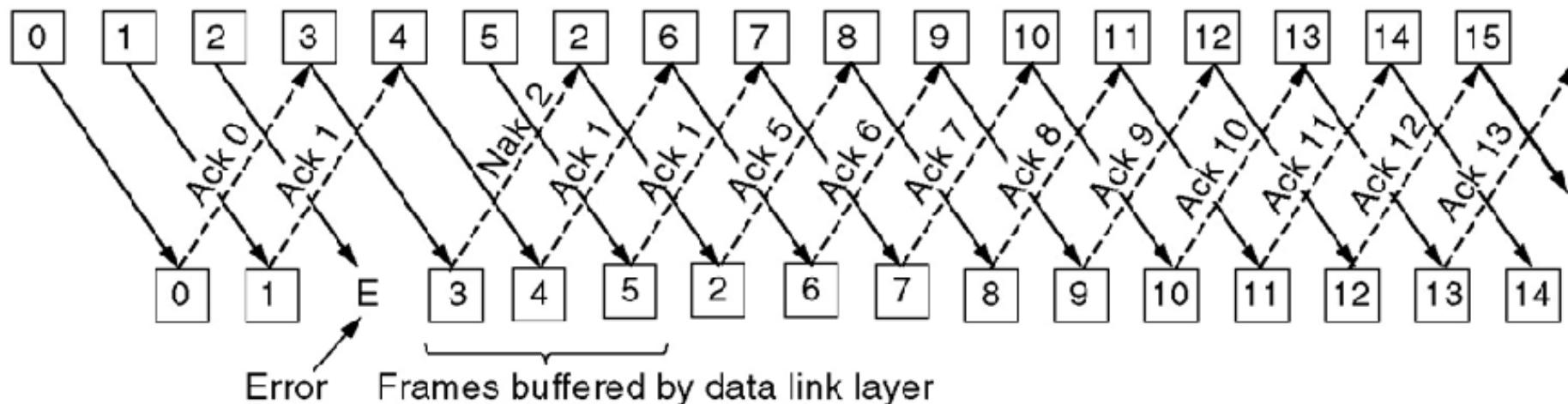


```
case TimeOut: // retransmite toate cadrele din
                // tampon (fereastra transmitatorului)
                // incepand cu pozitia ConfAsteptata
    CadruUrmator=ConfAsteptata;
    for(i=1;i<=ntampon;i++) {
        transmite(CadruUrmator);
        inc(CadruUrmator);
    }
             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

# Protocol cu retransmitere selectiva

Fereastra receptorului este supraunitara

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





## Dimensiunea ferestrei de receptie

Fereastra receptorului nu poate fi egală cu cea a transmițătorului

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



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



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



```
//continua case SosireCadru
    if (r.fel == nak) retransmite_cadru_cerut;
    trateaza_confirmare_cadre_eliberind_buffere;
    break;
case EroareControl:
    transmite_nak;
    break;
case TimeOut:
    retransmite_cadrul_corespunzator;
    // timeout este asociat nr. secv al unui cadru
}
activeaza_sau_deactiveaza_nivel_retea;
}forever;
}
```



# Exemple Protocoale Data Link

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



# Priority Flow Control – IEEE 802.1Qbb

- Standardul Ethernet (IEEE 802.3) este folosit pe scară largă
  - Asigura conexiuni punct-la-punct
  - În mod standard livrarea este best-effort.

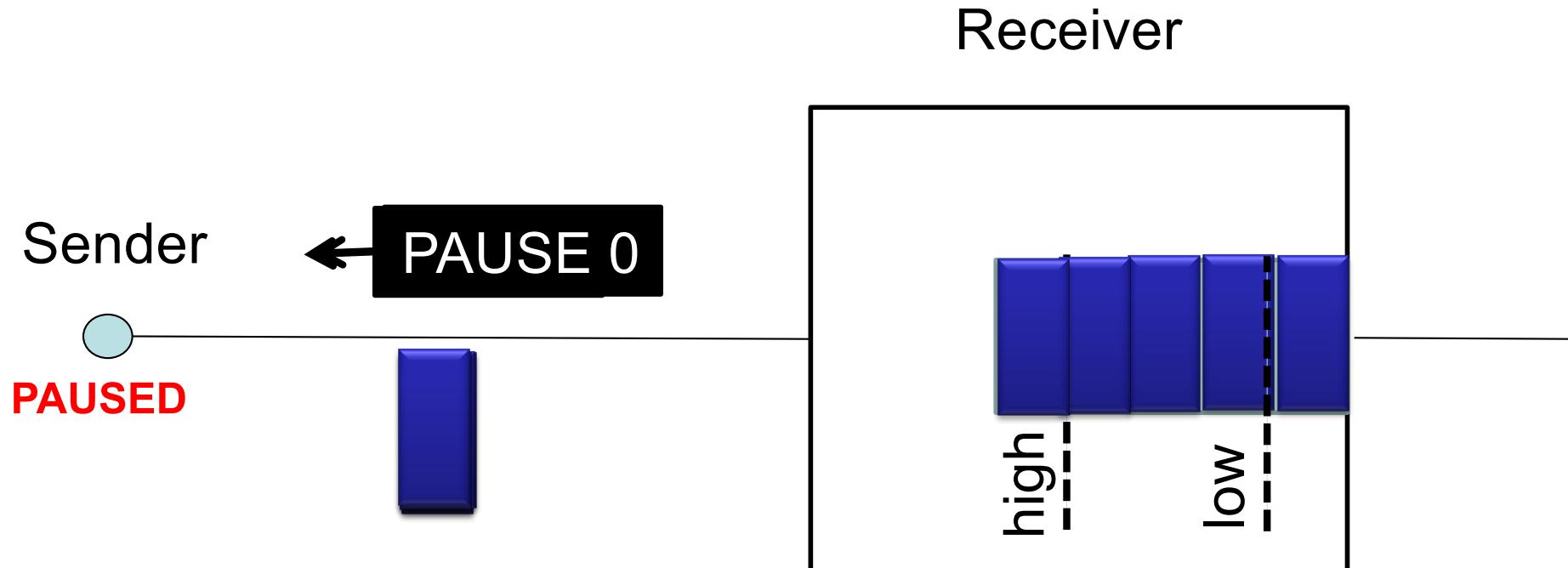
## Format Cadru Ethernet

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

- Priority Flow Control este o variantă a Ethernet care garantează livrarea cadrelor.



# Funcționare Priority Flow Control





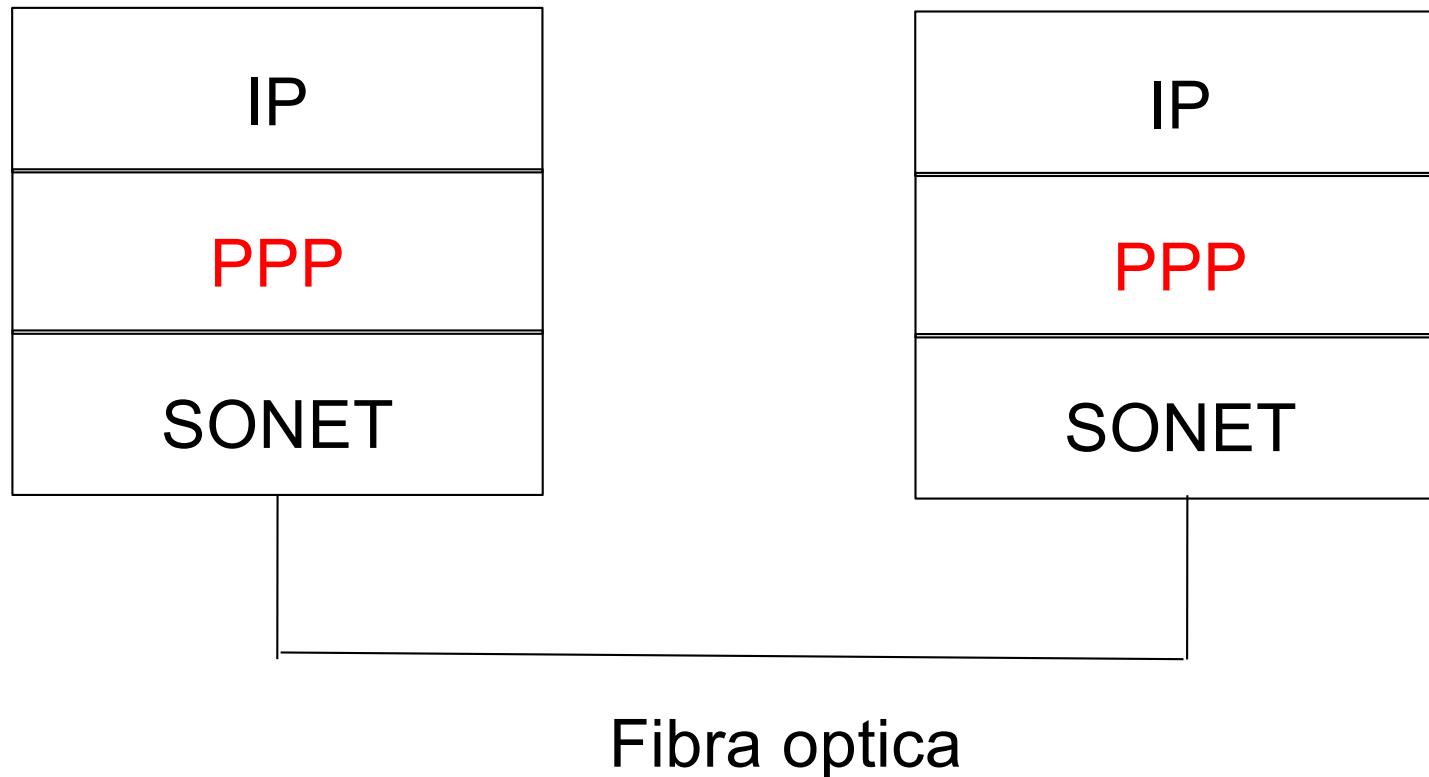
# Priority Flow Control

- PFC este folosit pentru a transporta RDMA peste Ethernet (ROCEv2)
  - RDMA necesita o rețea fără pierderi.
  - Folosit de Microsoft Azure pentru montarea diskurilor în rețea
- PFC are probleme de performanță
  - Tree saturation problem.
  - Head of line blocking.



# Legatura de date în Internet

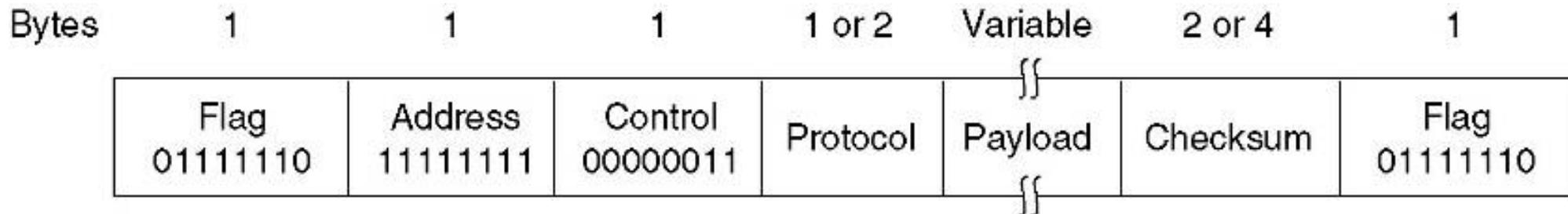
Comunicarea pe fibra optică





## PPP – Point to Point Protocol

- Oferă**
- incadrare
  - Link Control Protocol, LCP
  - Network Control Protocol, NCP



**Format de cadru PPP pentru modul nenumerotat**

Adresa 11111111 = toate stațiile acceptă cadrul

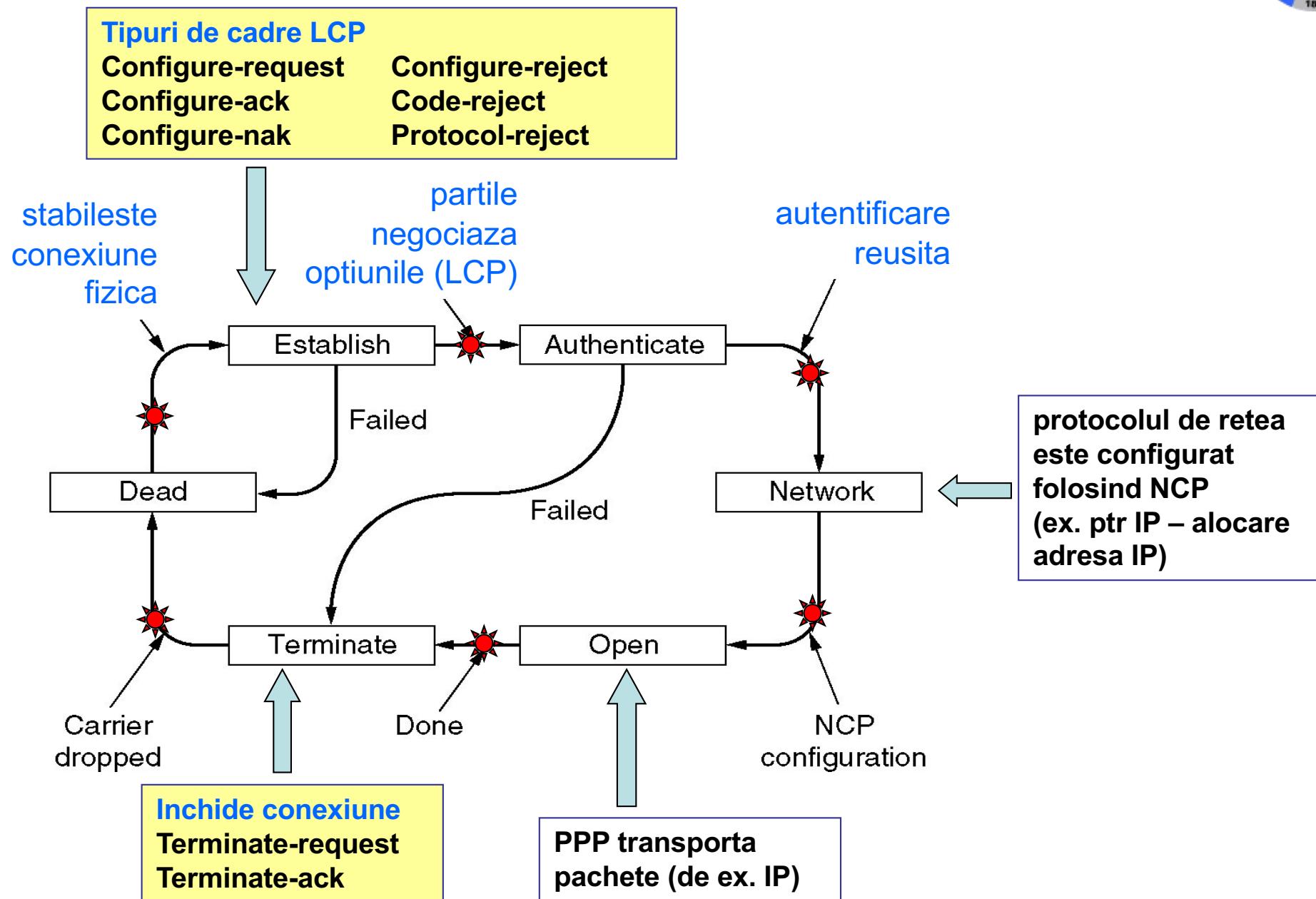
Control 00000011 = nenumerotat

Protocol = selectează dintre

LCP, NCP

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

# PPP – Point to Point Protocol (2)





# Tipuri de cadre LCP

Name	Direction	Description
Configure-request	I → 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**



# PPPoE

- Varianta a PPP encapsulată peste Ethernet; folosit pe scară largă
- PPPoE are două faze:
  - Descoperire (discovery)
  - Sesiune



# PPPoE Discovery

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

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



# Studiu individual

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

3.1 ASPECTE ALE PROIECTĂRII NIVELULUI LEGĂTURĂ DE DATE

3.2.2 Coduri detectoare de erori

3.3 PROTOCOALE ELEMENTARE PENTRU LEGĂTURA DE DATE

3.4 PROTOCOALE CU FEREASTRĂ GLISANTĂ

3.6 EXEMPLE DE PROTOCOALE ALE LEGĂTURII DE DATE

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

3.1 DATA LINK LAYER DESIGN ISSUES

3.2.2 Error-Detecting Codes

3.3 ELEMENTARY DATA LINK PROTOCOLS

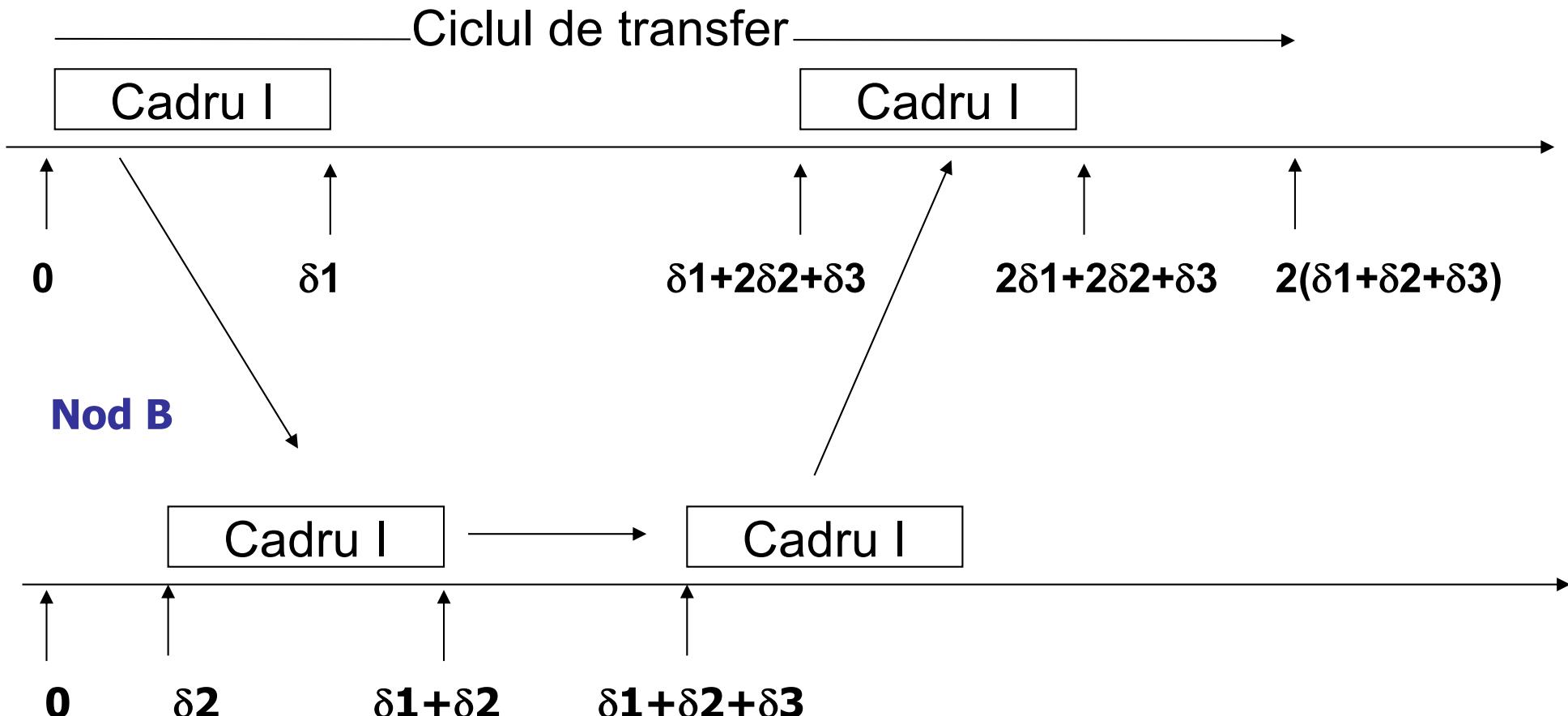
3.4 SLIDING WINDOW PROTOCOLS

3.5.1 Packet over SONET

## Analiza performanțelor protocolelor start-stop

**Transmitere cu confirmare în cadre de informație I**

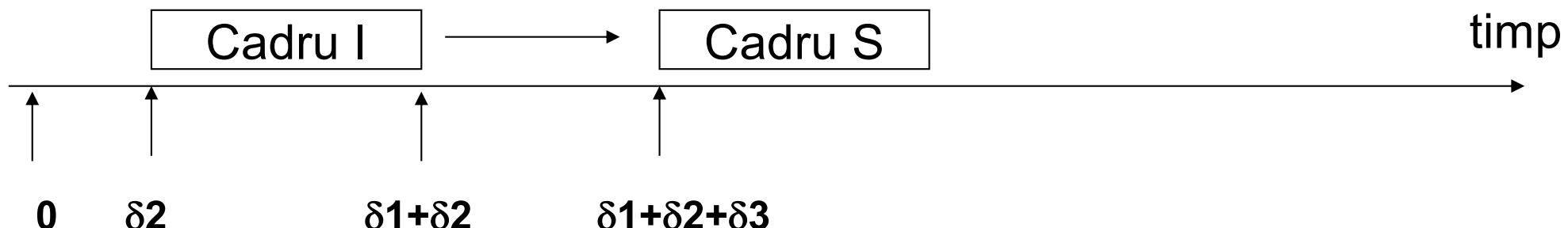
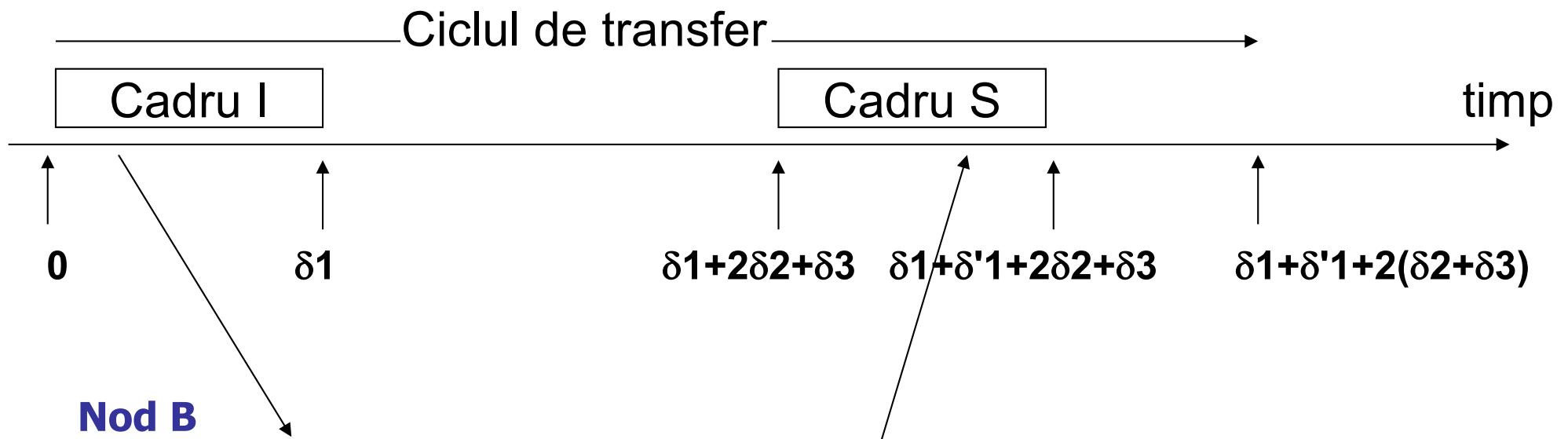
**Nod A**



## Transmitere cu confirmare în cadru supervisor S

$\delta'1$  – durata de transmitere a unui cadru S (sec)

Nod A





# Eficiența în absență erorilor

Cazul confirmării prin cadre S

- $\rho$  = timpul de transmitere a informației / durata unui ciclu de transfer
- $\rho = \delta_1 / (\delta_1 + \delta'_1 + 2(\delta_2 + \delta_3))$

Mai precis:

$$\rho = \frac{D/C}{2(\delta_2 + \delta_3) + (2H+D)/C} = \frac{D}{2(\delta_2 + \delta_3)C + 2H+D} = \frac{D}{LC + 2H + D}$$

unde: D - lungime câmp **date** din cadru I (nr biti)

H - lungime cadru S (= lung câmp **control** din cadru I)

C - **capacitatea** canalului (biti / sec)

L =  $2(\delta_2 + \delta_3)$ , **latență** (sec)



## Exemple

(1) Canal lent, distanță mică

$$D = 352 \text{ biți} \quad H = 48 \text{ biți}$$

$$\delta_2 = 5 \text{ msec}$$

$$\delta_3 = 1 \text{ msec}$$

$$\text{Capacitatea canalului} \quad C = 9600 \text{ biți / sec}$$

Rezultă:

$$L = 2(\delta_2 + \delta_3) = 12 \text{ msec}$$

$$\rho = D / (LC + 2H + D) = 0.625$$



## (2) Canal rapid, latență mare

$$D = 104 \text{ biți} \quad H = 48 \text{ biți}$$

$$\delta_2 = 100 \text{ msec}$$

$$\delta_3 = 1 \text{ msec}$$

Capacitatea canalului       $C = 150 * 10^6 \text{ biți / sec}$

Rezultă:

$$L = 202 \text{ msec}$$

$$\rho = D / (LC + 2H + D) = \mathbf{0.00000343}$$



# Problema

Legatura punct-la-punct de **128-kbps** este facuta intre Pamant si un satelit pe Marte. Distanta de la Pamant la Marte (cand sunt apropiate unul de altul) este de aproximativ **55 Gm** ( $55 \times 10^9$  m).

Datele traverseaza legatura la viteza luminii =  **$3 \times 10^8$  m/s.**

- (a) Calculati cel mai mic **RTT** (Round Trip Time) pentru legatura.
- (b) Calculati produsul **intarziere \* largime\_banda** pentru legatura.
- (c) O camera pe satelit face poze ale vecinatatii pe Marte si le trimit pe Pamant. In **cat timp** poate ajunge poza la Centrul de Control al Misiunii de pe Pamant? Fisierul are volumul de **5 MB**.



# Convenții

$1 \text{ B} = 8 \text{ biti}$

În transmisii de date

$$1 \text{ kb} = 10^3 \text{ b} = 1\ 000 \text{ b}$$

$$1 \text{ Mb} = 10^6 \text{ b} = 1\ 000\ 000 \text{ b}$$

$$1 \text{ Gb} = 10^9 \text{ b} = 1\ 000\ 000\ 000 \text{ b}$$

În calculatoare, volumul datelor

$$1 \text{ kb} = 2^{10} \text{ b} = 1\ 024 \text{ b}$$

$$1 \text{ Mb} = 2^{20} \text{ b} = 1\ 048\ 576 \text{ b}$$

$$1 \text{ Gb} = 2^{30} \text{ b} = 1\ 073\ 741\ 824 \text{ b}$$



# Raspuns

(a) **Intarzierea** de propagare pe legatura este **distanta/viteza**

$$(55 \cdot 10^9 \text{ m}) / (3 \cdot 10^8 \text{ m/s}) = 184 \text{ secunde.}$$

RTT este 368 secunde.

(b) Produsul **intarziere \* largime\_banda** pentru legatura

$$184 \text{ s} * 128 \cdot 10^3 \text{ kbps} = 23552000 \text{ kb} = 2.81 \text{ MB}$$

(c) **Transmiterea** a 5 MB = 41943040 biti de date dureaza

$$41943040 \text{ biti} / 128 \cdot 10^3 \text{ biti/s} = 328 \text{ s.}$$

Intarzierea de propagare + timp transmitere

$$184 \text{ s} + 328 \text{ s} = 512 \text{ s.}$$



## (Optional) Start stop cu erori de canal

Presupunem:

$p_I$  - probabilitatea ca I să fie recepționat fără erori

$p_S$  - probabilitatea ca S să fie receptionat fără erori

transmisiile succesive sunt independente

Un transfer este reușit dacă:

- transmisia fără erori detectabile (eveniment E1)
- receptia confirmării fără erori detectabile (E2)

Probabilitatea

$$p(E1 \text{ } \& \text{ } E2) = p_I \text{ } p_S$$



Livrare corectă => N cicluri de transfer (N-1 cu erori)

N = var. aleatoare cu distrib. geometrică:

$$\Pr\{N=k\} = p_I p_S (1 - p_I p_S)^{k-1}, \quad k \geq 1$$

Pentru k cicluri, eficiență

$$\rho_k = D / (D + 2H + CL) / k$$

Eficiență probabilă pentru start-stop

$$E(\rho) = \sum_{k=1,\omega} \rho_k * \Pr\{N=k\}$$

$$= \sum_{k=1,\omega} D / (D + 2H + CL) / k * p_I p_S (1 - p_I p_S)^{k-1}$$

$$= D / (D + 2H + CL) * p_I p_S + D / (D + 2H + CL) O (1 - p_I p_S)$$



Considerăm

- erorile succesive pe bit sunt independente
- probabilitatea de eroare la un bit este  $\varepsilon$ .

Pentru un canal binar simetric avem:

$$p_I p_S = (1 - \varepsilon)^{2H+D}$$

$$E(p) = D / (D + 2H + CL) * p_I p_S + D / (D + 2H + CL) O (1 - p_I p_S)$$

$$= D / (D + 2H + CL) (1 - \varepsilon)^{2H+D} + D / (D + 2H + CL) O (1 - (1 - \varepsilon)^{2H+D})$$



## (Optional) Lungimea optimă a câmpului de date

$O(1 - (1 - \varepsilon)^{2H+D})$  neglijabil.

Funcția care aproximează eficiența:

$$F(D) = D / (D + 2H + CL) (1 - \varepsilon)^{2H+D}$$

Pentru optim:  $(\partial / \partial D) \log F(D) = 0$

$$\log (1 - \varepsilon) + 1/D - 1 / (D+2H+CL) = 0$$

$$D^2 + (2H + CL) D + (2H + CL) / \log (1 - \varepsilon) = 0$$

cu rădăcina pozitivă aproximativă (pentru  $\varepsilon$  mic)

$$D^+ = \sqrt{2(H + CL / 2) / \varepsilon}$$



# Nivelul retea



# Cuprins

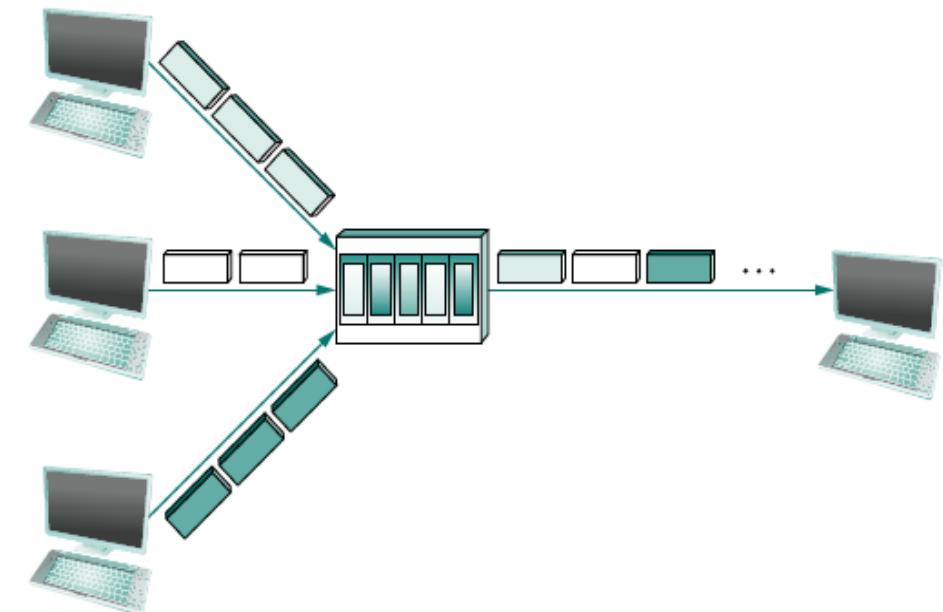
- de ce pachete ?
- organizarea internă
- protocolul IP – adresare și retransmiterea pachetelor
- algoritmi de dirijare
- protocoale de control
- structura internetului
  - protocolul OSPF – Open Shortest Path First
    - protocolul BGP – Border Gateway Protocol
- dirijarea în rețele ad hoc
- IPv6

# De ce este nevoie de pachete?

Legaturile unei rețele pot fi folosite simultan de transmisii paralele între mai multe perechi de noduri

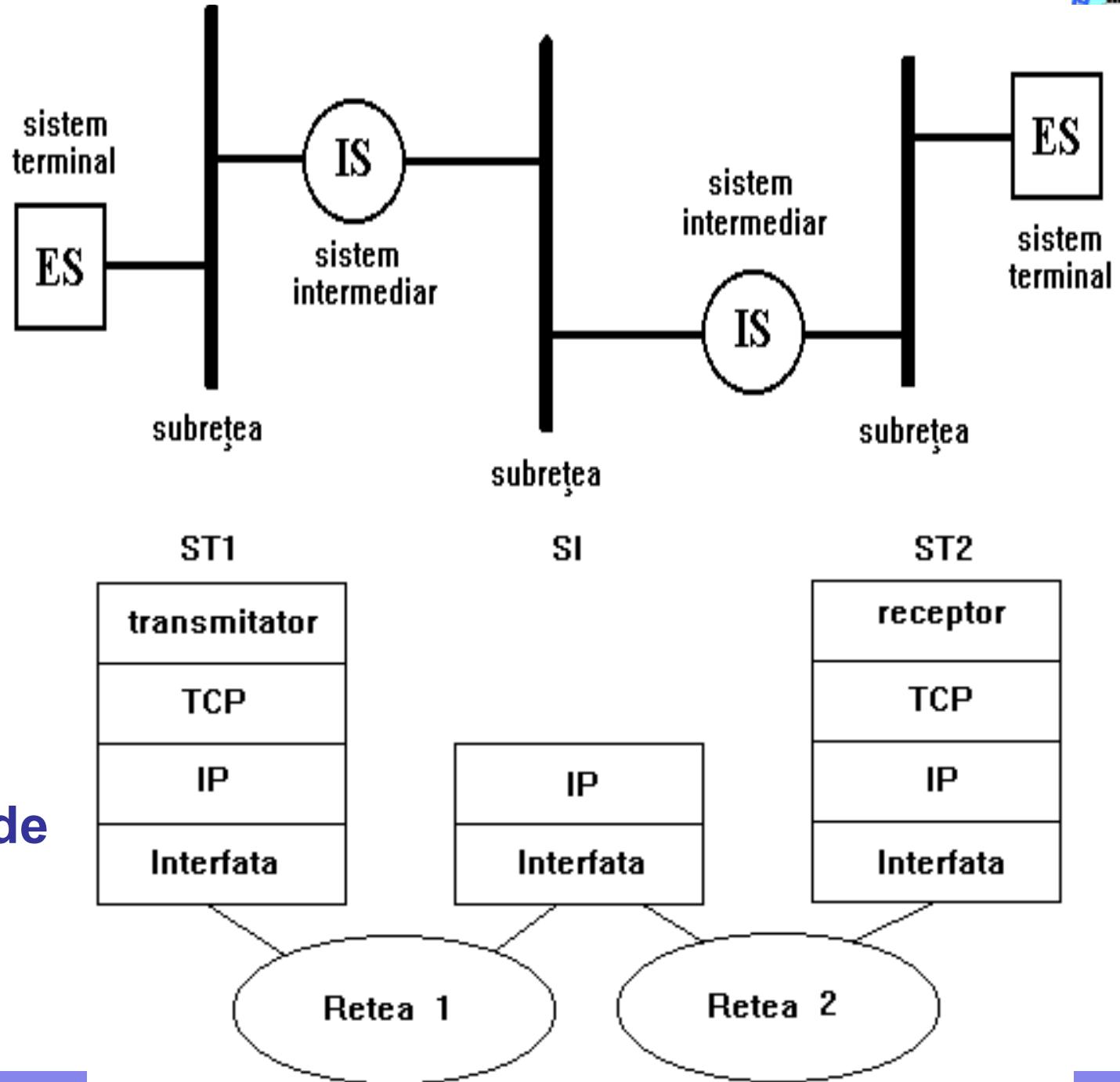
Multiplexare – se aloca transmisiilor

- sloturi de timp – STDM – Synchronous Time Division Multiplexing
- subcanale de frecvențe diferite – FDM – Frequency Division Multiplexing
- multiplexare statistică – STMD – sloturile sunt alocate la cerere;  
daca o singura pereche de noduri are de transmis, nu asteapta ptr. slot;
- pentru a evita acapararea legăturii, transmisia se face în pachete cu dimensiune limitată



## Modelul Internetului

Nivelele străbătute de pachete





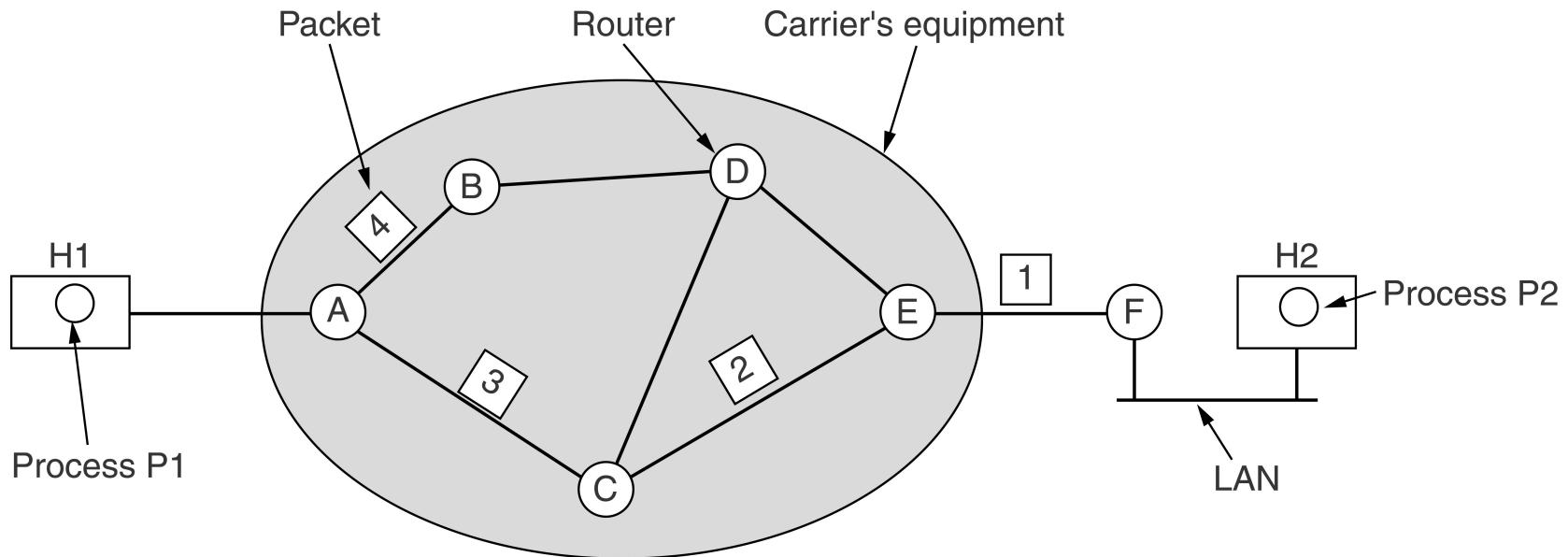
## Funcțiile nivelului rețea

- dirijarea pachetelor
- adresarea

## Aspecte principale

- servicii
  - ne-orientate pe conexiune
  - orientate pe conexiune
- organizarea internă corespunde tipurilor de servicii
  - datagrame
  - circuite virtuale

# Organizarea internă - datagrame



Folosita de  
pachetele  
1, 2 si 3

A's table

A	A
B	A
C	-
D	D
E	E
F	E

A	C
B	D
C	C
D	D
E	-
F	F

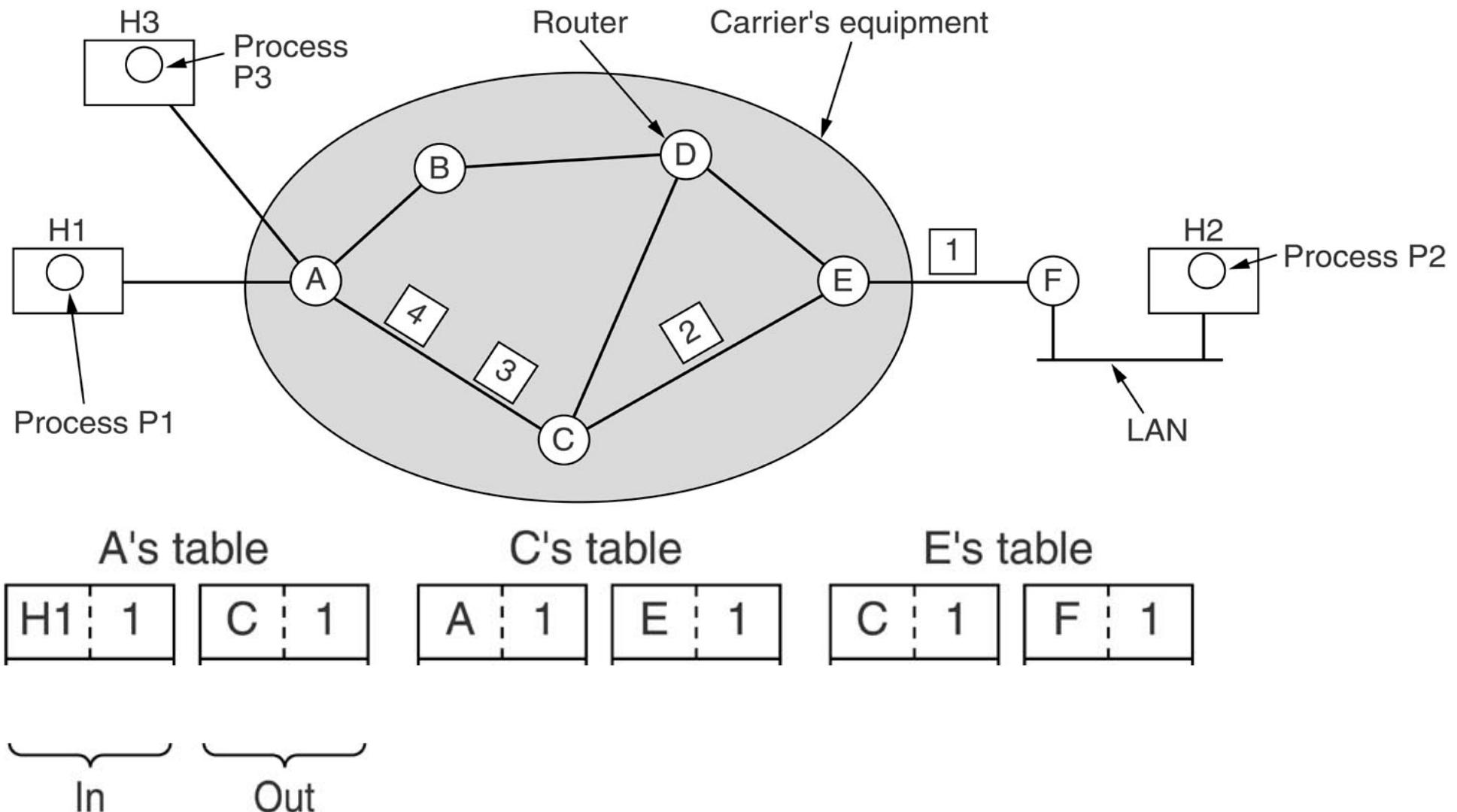
Folosita de  
pachetul 4



# Caracteristici datagrame

- Pachetul contine **toate** informațiile necesare ruterelor pentru “pasarea” lui către destinație (inclusiv adresele sursă și destinație)
- Organizare **fără conexiune**
  - Un calculator gazda (*host*) poate trimite pachetul **oricand și oriunde** în rețea
- Nu asigura **corectitudinea**
  - Transmisorul nu poate să stie dacă pachetul este livrat sau dacă destinatarul mai este conectat
- Nu **pastrează ordinea** pachetelor
  - Pachetele sunt dirijate independent unele de altele
- **Robusta**
  - La defectarea unei legături se gasesc rute alternative
- Utilizare largă în Internet

# Organizarea internă – circuit virtual



**Nodul A re-numeroteaza circuitul virtual**



# Caracteristici circuit virtual

- Organizare **bazata pe conexiune**
  - Transferul incepe dupa stabilirea conexiunii
  - Pachetul contine id-ul conexiunii
  - Se pot **aloca resurse** la stabilirea conexiunii (memorie tampon pentru pachete)
- Transmitatorul stie
  - ca exista o conexiune
  - ca receptorul este **pregatit** sa primeasca pachete
- Se pastreaza **ordinea** pachetelor
- Se poate controla **fluxul**
- Folosit in
  - MPLS (MultiProtocol Label Switching)
  - retele virtuale private (VPN – Virtual Private Network)



# Protocolul IP

- Are
  - o **schema de adresare** care permite identificarea oricărui calculator din Internet
  - un model - **datagrama** - pentru transmiterea datelor de la un nod gazda la altul
    - **datagrama = pachet**
- Modelul de serviciu – **best effort**
  - rețeaua “face toate eforturile” să livreze pachetele la destinație
  - nu face nici o încercare să corecteze erorile



# Protocol IPv4 – formatul pachetului

SERVICE TYPE = precedence (3), delay, throughput, reliability, cost

PROTOCOL = (TCP, UDP, etc.)

IDENTIFICATION Id datagrama de care aparține fragmentul

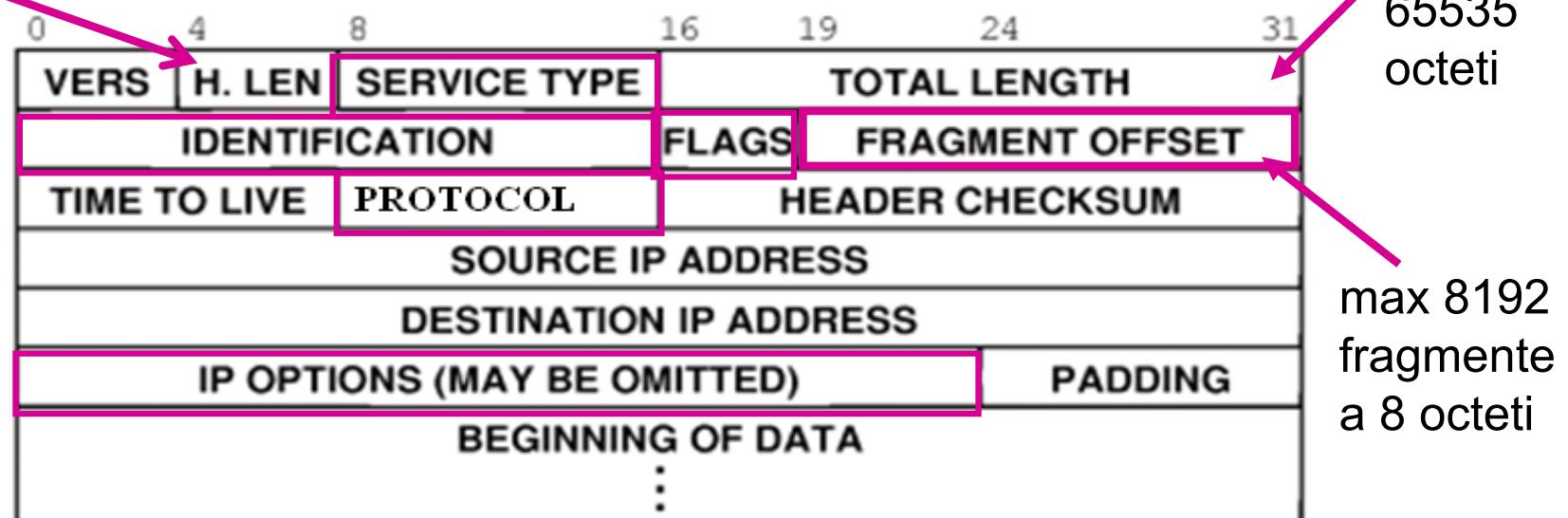
FRAGMENT OFFSET pozitia fragmentului in pachet

FLAGS **DF** = Don't Fragment / **MF** = More Fragments

## OPTIONS:

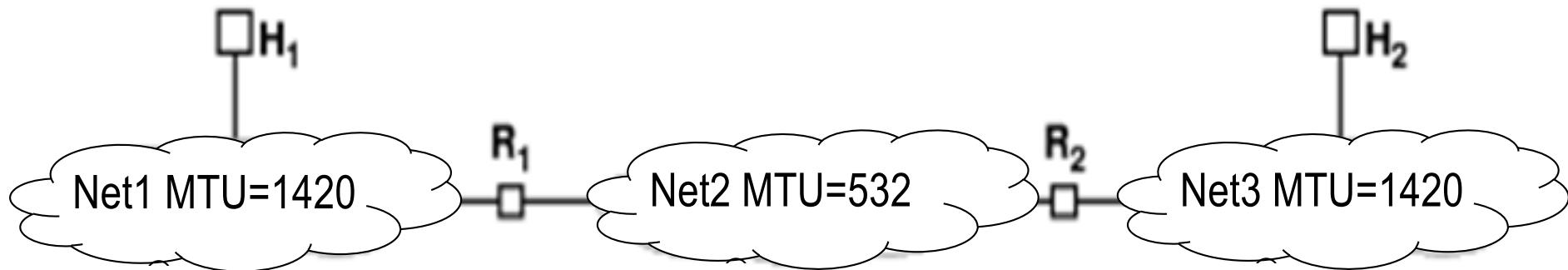
- Security
- Strict source routing
- Loose source routing
- Record route
- Timestamp

numar  
cuvinte  
de 32  
biti

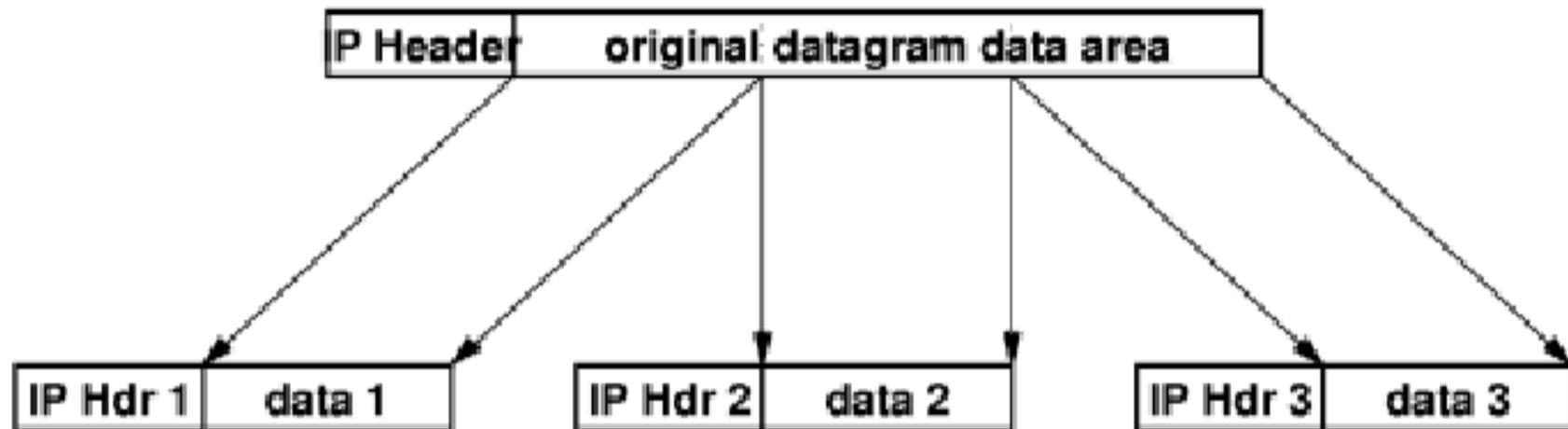




# MTU – Maximum Transmission Unit



**Fragmentarea se face la  $R_1$**



**Reasamblarea se face la  $H_2$  – de ce?**

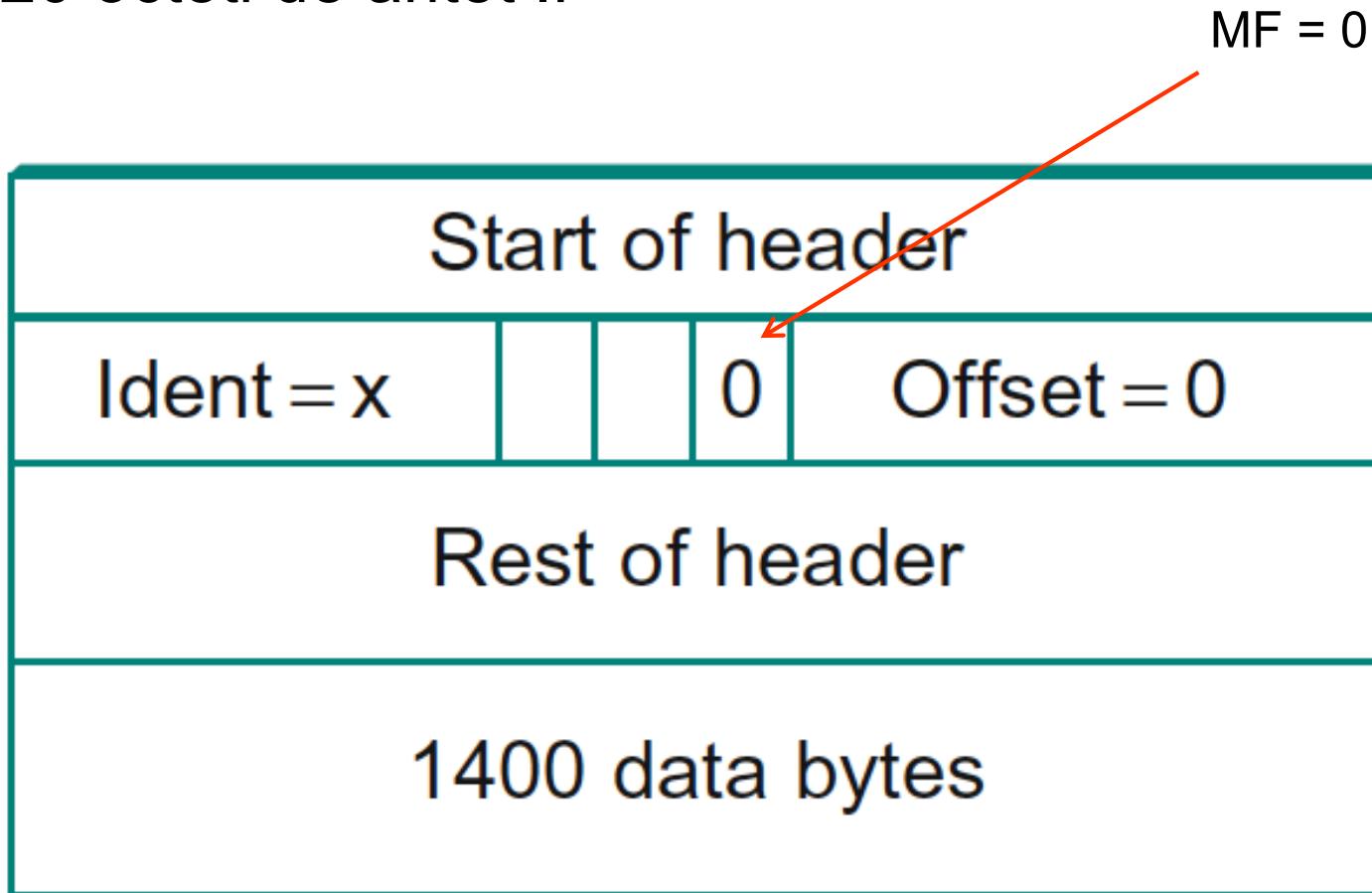


# Campurile de antet pentru fragmentare (1)

(a) pachet transmis ne-fragmentat prin Net1 cu MTU = 1420

are: 1400 octeti de date

20 octeti de antet IP





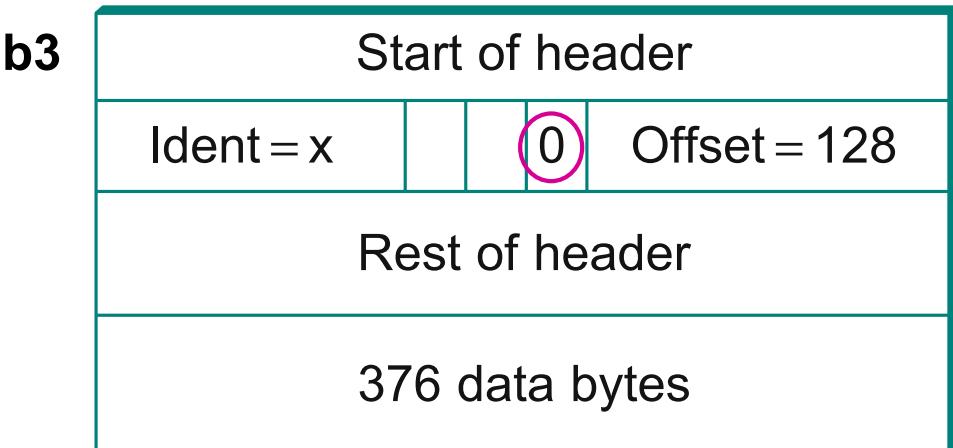
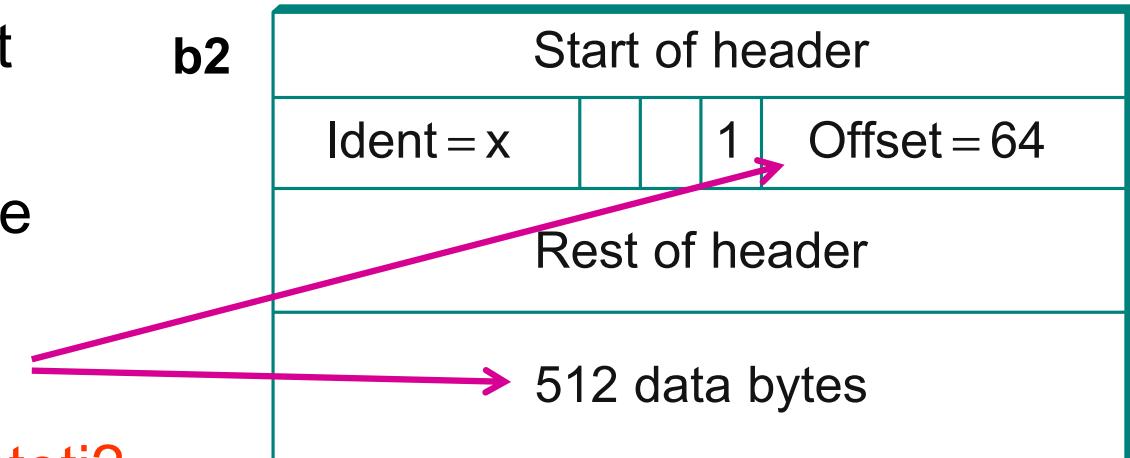
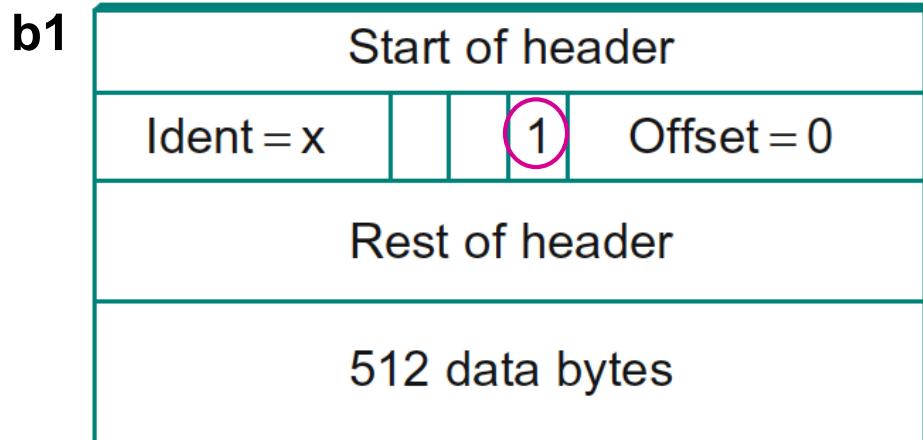
# Campurile de antet pentru fragmentare (2)

Net2 are MTU = 532

Ruterul R1 segmenteaza pachetul in 3 fragmente (fig b1-b3)

- lungimea fiecarui segment este multiplu de 8 octeti
- **offset**-ul numara grupuri de cate 8 octeti
- Offset =  $64 = 512 / 8$

unde e restul pana la 532 octeti?





# Clase de adrese IP

Fiecare **nod** (gazda/ruter) are o **adresa IP** asociată cu **interfata lui de rețea**

Un nod cu mai multe interfețe are mai multe adrese IP (ex. rutere)

0	7 8	hostid	31	clasa A
0	15 16	hostid	31	clasa B
10	netid	hostid	31	clasa C
110	netid	hostid	31	clasa D
1110	adresa multicast		31	clasa E
11110	rezervat pentru utilizare viitoare		31	

Clasa de adrese	Biți în prefix	Număr maxim de rețele	Biți în sufix	Număr maxim de gazde per rețea
A	7	128	24	167777216
B	14	16384	16	65536
C	21	2097152	8	256



# Câteva adrese speciale

Prefix (rețea)	Suffix (gazdă)	Semnificație	Scop
toți 0	toți 0	acest calculator	Folosită cand nodul încă nu are o adresă
network	toți 0	network	Identifică rețeaua
network	toți 1	broadcast	broadcast în rețeaua specificată
toți 1	toți 1	broadcast	broadcast în rețeaua locală
127	orice	loopback	testare

## Notări pentru adrese

binară      11000010 00011000 00010001 00000100

zecimală    194.24.17.4



# Tabele de dirijare

Orice pachet conține o **adresa IP** a destinatarului, cu două părți

**<adresa\_retea, adresa\_nod>**

Un pachet este transmis de la sursă la destinație trecând prin noduri intermediare (**rutere**), fiecare legând între ele cel puțin două rețele

**Rol ruter** – primește un pachet și

- îl livrează **gazdei** de destinație (dacă este în aceeași rețea)
  - Toate nodurile care au aceeași **adresa\_retea** sunt situate în **aceeași rețea fizică** și pot comunica direct prin **legătura de date** (transmit cadre)
- altfel, il re-transmite (**forward**) către un alt nod **NextHop**
  - Folosește **tabela de dirijare (rutare)** care are intrări de forma

**<adresa\_retea, NextHop>**



# Algoritm de *forwarding IP*

Extrage **<adresa\_retea, adresa\_gazda>** destinație din datagrama

Caută o intrare cu **adresa\_retea** în tabela de dirijare

**if** **adresa\_retea** apare în tabela de dirijare

**if** **adresa\_retea** indică o rețea direct conectată **then**

        transmite datagrama direct la **adresa\_gazda**

**else** transmite datagrama urmatorului ruter (**Next Hop**)

**else** transmite datagrama unui **ruter implicit**

Adresarea **ierarhica** **<adresa\_retea, adresa\_gazda>** reduce numărul de intrări în tabela de dirijare (o intrare pentru o **adresa\_retea**)

În practică, tabelele de rutare sunt separate pe clase de adrese

- Căutare prin: **indexare** (A și B) sau **hashing** (C)

# Exemplu

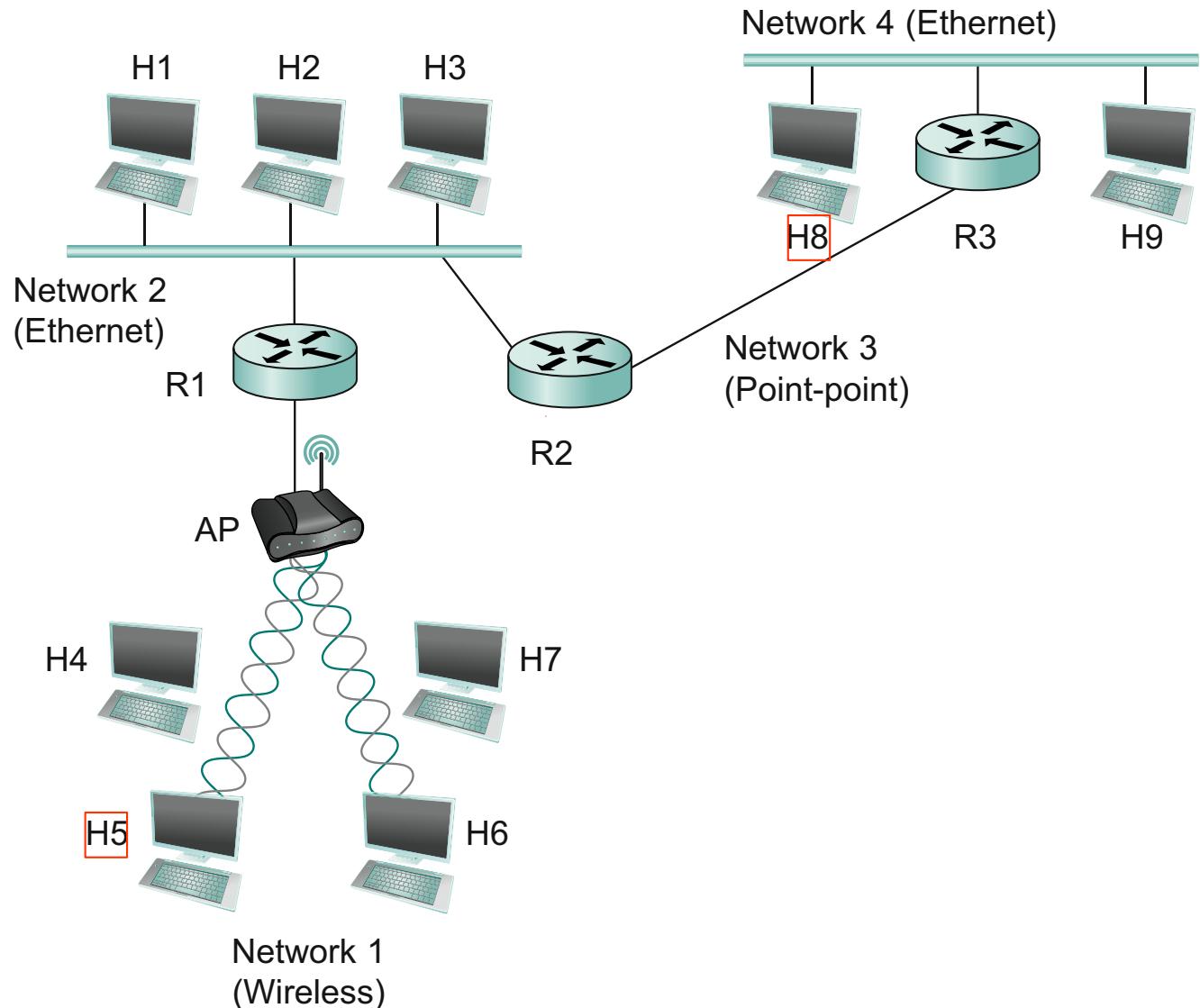
Transferul unei datagrame între **H5** și **H8**

**H5** are legatura (printr-un punct de acces AP) la ruterul **R1**

Pachetul trece prin rutele **R1**, **R2** și **R3**

**R3** este în aceeași retea cu **H8**

și îl livrează direct datagrama





# Conversie adresa IP – adresa fizică

La livrarea directă (destinatar în aceeași rețea) se folosește adresa fizică a receptorului (ptr care se cunoaște adresa IP)

**Nu există** o legătură biunivocă între adresa fizică și adresa IP

De ex.

- adresa IP are 32 biti
- adresa Ethernet are 48 biti

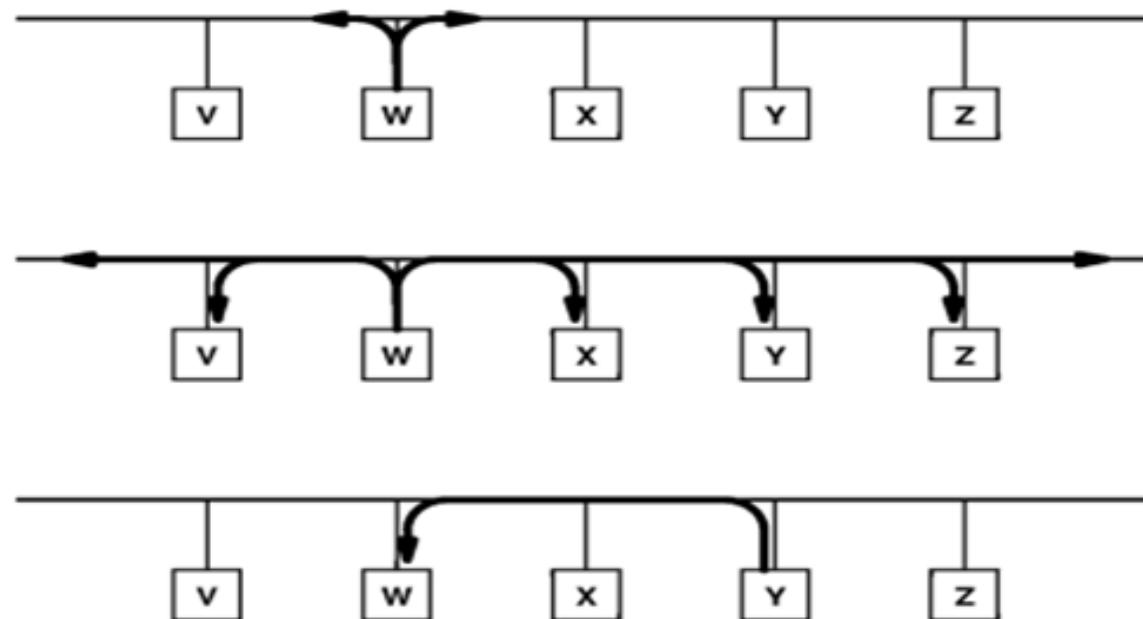
Pentru mapare se pot folosi

- tabele de corespondență
- formule de calcul
- schimb de mesaje
  - ex. ARP - Address Resolution Protocol
    - Face maparea între adresa de protocol și adresa hardware

# ARP - Address Resolution Protocol

Sevență de mesaje pentru a afla adresa fizică

- w difuzează o cerere ARP cu adresa IP cunoscută
- cererea este primită de toate gazdele din rețea locală
- y – care are adresa IP respectivă – trimite ca răspuns adresa fizică (ex. Ethernet)





# Conversie adresa IP – adresa fizică

0	8	15	16	31			
Hardware Type		Protocol Type					
HLEN	PLEN	Operation					
Sender HA (octets 0-3)							
Sender HA (octets 4-5)		Sender IP (octets 0-1)					
Sender IP (octets 2-3)		Target HA (octets 0-1)					
Target HA (octets 2-5)							
Target IP (octets 0-3)							

Format mesaje ARP – conceput pentru diverse categorii de adrese

- **Hardware type:** 1 pentru Ethernet
- **Protocol type:** 0x0800 pentru IP (0000.1000.0000.0000)
- **HLEN** - Hardware len: 6 octeti pentru Ethernet
- **PLEN** - Protocol len: 4 octeti pentru IP
- **Operation:** 1=cerere, 2=raspuns



## Subretele

- Regula “o adresă pentru fiecare rețea fizică separată”
  - foloseste ineficient spatiul de adrese
    - o retea de clasa C cu 2 noduri consumă doar 2 din totalul de 255 de adrese de nod
    - o retea de clasa B cu peste 255 noduri ocupă peste 64000 de adrese indiferent dacă le folosește pe toate sau nu
- solutia 1 – subretele
  - de ex. o retea clasa B este împărțita în mai multe subretele apropiate geografic
- solutia 2 - adrese fara clase



# CIDR – Classless InterDomain Routing

**Ideea:** alocă spațiul de adrese IP în blocuri de lungimi diferite

Notația specială pentru **adresa de rețea CIDR**

194.24.0.0/21 → din cei 32 de biți ai adresei IP

**adresa\_retea** ocupă 21 biți

**adresa\_gazda** ocupă 11 biti

<b>University</b>	<b>First address</b>	<b>Last address</b>	<b>How many</b>	<b>Written as</b>
Cambridge	194.24.0.0	194.24.7. <b>255</b>	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20



# CIDR – Exemplu

Pentru a afla **adresa\_retea** se folosesc **măști**

**Cambridge:** Adresă 11000010 00011000 00000000 00000000

Mască **11111111 11111111 11111000 00000000**

**Edinburgh:** Adresă 11000010 00011000 00001000 00000000

Mască **11111111 11111111 11111100 00000000**

**Oxford:** Adresă 11000010 00011000 00010000 00000000

Mască **11111111 11111111 11110000 00000000**

University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7. <b>255</b>	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20



# CIDR – reguli de alocare a adreselor

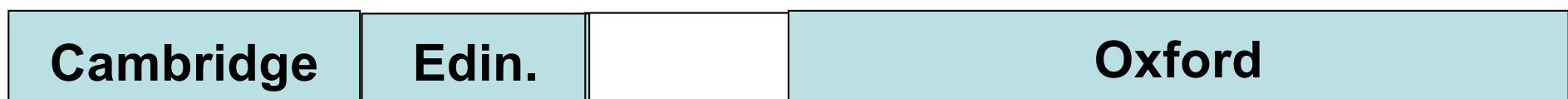
University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7. <b>255</b>	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Reguli pentru a avea o mască pentru un bloc de adrese

- lungimea blocului trebuie să fie o putere a lui 2
- toate adresele din bloc au aceeași adresa\_retea → adresa de inceput a blocului de adrese trebuie să fie multiplu de dimensiunea acestuia

Ex.: zona de adrese pentru Oxford începe la o frontieră de 4096 octeți

0                    2048                    3072                    4096





# Algoritm *forwarding*

Intrare în tabela de rutare - (adresa\_retea, Masca, NextHop)

Algoritmul alege intrarea pentru care

$$(\text{Adresa\_IP AND Masca}) = \text{adresa\_retea}$$

Ex. Sosește pachet cu adresa\_IP = 194.24.17.4

compara cu Cambridge /21 – adresa\_retea = 194.24.0.0

adresa\_retea: 11000010 00011000 00000000 00000000

Masca: 11111111 11111111 11111000 00000000

adresa\_IP: 11000010 00011000 00010001 00000100

Adresa\_IP AND Masca:

11000010 00011000 00010000 00000000

→ = 194.24.16.0 ≠ 194.24.0.0 → nepotrivire



## Algoritm **forwarding** (2)

Ex. Sosește pachet cu adresa\_IP = **194.24.17.4**

Cambridge /21 – **adresa\_rețea** = 194.24.0.0

(**Adresa\_IP AND Masca**) = 194.24.16.0 → **nepotrivire**

Edinburgh /22 - **adresa\_rețea** = 194.24.8.0

(**Adresa\_IP AND Masca**) = 194.24.16.0 → **nepotrivire**

Oxford /20 - **adresa\_rețea** = 194.24.16.0

(**Adresa\_IP AND Masca**) = 194.24.16.0 → **potrivire**

Dacă nu sunt alte potriviri -> folosește intrarea pentru Oxford



# Potriviri multiple

Prefixe de lungimi diferite

→ unele **adrese IP** se pot potrivi cu mai multe **adrese\_retea** din tabela de dirijare

Ex.

adresa_IP	171.69.10.5	se potrivește cu
adresele de rețea	171.69.0.0/16	
	171.69.10.0/24	

Regula: se alege potrivirea “mai lungă”



# Reducere dimensiune tabelă rutare

Soluție - agregarea unor adrese

Consideram adresele de retea:

C - Cambridge: 194.24.0.0/21

E - Edingurgh: 194.24.8.0/22

O - Oxford: 194.24.16.0/20

C: adresa\_retea 11000010 00011000 00000000 00000000

E: adresa\_retea 11000010 00011000 00001000 00000000

O: adresa\_retea 11000010 00011000 00010000 00000000

Presupunem: pentru rutare la C, E, O, nodul NewYork are în tabela de dirijare **același** NextHop

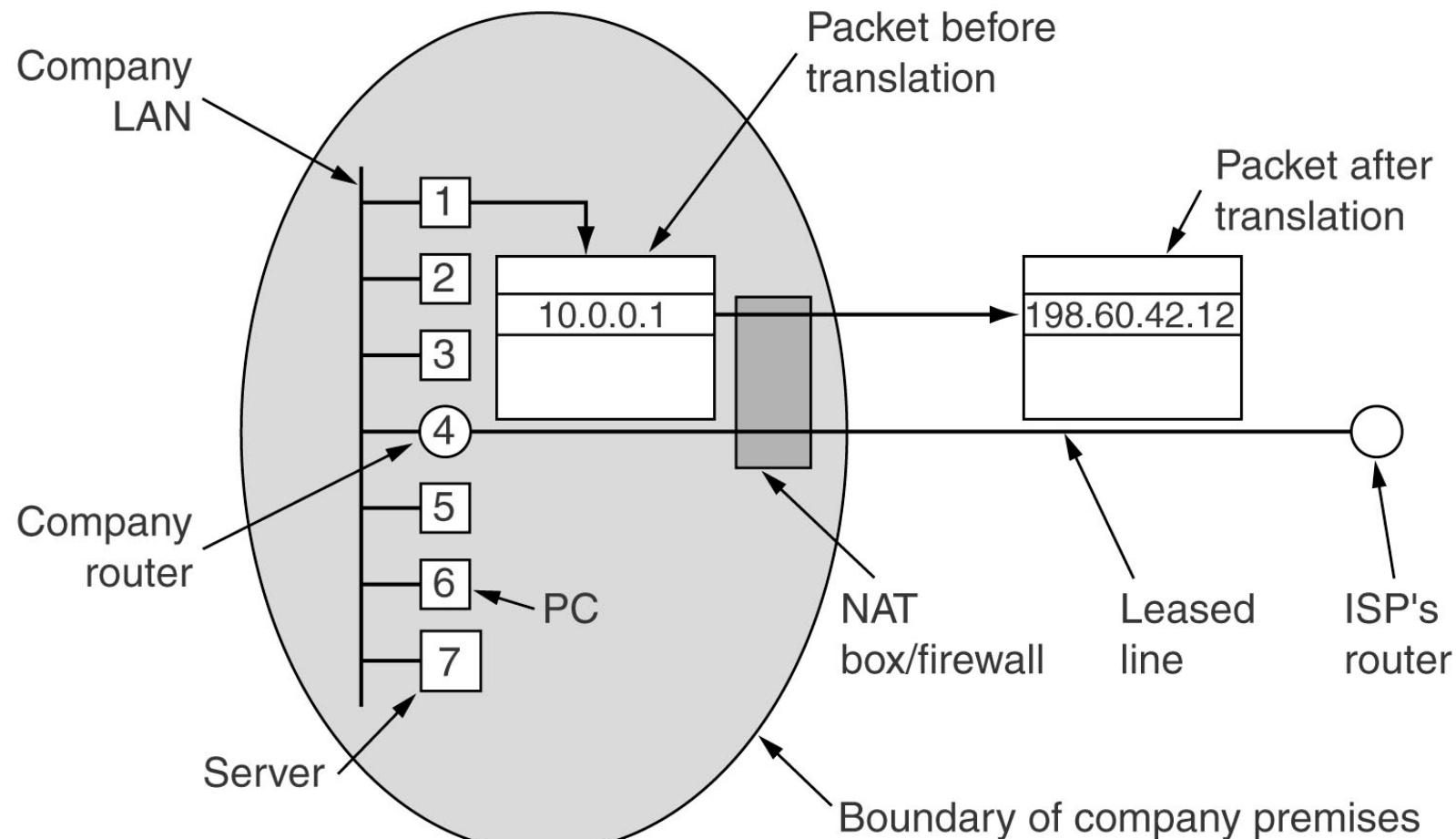
Tabela de dirijare poate include o singură intrare comună pentru adresa\_retea 11000010 00011000 00000000 00000000  
adică **194.24.0.0/19**

# NAT – Network Address Translation

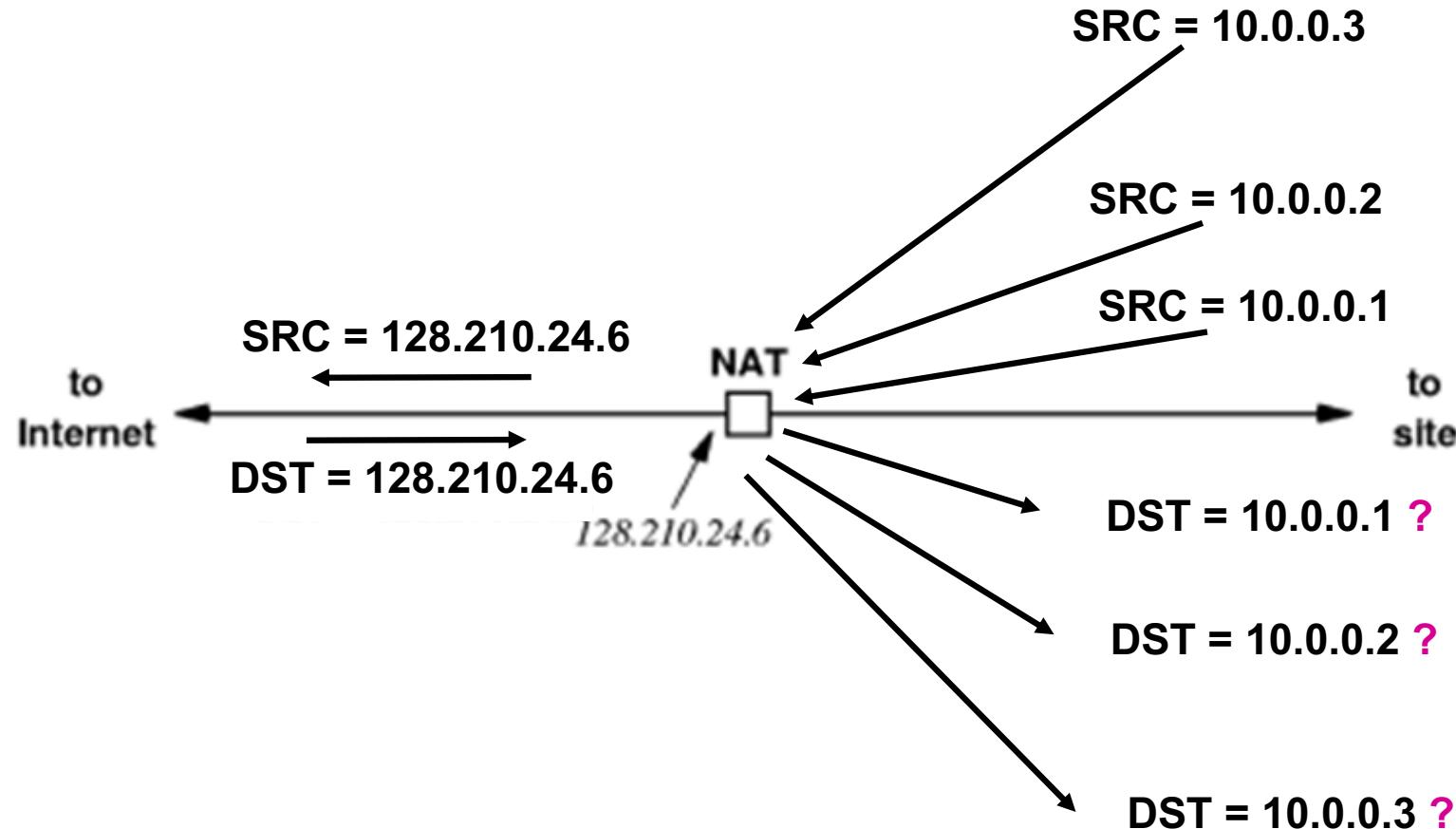
O adresă pentru mai multe calculatoare

Folosește **adrese locale** (private sau non-rutabile)

NAT translatează între adresa privată și o adresă globală



# Translatarea adresa globală → adresa privată



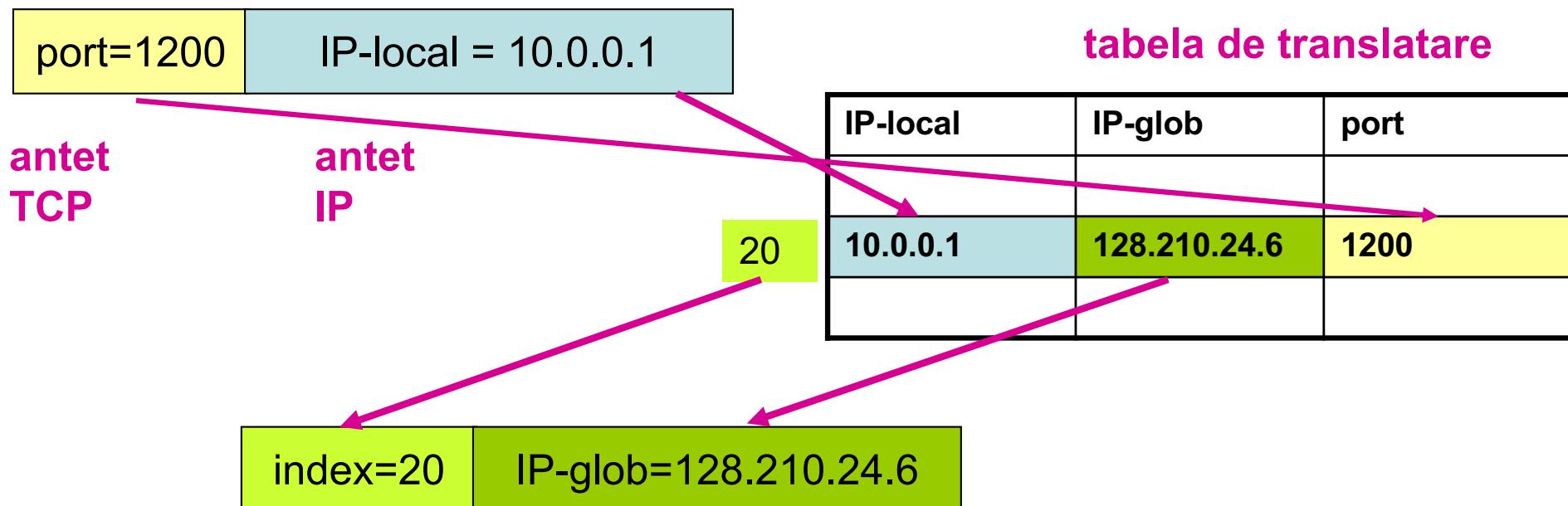
# Principiul NAT

Folosește

adresa IP + număr port transmititor  
tabela de translatare

Transmisie

înlocuiește adresa IP locală cu o adresă IP globală  
memorează (in tabela de translatare) corespondența și număr port  
înlocuiește număr port cu index în tabela translatare  
re-compune sumele de control IP și TCP





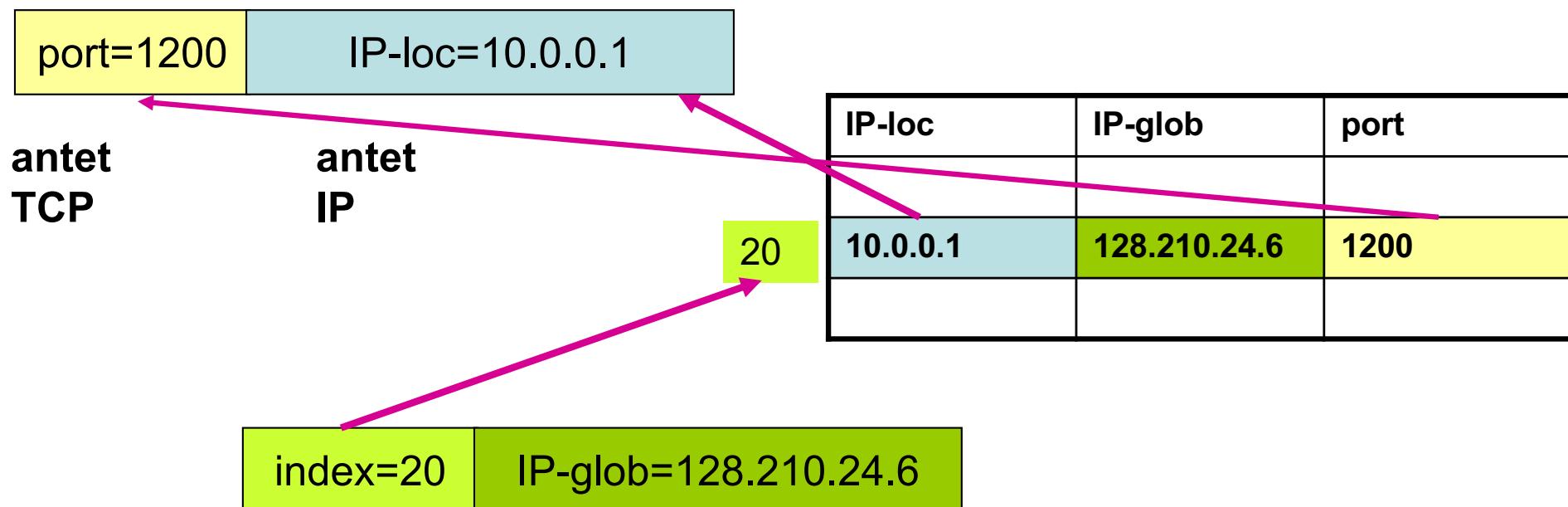
## Recepție

obține număr port din pachet (= index în tabela translatare)

extrage adresa IP locală și număr port

înlocuiește adresa IP și număr port din pachet

re-calculează sumele de control IP și TCP



# ICMP- Internet Control Message Protocol

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp



ICMP folosește IP ptr transmisie & IP folosește ICMP pentru raportare de erori

Test accesibilitate (**ping** trimite ICMP Echo și așteaptă un timp răspunsul)

Trasare ruta (**traceroute** trimite serie de datagrame cu valori TIME TO LIVE crescătoare și primește mesaje ICMP Time exceeded din care extrage adresa ruterului)



# Folosire ICMP pentru aflare path MTU

**Path MTU = Maximum Transmission Unit minimă pentru o cale**

- Folosește mesaj eroare ICMP = fragmentare necesară dar nepermisă
  - Sursa trimite probe cu DF în datagrama IP
  - Dacă **datagrama > MTU** => sursa primește eroarea ICMP **Destination Unreachable** cu **Fragmentation Needed and Don't Fragment was Set**
  - Sursa trimite probe mai scurte



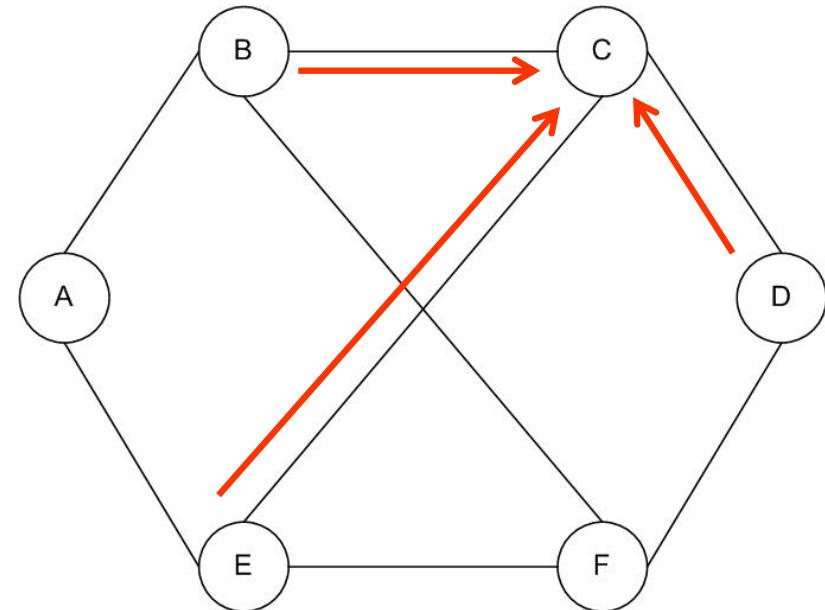
# Dirijarea - clasificare

- Fără tabele de dirijare
  - inundarea
  - hot potato
- Cu tabele de dirijare – criterii diverse
  - adaptarea la condițiile de trafic
    - statică
    - dinamică
  - locul unde se fac calculele
    - descentralizată
    - centralizată
    - distribuită
  - criterii de dirijare
    - calea cea mai scurtă
    - întârzierea medie globală
    - folosirea eficientă a resurselor
    - echitabilitatea
  - informații schimbate între noduri
    - starea legăturii
    - vectorul distanțelor
  - tipul rețelei
    - uniformă
    - ierarhică

# Vectorii distanțelor

Algoritm distribuit !

Fiecare nod trimite periodic vecinilor săi o lista cu distantele de la el la celelalte noduri.



Următorii vectori au fost primiți de nodul **C** (lista include distanțele de la B, D, E la nodurile A, B, C, D, E, F, în această ordine):

De la B: (5, 0, 8, 12, 6, 2);

De la D: (16, 12, 6, 0, 9, 10);

De la E: (7, 6, 3, 9, 0, 4).

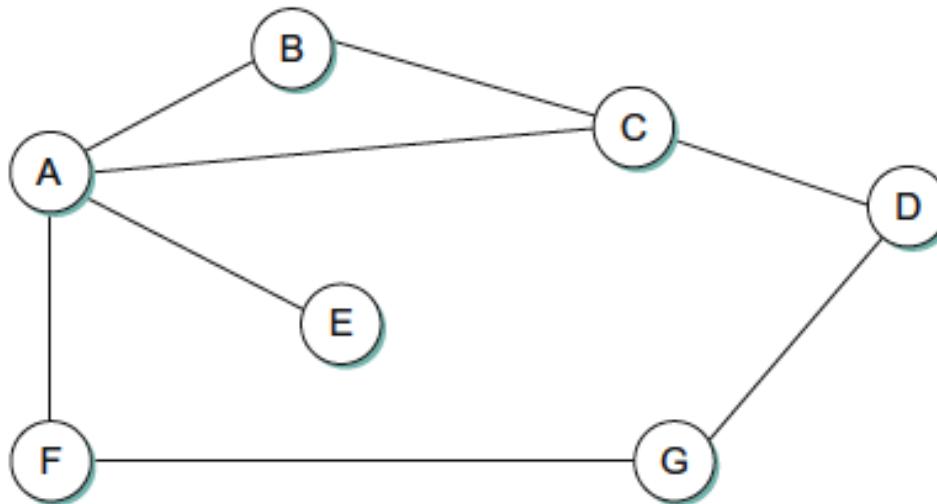


De la La	B	D	E
A	5	16	7
B	0	12	6
C	8	6	3
D	12	0	9
E	6	9	0
F	2	10	4

În plus, întârzierea măsurată de la C la B, D și E este 6, 3 și 5 respectiv.

Costurile de la C La ↓	Prin →	B	D	E		Cost Min	Pas urmator
A		$5 + 6$	$16 + 3$	$7 + 5$		11	B
B		$0 + 6$	$12 + 3$	$6 + 5$		6	B
C		-	-	-		0	-
D		$12 + 6$	$0 + 3$	$9 + 5$		3	D
E		$6 + 6$	$9 + 3$	$0 + 5$		5	E
F		$2 + 6$	$10 + 3$	$4 + 5$		8	B

# Problema numărării la infinit



De la A B C D E F G cost 1 pentru orice legatura

1 2 2 3 0 2 3 ← distante initiale la E

$\infty$  **2** **2** 3 0 2 3 legatura A – E cade

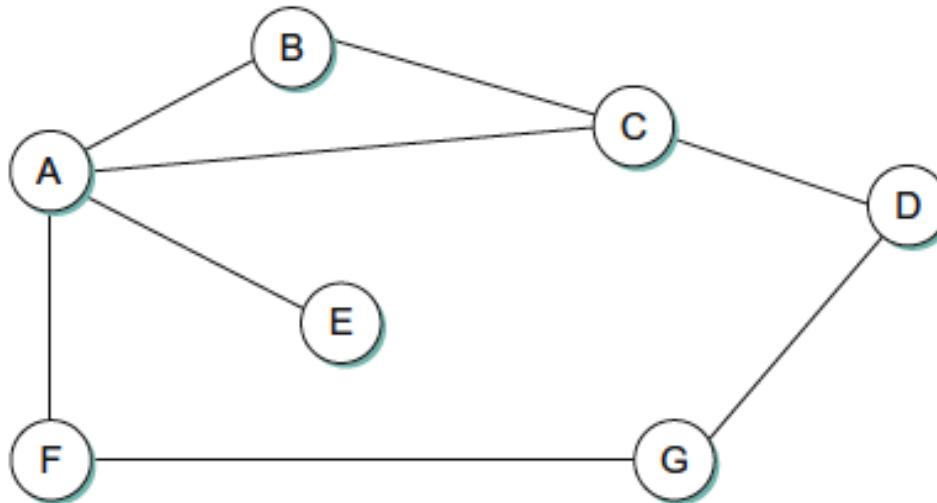
distanțele catre E anunțate de: **A =  $\infty$ , B = 2, C = 2**

In functie de **ordinea evenimentelor**, se pot face modificarile:

$\infty$  **3** **2** 3 0 2 3 B alege ruta prin C, dist=3

B anunta dist. 3 lui A

## Problema numărării la infinit (2)



De la A B C D E F G

**4 3 2 3 0 2 3**

A alege ruta prin B dist=4

A anunta dist. 4 lui C

**4 3 5 3 0 2 3**

C recalculeaza dist=5

distanțele cresc teoretic la infinit

practic, se poate limita la un numar > diametru graf (ex. 16)

## Problema numărării la infinit - solutii (3)

**split horizon:** noile distante nu se trimit vecinului prin care trec actualele rute

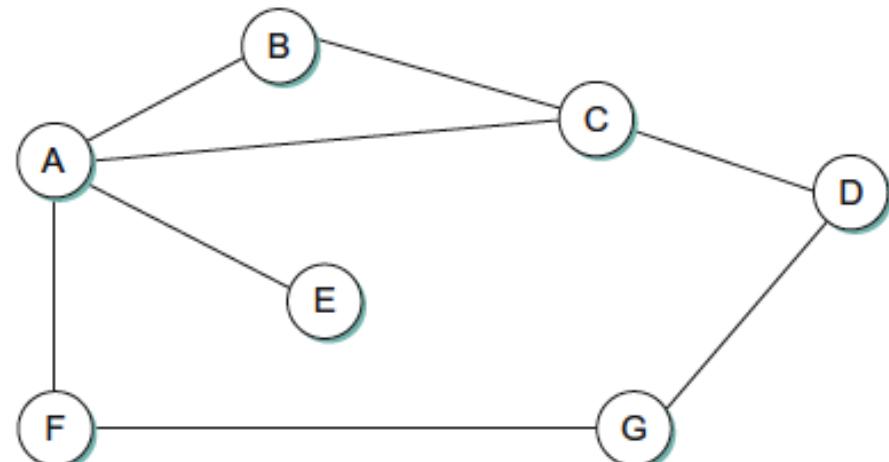
- B are ruta de distanță 2 către E prin A
- B nu include noua distanță către E în actualizarea trimisă lui A

**split horizon with poison reverse:** trimit o valoare f. mare

- B trimit distanța  $\infty$  către A
- A nu va mai alege o cale prin B

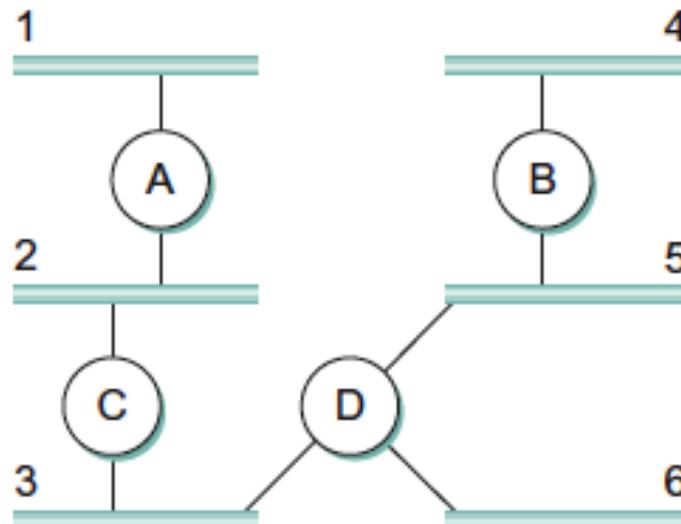
**Neajuns:** solutiile nu funcționează în toate cazurile

- de ex. pentru bucle cu mai multe noduri



# 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 distanteelor la fiecare 30 secunde
- distante maxime de 15 hop-uri (16 inseamna infinit)
  - rețele de mici dimensiuni

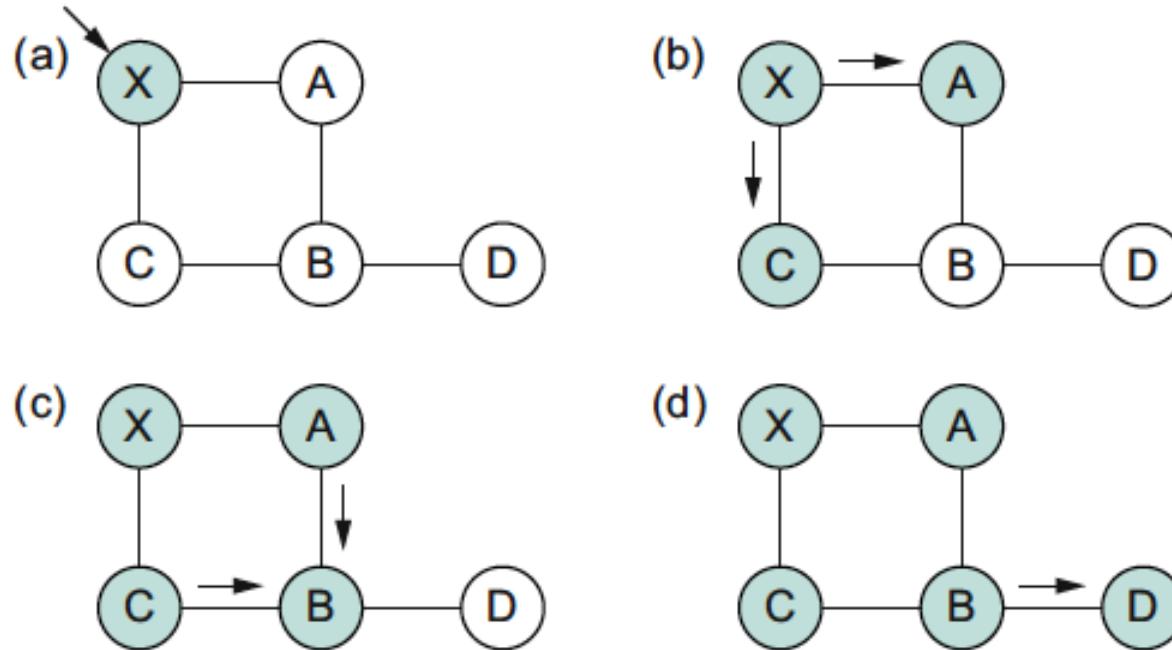




# Starea legaturii

- Presupune ca fiecare nod poate găsi **legaturile cu vecinii** și **costul** fiecărei legaturi
- Informatiile sunt **diseminate prin inundare** tuturor celorlalte noduri
  - LSP – Link State Packet transmis prin inundare;
  - Pachetul contine
    - Id-ul nodului care crează pachetul
    - lista **nodurilor** conectate cu **costul** fiecărei legaturi
    - un număr de secvență
    - durată de viață a pachetului (număr)
- Cu informațiile primite, **fiecare nod** va calcula **rutele cele mai scurte** către celelalte noduri

# Transmiterea prin inundare



- ID și nr. secvență:
  - nodul are o copie a pachetului sau pachetul este vechi - ignorat;
  - pachet nou - memorat și retransmis vecinilor (mai puțin vecinului de la care l-a primit)
- durata de viață:
  - număr decrementat la fiecare nod (hop) – asigura eliminare pachete vechi



# Algoritmul căii celei mai scurte

## Algoritmul lui Dijkstra

- nnod numărul nodurilor rețelei;
- sursa nodul sursă;
- $I[i][j]$  costul legăturii  $(i,j)$ , având valorile  
0 dacă  $i = j$ ;  
 $\text{lungmax}$  dacă  $i$  și  $j$  nu sunt adiacente;  
o valoare între 0 și  $\text{lungmax}$  în celealte cazuri;
- $D[i]$  costul minim al legăturii de la sursă la  $i$ ;
- $S$  mulțimea nodurilor deja selectate;
- $V$  tabloul de dirijare;  
 $V[i] = \text{vecinul prin care se transmit date de la nodul curent la nodul } i.$



```

void Dijkstra (int sursa)
{ int i, j, k;
  for (i=1; i <= nnod; i++)
  {   S[i] = 0;                                // nod neselectat
      D[i] = l[sursa][i];                      // distantele minime de la sursa
      if (D[i] < lungmax)
          V[i] = i;                            // initializeaza vecinii
      else
          V[i] = 0;
  }
  S[sursa] = 1;                                // selecteaza nodul sursa
  D[sursa] = 0;

  for ( i=1; i < nnod; i++)
  {   gaseste nodul k neselectat cu D[k] minim;
      S[k] = 1;
      for (j=1; j <= nnod; j++)              // recalculeaza distantele
          if ((S[j] == 0) && (D[k] + l[k][j] < D[j]))
              { D[j] = D[k] + l[k][j];
                V[j] = V[k];                  // modifica tabela de dirijare
              }
  }
}

```



# Structura ierarhica a Internet-ului

- Internet-ul este partajat în mai multe domenii numite sisteme autonome **AS – Autonomous Systems**
  - ASs sunt rețele independente operate de organizații diferite
- Într-un AS se folosește același algoritm de rutare “intra-domeniu”
  - denumit **interior gateway protocol**
  - ex. **OSPF – Open Shortest Path First**
- Rutarea între AS-uri (între domenii) folosește, de asemenea, un protocol comun
  - denumit **exterior gateway protocol**
  - ex. **BGP – Border Gateway Protocol** (bazat pe RIP)

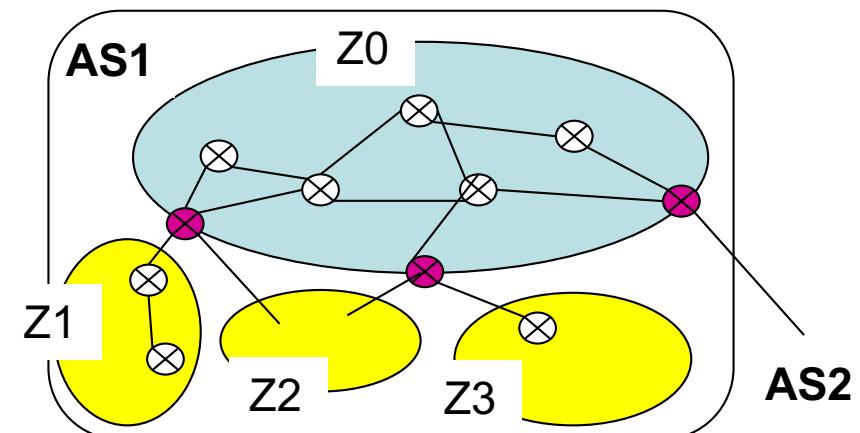
# Structura ierarhica AS

**Fiecare AS** este partitionat în mai multe **zone (areas)** – fiecare zonă reprezentând un **grup de rețele**

- zona **Z0** – coloana vertebrală (**backbone**)
- zone “**stub**” **Z1, Z2 ...** – legate la **backbone**
- **rutele** între noduri din **zone stub** diferențiate trec prin zona **Z0**

• Ierarhizarea crește **scalabilitatea**

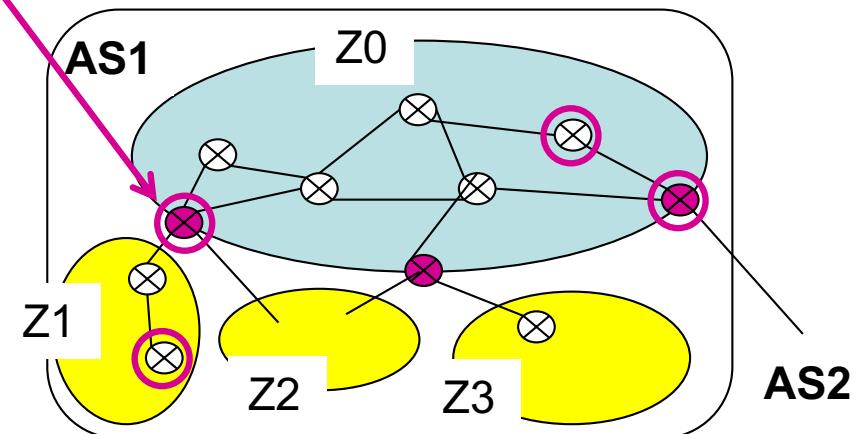
- un ruter dintr-un AS nu trebuie să stie cum se ajunge la fiecare **rețea** din AS, fiind suficient să stie cum se ajunge la **zona** în care se află rețeaua respectivă → reduce volumul tabelelor de dirijare



# Structura ierarhica AS (2)

Tipuri de rutere

- **interne** unei zone
- **de coloană vertebrală** (backbone)
- **de graniță zonală** (apartin zonei 0 si zonelor conectate)
  - nodul incercuit face parte din zonele Z0, Z1, Z2
- **de graniță AS**





# Mesaje OSPF

Mesajele OSPF permit schimbul de informații între noduri

Sunt transmise în **pachete IP** cu 89 ca număr de protocol

**Hello** – stabilește și pastrează legături cu vecinii

descoperă nodurile (ruter) cu care este conectat direct

**Link state request** - Cerere stare legătură

cere info despre anumite legături de la un alt ruter

**Link state update** - Actualizare stare legătură

trimit info despre legături, ca răspuns la o cerere

**Link state ack** - Confirmare stare legătură

confirmă primirea unui mesaj de actualizare

**Database description** – trimit LSBD – Link State Data Base

mesajele contin info despre AS sau zonă

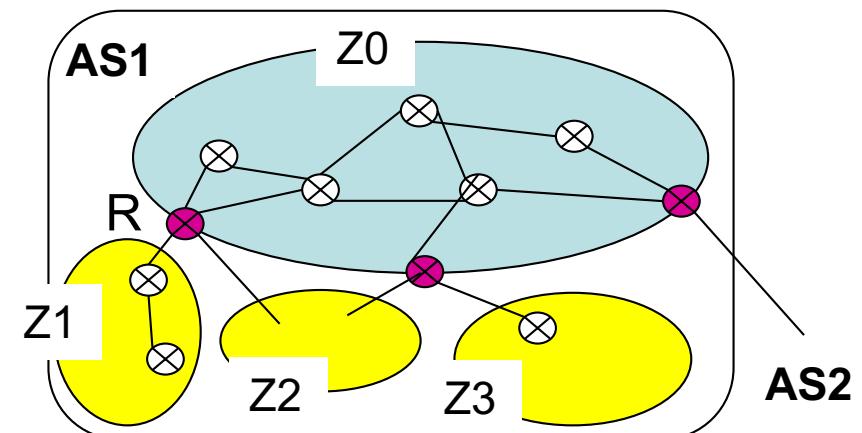
## Calcul rute - Nivel 1 (zona)

Folosind **inundarea**, fiecare ruter informeaza celelalte rutere din **zona** despre legaturile sale si costurile acestora

- ex. informatiile de starea legaturilor schimbată **intre noduri** din Z1 nu se transmit în afara acestei zone!

Fiecare ruter (**inclusiv cele de graniță zonală**) din **zonă** calculează separat căile cele mai scurte către ruterele din aceeași zona

- în final, un **ruterul de graniță zonală R** va cunoaște căile cele mai scurte către oricare rețea din Z1 și Z2



# Calcul rute - Nivel 2 (AS)

**Ideea:** calea unui pachet între două zone diferite are trei parti:

- de la nodul sursă la zona backbone
- traversează backbone
- de la backbone la rețeaua de destinație

Ruterele de **coloana vertebrală (backbone)**

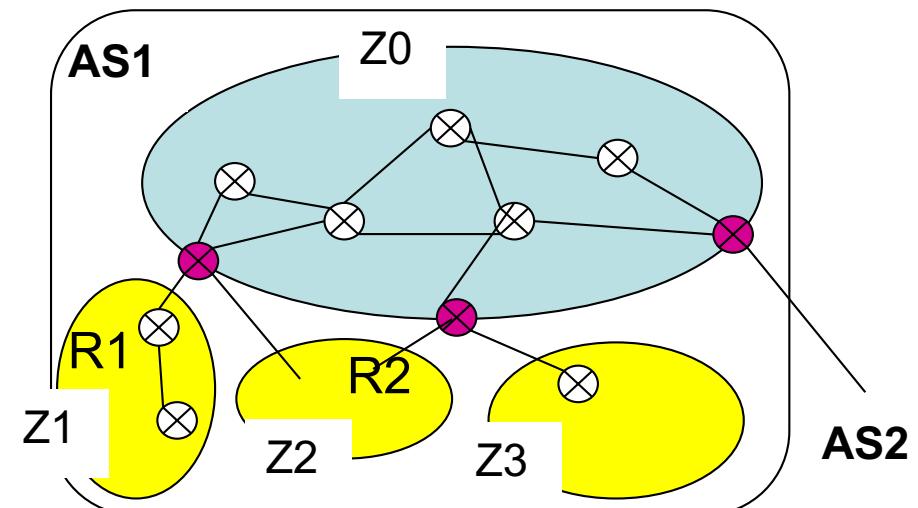
primesc informații de la ruterele de granită zonale și calculează cele mai bune rute la rețele din orice zona

ex. R1 oferă căile cele mai scurte pentru Z1 și Z2

ruterele din Z0 calculează căile către orice rețea din Z1, Z2

Rezultatele sunt **difuzate de la Z0** inapoi la **zonele stub**, care actualizează căile cele mai scurte la rețele din alte zone

**Ex.** ruterele din Z2 vor să știe să aleagă între R1 și R2 pentru rutare spre rețele din alte zone



# BGP – Border Gateway Protocol

Algoritmi orientați pe aspectele politice, de securitate, economice

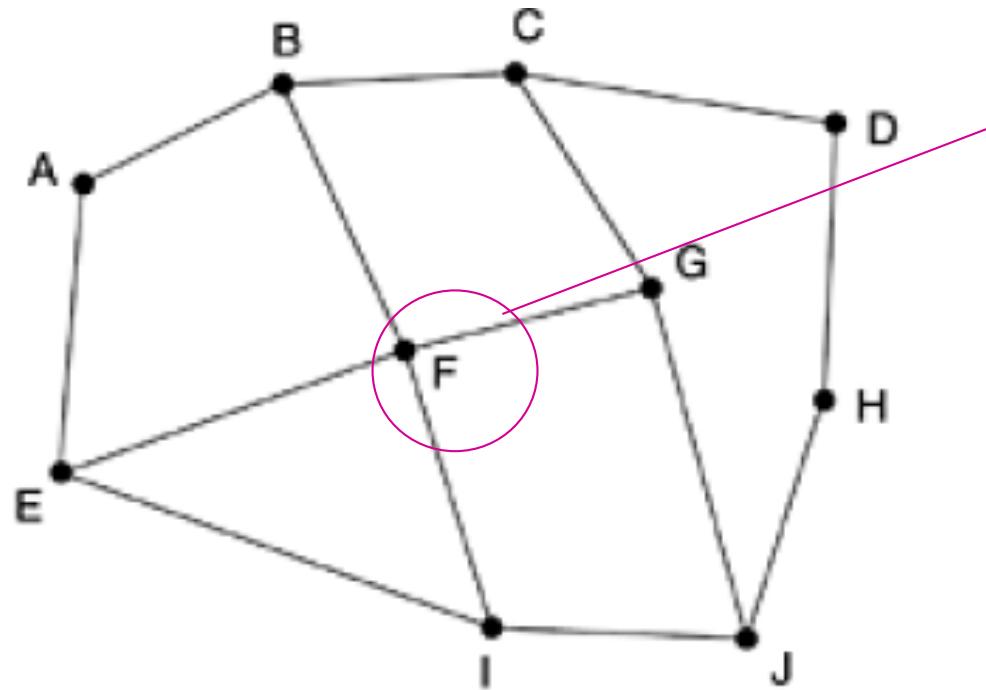
**Rețea = ASes și conexiunile**

**Protocol = vectorul distanțelor**

**Tabelele de dirijare conțin și rutele spre destinație**

**Comunică vecinilor căile utilizate efectiv (ocoleste numaratoarea la infinit)**

**Ex. pp. F folosește calea FGCD la D**



**G se defectează**  
**Informatiile primite de F de la vecinii ramasi, despre D:**

**De la B: "Eu folosesc BCD"**  
**De la I: "Eu folosesc IFGCD"**  
**De la E: "Eu folosesc EFGCD"**

**F elimina caile care contin F si alege calea FBCD**



# Dirijarea în rețele ad hoc

**AODV – Ad hoc On demand Distance Vector - Determină ruta la cerere**

**rețea ad hoc = graf**

**Muchie = conexiune – nodurile pot comunica direct (radio)**

**Fiecare nod = ruter + gazdă**

**Conține**

**Tabela dirijare**

**destinație,**

**pas următor,**

**distanță**

**nr secv destinație**

**altele**

**Tabela history**

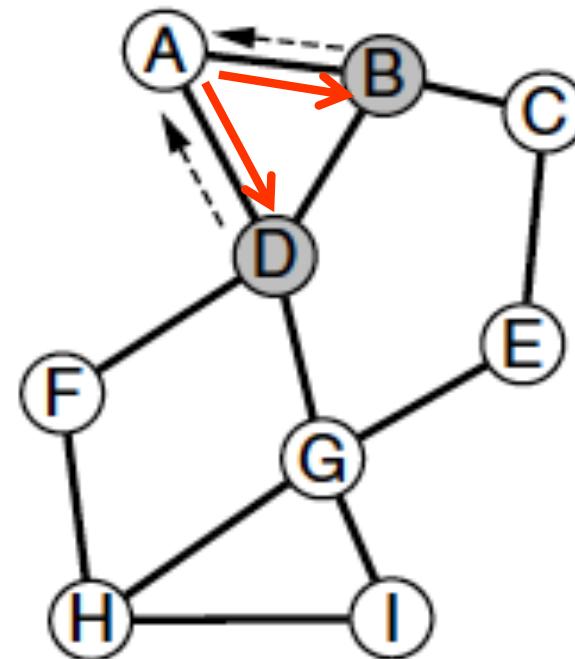
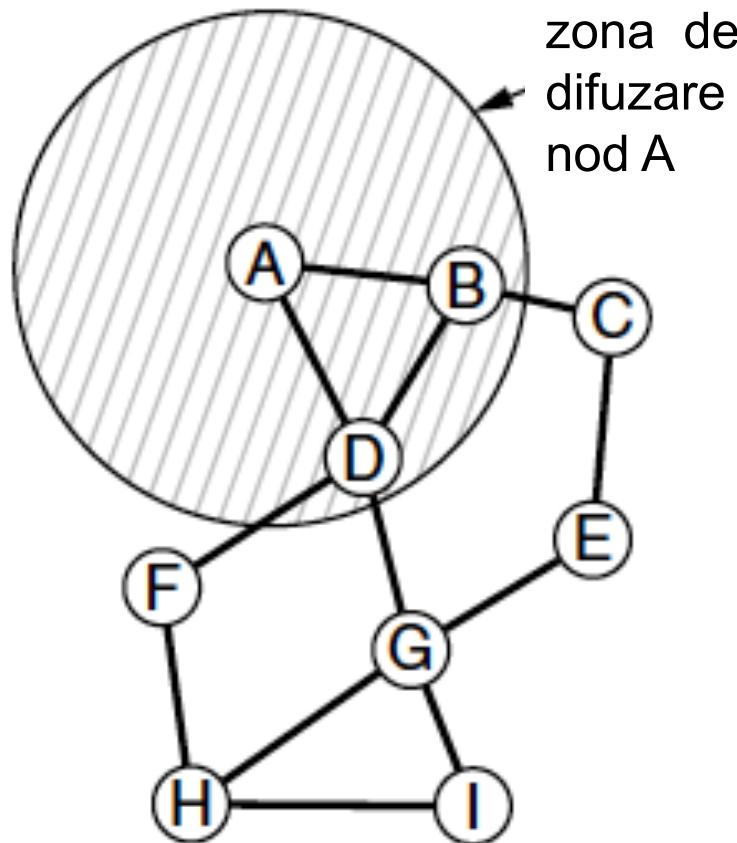
**identitatile cererilor precedente**

**Tabela reverse route**

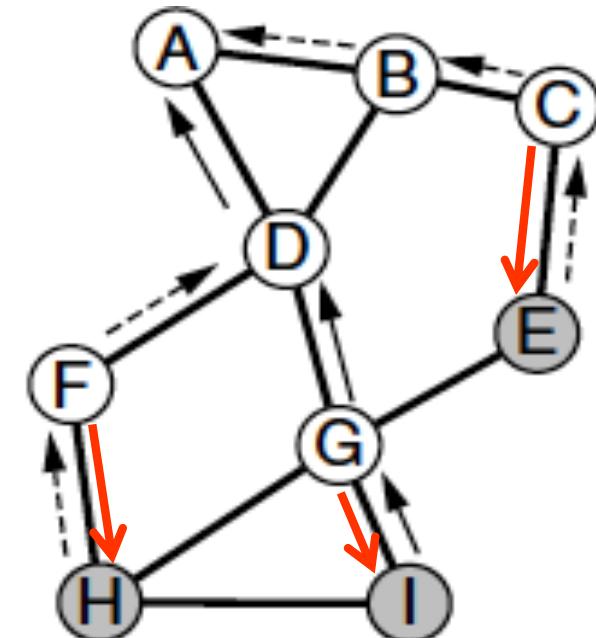
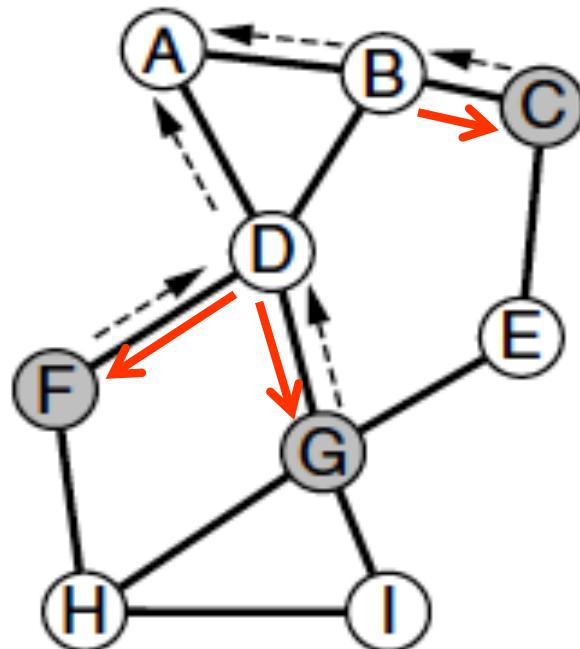
**calea spre sursa unui pachet de cerere**

# Cum functioneaza?

**Exemplu:** A vrea să comunice cu I care nu e în tabela sa  
trebuie să descopere ruta  
trimite (broadcast) o **cerere de ruta**, care ajunge la nodurile B și D



- daca B si D nu cunosc ruta spre I, cererea este re-transmisa, prin **inundare**, celorlalte noduri
  - mai intai C, F si G
  - apoi E, H si I
- daca cererea ajunge la I, acesta va returna un **raspuns** catre A
  - raspunsul parcurge **calea inversă** prin G si D la A (sageti negre pline)
  - fiecare nod trimite raspunsul vecinului de la care a primit cererea (sageti punctate)



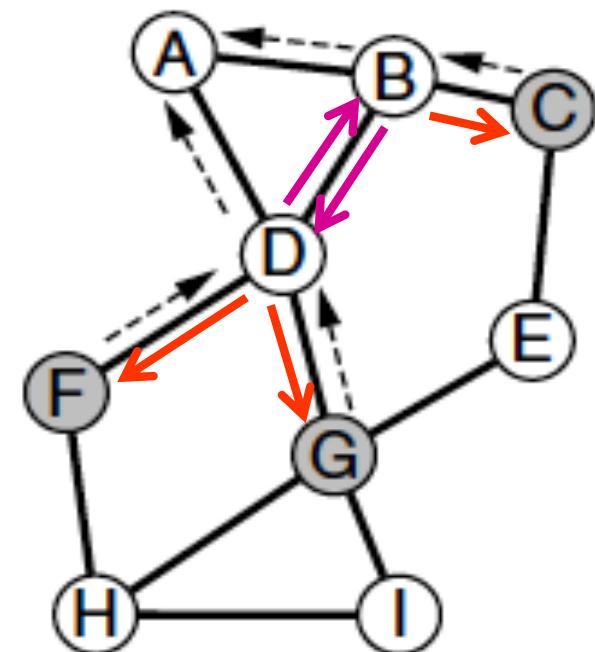
# Probleme

## Eliminarea replicilor

- pachete transmise prin inundare
- B primește de la D o copie a pachetului
- D primește de la B o copie a pachetului
- pentru a descoperi și elmina copiile
  - pachetul contine un **ID al cererii**
  - nodurile pastrează cererile primite în **tabela history**

## Lungimea caii A → I

- pachetul de cerere are un camp **Hop count**
- initializat de A la 0
- incrementat de fiecare nod care retransmite pachetul
- returnat de I în răspunsul lui



# Probleme-2

Nodul A poate avea o **ruta veche**

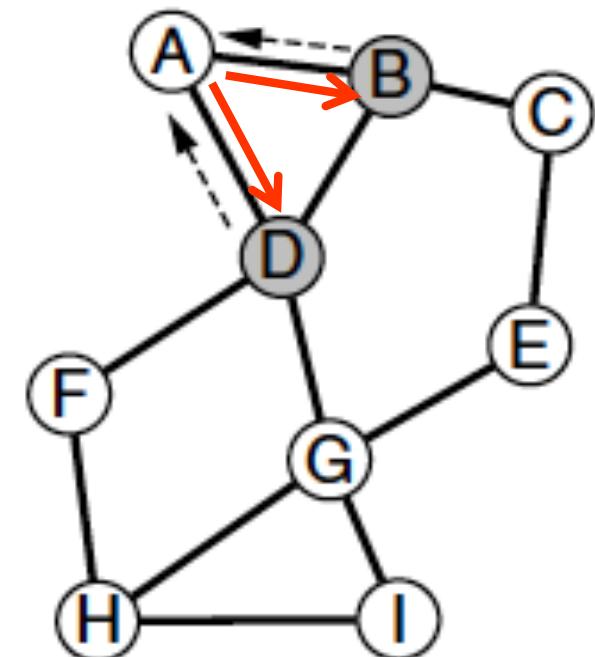
- trimit o cerere pentru actualizarea ei

Se folosește **Sequence #** pentru a deosebi rutele noi de cele vechi

- inclus de A în pachet cerere
- asociat de fiecare nod rutei curente din tabela de dirijare

Exemplu

- dacă nodul D are o **ruta mai nouă** decât cea din cerere, nu va mai re-difuză cererea și va trimite răspunsul pe calea inversă





# Informatii pastrate de noduri

## Tabela [dirijare](#)

destinație,  
pas următor,  
distanță,  
nr secv destinație  
altele

## Tabela [history](#)

identitatile cererilor precedente

## Tabela [reverse route](#)

calea spre sursa unui pachet de cerere



# Pachete ROUTE REQUEST

A difuzează un pachet ROUTE REQUEST

Identificat unic prin **Source address + Request ID**

Folosește **Sequence #** pentru a deosebi rutele noi de cele vechi

Prelucrarea **ROUTE REQUEST** în fiecare nod

Verifica dupăt în tabela **history** locală (Source address + Request ID)

Transmite **ROUTE REPLY** dacă găsit ruta nouă, adică

**Dest sequence #** în routing table > **Dest sequence #** în packet

Altfel,

incrementează **Hop count** și re-difuzează ROUTE REQUEST

memorează informația în **reverse route table**

**Source sequence #** folosit pentru actualizare tabela dirijare locală

Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count



# Pachete ROUTE REPLY

I construieşte ROUTE REPLY şi-l trimit pe legătura inversă

Source address, Destination address sunt copiate

Hop count pus pe zero

Destination sequence # luat din contorul propriu

Lifetime = cât timp rămâne valid

Prelucrarea la alte noduri

Actualizează tabela dirijare locală

Transmite pe legătura inversă

Trece prin anumite noduri – celelalte şterg intrarea în reverse route table

Source address	Destination address	Destination sequence #	Hop count	Lifetime
----------------	---------------------	------------------------	-----------	----------

# Intreținerea rutelor

Actualizare: (1) la cerere; (2) la defectări

G cade → D descoperă (se folosesc mesaje Hello periodice)

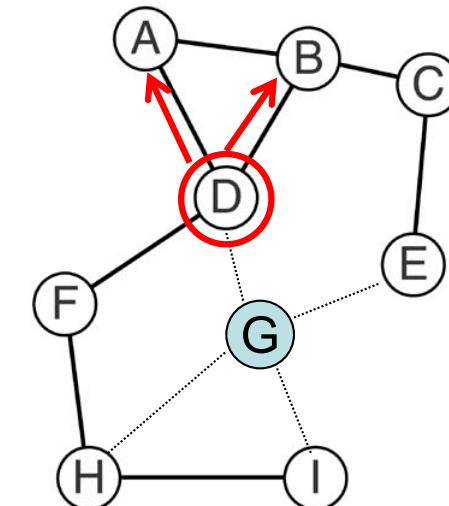
D află că G a fost utilizat pe rute către E, G și I

D anunță vecinii activi (active neighbors) care folosesc G, anume {A, B}

D golește intrările pentru E, G și I din tabela de rutare

Dest.	Next hop	Distance	Active neighbors	Other fields
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E				
F	F	1	A, B	
G				
H	F	2	A, B	
I				

(a)



(b)



# IPv6 - Motivații

## Spațiul de adrese

32 biți = peste un milion de rețele

Dar...multe sunt Clasa C, prea mici pentru multe organizații

## Tip servicii

Aplicații diferite au cerințe diferite de livrare, siguranță și viteză

IPv4 are **tip de serviciu** dar adesea nu este implementat



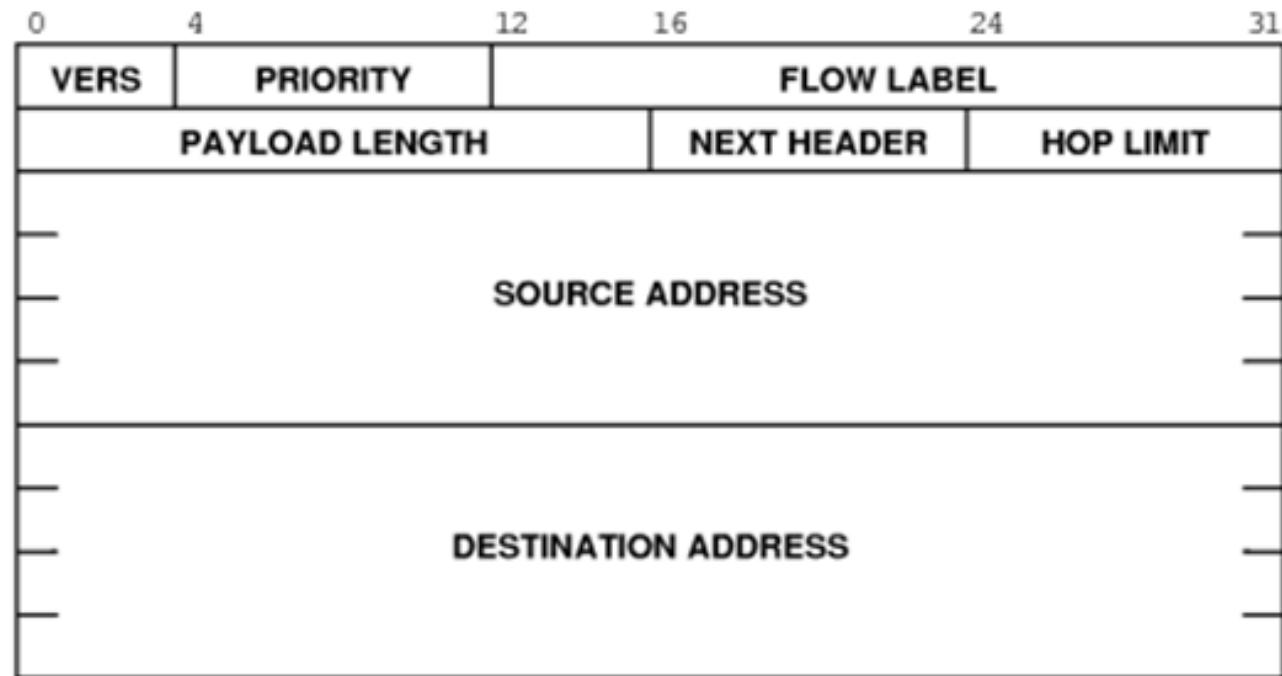
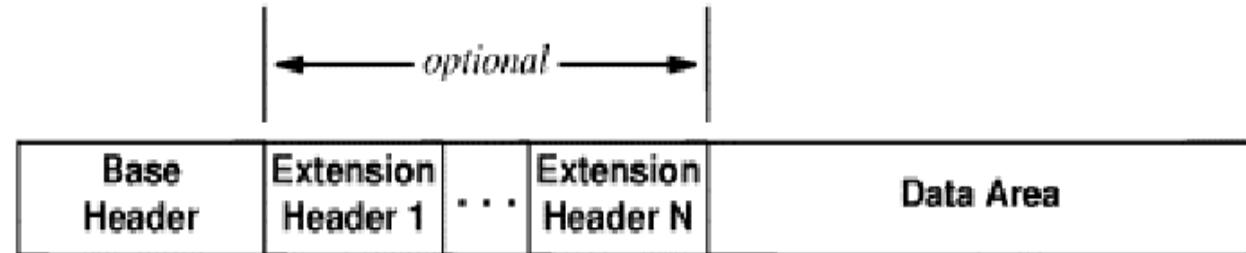
# IPv6 - format Base header

lungime fixă = 40 octeti

Priority - clasa de trafic

**FLOW LABEL** - asociază datagramele unui flux

Diferențe circuit virtual  
două fluxuri cu aceeași etichetă se diferențiază prin adr sursă + adr dest  
aceeași pereche sursă+dest poate avea mai multe fluxuri





## Conține mai puține info decât antet IPv4

Restul de info în extensii

NEXT HEADER definește tipul datelor (ex. TCP)

NEXT HEADER definește tipul antetului de extensie urmator (ex. Route Header)



(a)



(b)



## IPv6 – antete extensie

Hop-by-hop header – info pentru rutere – deocamdata:

suport datagrame excedând 64K (jumbograme)

specifica lungimea;

campul de lungime din antetul de baza este 0

Destination header – info aditionale pentru destinație

nefolosit

Routing – lista rutere de vizitat

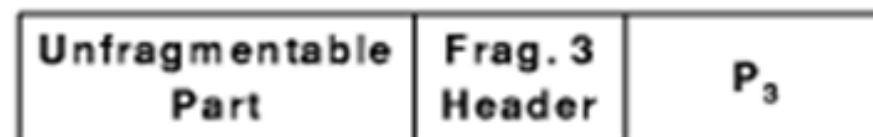
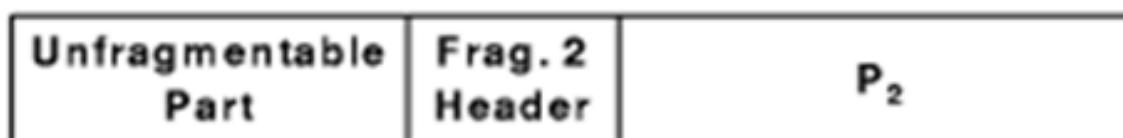
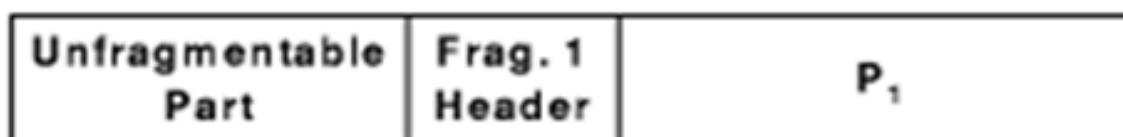
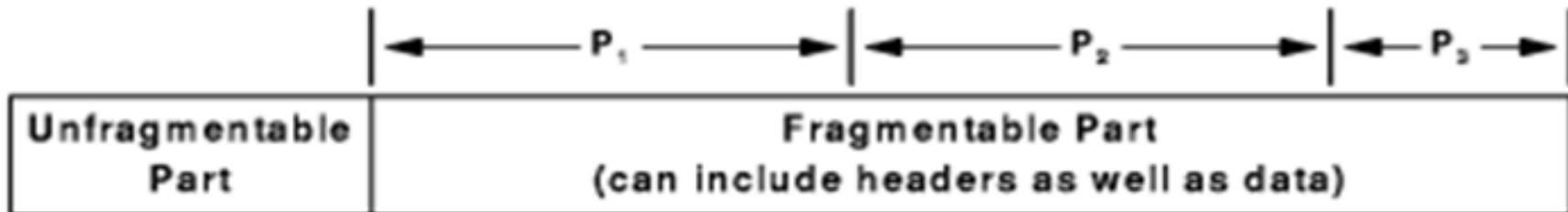
Fragmentation – identificare fragmente

Authentication – verificare identitate transmițător

Encrypted security payload – info despre conținut criptat



# Fragmentarea





## Fragmentare IPv6 – la sursă

Ruterele ignoră datagramele mai lungi decât MTU

### Sursa

Fragmentează pachetele

Descoperă path MTU

### Caracter dinamic

- calea se poate schimba

Eficiență – antet nu are spațiu pierdut

Flexibilitate – noi antete pentru noi caracteristici

Dezvoltare incrementală – ruterele care tratează anumite antete coexistă cu altele care le ignoră



## adrese 128-bit

Includ prefix rețea și suffix gazdă

Fără clase de adresă – limita prefix/suffix oriunde

Tipuri speciale de adrese:

- unicast
- multicast
- cluster – colecție de calculatoare cu același prefix; datagrama livrată unei din ele (permite duplicare servicii)



# Notația adresei

16 numere

105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255

Notație hexazecimală

69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF

Compresie zerouri

FF0C:0:0:0:0:0:B1

FF0C::B1

adrese IPv6 cu 96 zerouri prefix sunt interpretate ca adrese IPv4



# Studiu individual

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

5.1 CERINȚELE DE PROIECTARE ALE NIVELULUI REȚEA

5.2.2 Dirijarea pe calea cea mai scurtă

5.2.3 Inundarea

5.2.4 Dirijare cu vectori distanță

5.2.10 Dirijarea în rețele AD HOC

5.6.1 Protocolul IP

5.6.2 Adrese IP

5.6.4 Protocole de control în Internet

5.5.5 Protocolul de dirijare folosit de porțile interioare: OSPF

5.6.5 Protocolul de dirijare pentru porți externe: BGP

5.6.8 IPv6



# Studiu individual

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

5.1 NETWORK LAYER DESIGN ISSUES

5.2.2 Shortest Path Algorithm

5.2.3 Flooding

5.2.4 Distance Vector Routing

5.2.11 Routing in Ad Hoc Networks

5.6.1 The IP Version 4 Protocol

5.6.2 IP Addresses

5.6.3 IP Version 6

5.6.4 Internet Control Protocols

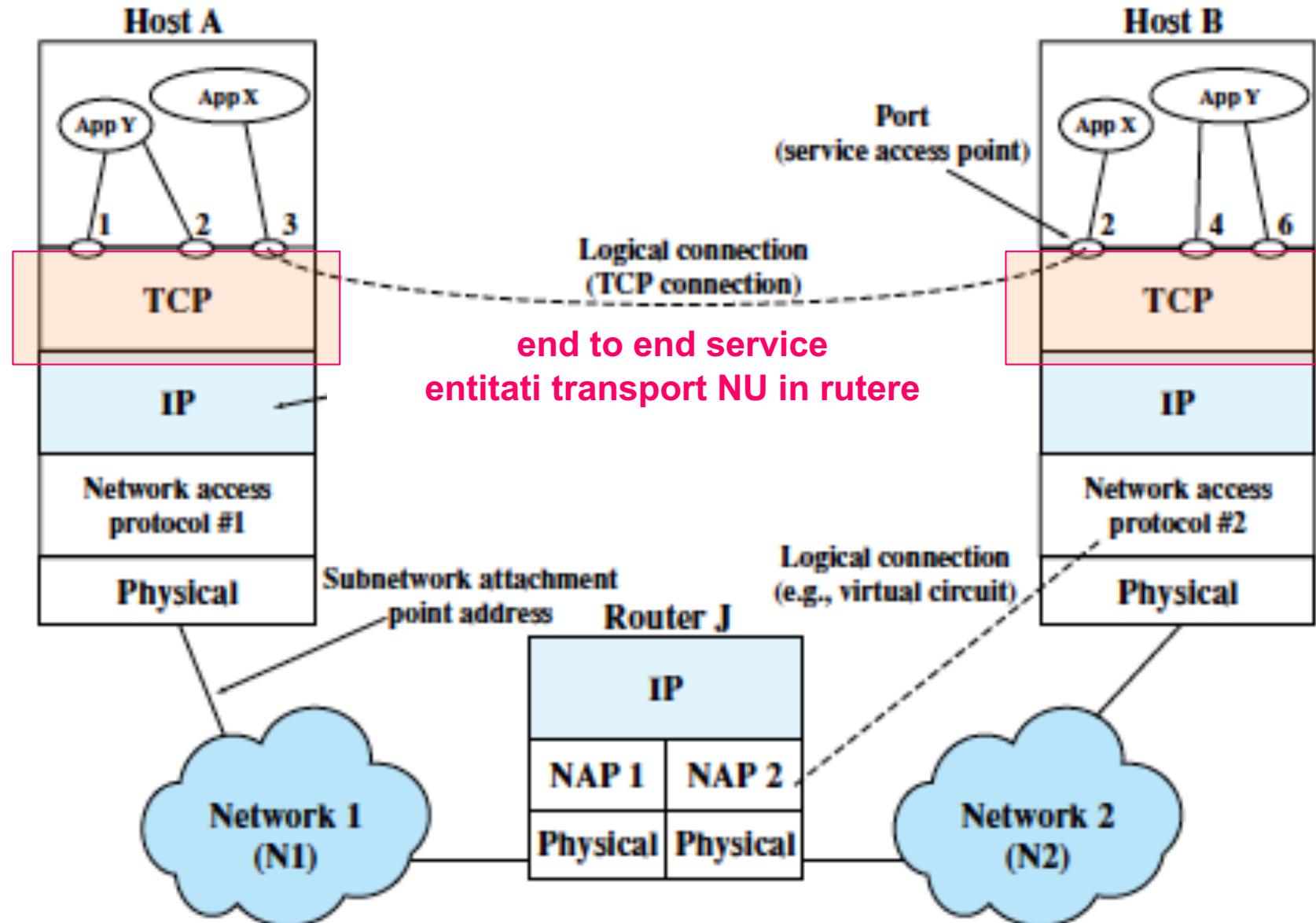
5.6.6 OSPF—An Interior Gateway Routing Protocol

5.6.7 BGP—The Exterior Gateway Routing Protocol



# Nivelul transport

# Nivelul transport in ierarhia TCP/IP

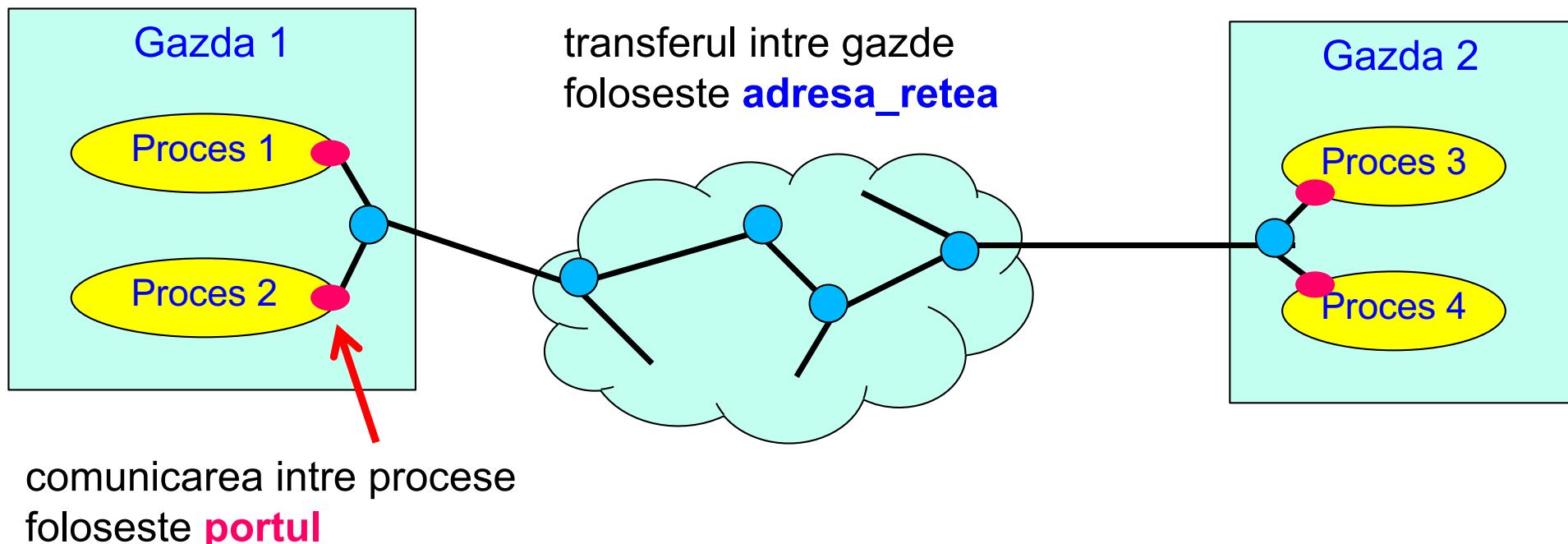


# Comunicarea intre aplicatii

Reteaua asigura transmiterea **pachetelor** (datagramme) intre **calculatoare** gazda

Transportul asigura comunicarea intre aplicatii

- legatura proceselor de aplicatie cu **punctele de acces la retea**
- **identificarea unica a unui punct de acces prin <adresa\_retea, port>**





# Servicii ale nivelului Transport

- **Servicii furnizate**
  - transfer de date intre procese de aplicatie, folosind retele de diverse tipuri
  - interfața uniforma cu utilizatorii
- **Caracteristici**
  - două tipuri de servicii:
    - orientate pe conexiune (connection oriented) - TCP
    - fără conexiune (connectionless) - UDP



# Socket interface

- Serviciile nivelului transport sunt accesibile ca **API** – Application Programming Interface
- Oferită ca bibliotecă utilizator sau funcții OS
  - API descrie cum se apelează aceste funcții
- **Socket API**
  - Originară din Berkeley BSD UNIX
  - Disponibilă și pe Windows, Solaris, etc.
  - **socket** = punctul în care procesul de aplicație se atașază la rețea
  - identificat prin descriptor - **număr** (ca la fișiere)
- Nu e standard de jure ci *standard de facto*



# Creare socket

`int socket(int family, int type, int protocol);`

`socket_descr = socket (protocol_family, comm_type, protocol)`

- deschide un socket
- intoarce `socket_descriptor` folosit în apelurile următoare

`protocol_family` selectează familia de protocoale

`PF_INET` - protocoale `Internet`

`PF_APPLETALK` - protocoale AppleTalk etc.

`comm_type` selectează tipul de comunicare

`SOCK_DGRAM` - fără conexiune - datagramă

`SOCK_STREAM` - orientat pe conexiune – flux de octetii

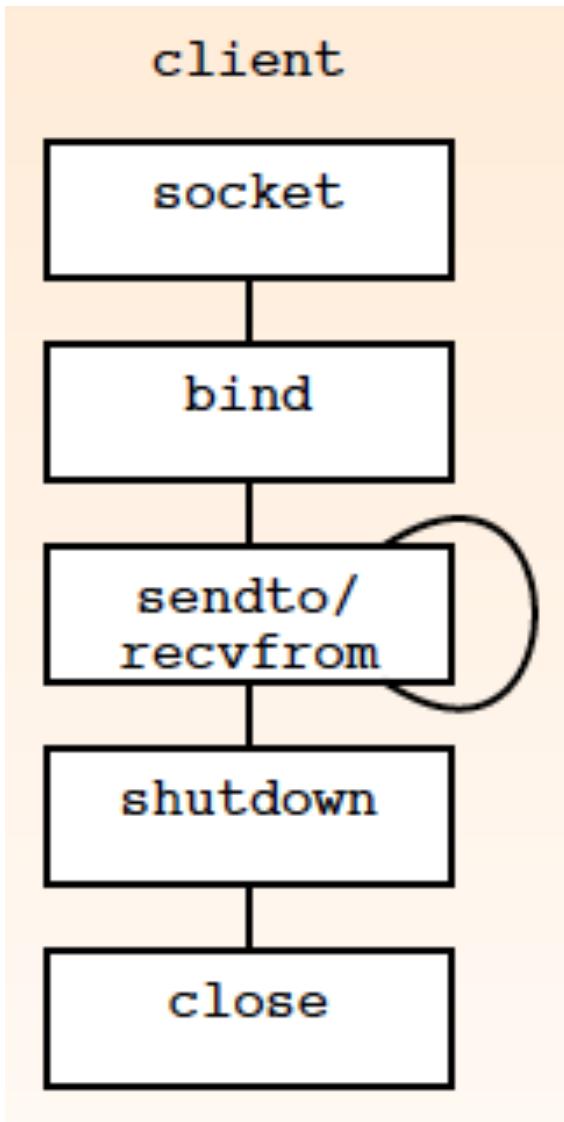
`protocol` specifică protocolul

`IPPROTO_TCP` - TCP

`IPPROTO_UDP` - UDP



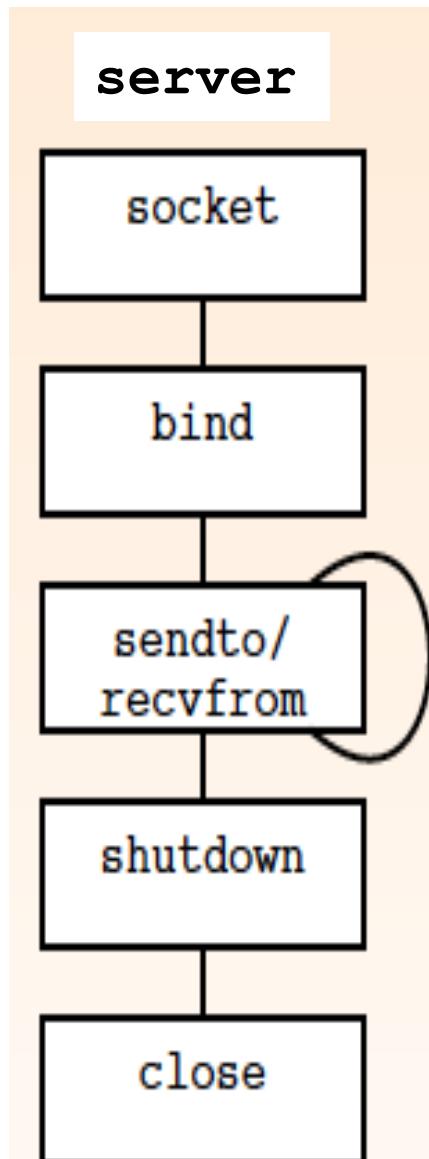
# Serviciu fără conexiune



**server** este la fel!

bind - optional pentru initiator

# 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 receptie 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
```

# Exemplu client

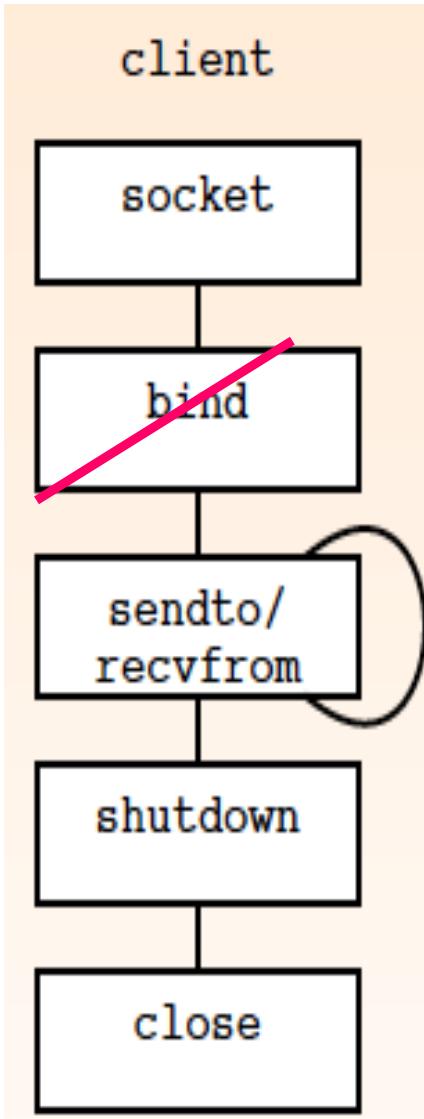
```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4 #include <netdb.h>
5
6 #define SERVER PORT 12345 /* arbitrary, but client & server must agree */
7 #define BUF SIZE 4096 /* block transfer size */
8
9 int main(int argc, char **argv) {
10
11     int c, s, bytes;
12     char buf[BUF SIZE]; /* buffer for incoming file */
13     struct hostent *h; /* info about server */
14     struct sockaddr in channel; /* holds IP address */
15
16     if (argc != 3) fatal("Usage: client server-name file-name");
17     h = gethostbyname(argv[1]); /* look up host's IP address */
18     if (!h) fatal("gethostbyname failed");
19
20     s = socket(PF INET, SOCK STREAM, IPPROTO TCP);
21     if (s < 0) fatal("socket");
22
23     memset(&channel, 0, sizeof(channel));
24     channel.sin family= AF INET;
25     memcpy(&channel.sin addr.s addr, h->h addr, h->h length);
26     channel.sin port= htons(SERVER PORT);
27     c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
28     if (c < 0) fatal("connect failed");
29
30     /* Connection is now established. Send file name including 0 byte at end. */
31     write(s, argv[2], strlen(argv[2])+1);
32     /* Go get the file and write it to standard output. */
33     while (1) {
34         bytes = read(s, buf, BUF SIZE); /* read from socket */
35         if (bytes <= 0) exit(0); /* check for end of file */
36         write(1, buf, bytes); /* write to standard output */
37     }
38 }
39
40 fatal(char *string) {
41     printf("%s\n", string);
42     exit(1);
43 }
```



# Exemplu server

```
2 #include <sys/fcntl.h>
3 #include <sys/socket.h>
4 #include <netinet/in.h>
5 #include <netdb.h>
6
7 #define SERVER PORT 12345 /* arbitrary, but client & server must agree */
8 #define BUF SIZE 4096 /* block transfer size */
9 #define QUEUE SIZE 10
10
11 int main(int argc, char *argv[]) {
12
13     int s, b, l, fd, sa, bytes, on = 1;
14     char buf[BUF SIZE]; /* buffer for outgoing file */
15     struct sockaddr in channel; /* holds IP address */
16
17     /* Build address structure to bind to socket. */
18     memset(&channel, 0, sizeof(channel)); /* zero channel */
19     channel.sin family = AF INET;
20     channel.sin addr.s addr = htonl(INADDR ANY);
21     channel.sin port = htons(SERVER PORT);
22     /* Passive open. Wait for connection. */
23     s = socket(AF INET, SOCK STREAM, IPPROTO TCP); /* create socket */
24     if (s < 0) fatal("socket failed");
25
26     setsockopt(s, SOL SOCKET, SO REUSEADDR, (char *) &on, sizeof(on));
27     b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
28     if (b < 0) fatal("bind failed");
29
30     l = listen(s, QUEUE SIZE); /* specify queue size */
31     if (l < 0) fatal("listen failed");
32
33     /* Socket is now set up and bound. Wait for connection and process it. */
34     while (1) {
35         sa = accept(s, 0, 0); /* block for connection request */
36         if (sa < 0) fatal("accept failed");
37
38         read(sa, buf, BUF SIZE); /* read file name from socket */
39         /* Get and return the file. */
40         fd = open(buf, O RDONLY); /* open the file to be sent back */
41         if (fd < 0) fatal("open failed");
42         while (1) {
43             bytes = read(fd, buf, BUF SIZE); /* read from file */
44             if (bytes <= 0) break; /* check for end of file */
45             write(sa, buf, bytes); /* write bytes to socket */
46         }
47         close(fd); /* close file */
48         close(sa); /* close connection */
49     }
50 }
```

# Serviciu fara conexiune - client



Creaza socket și aloca-i resurse de sistem:

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

Trimit 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));`

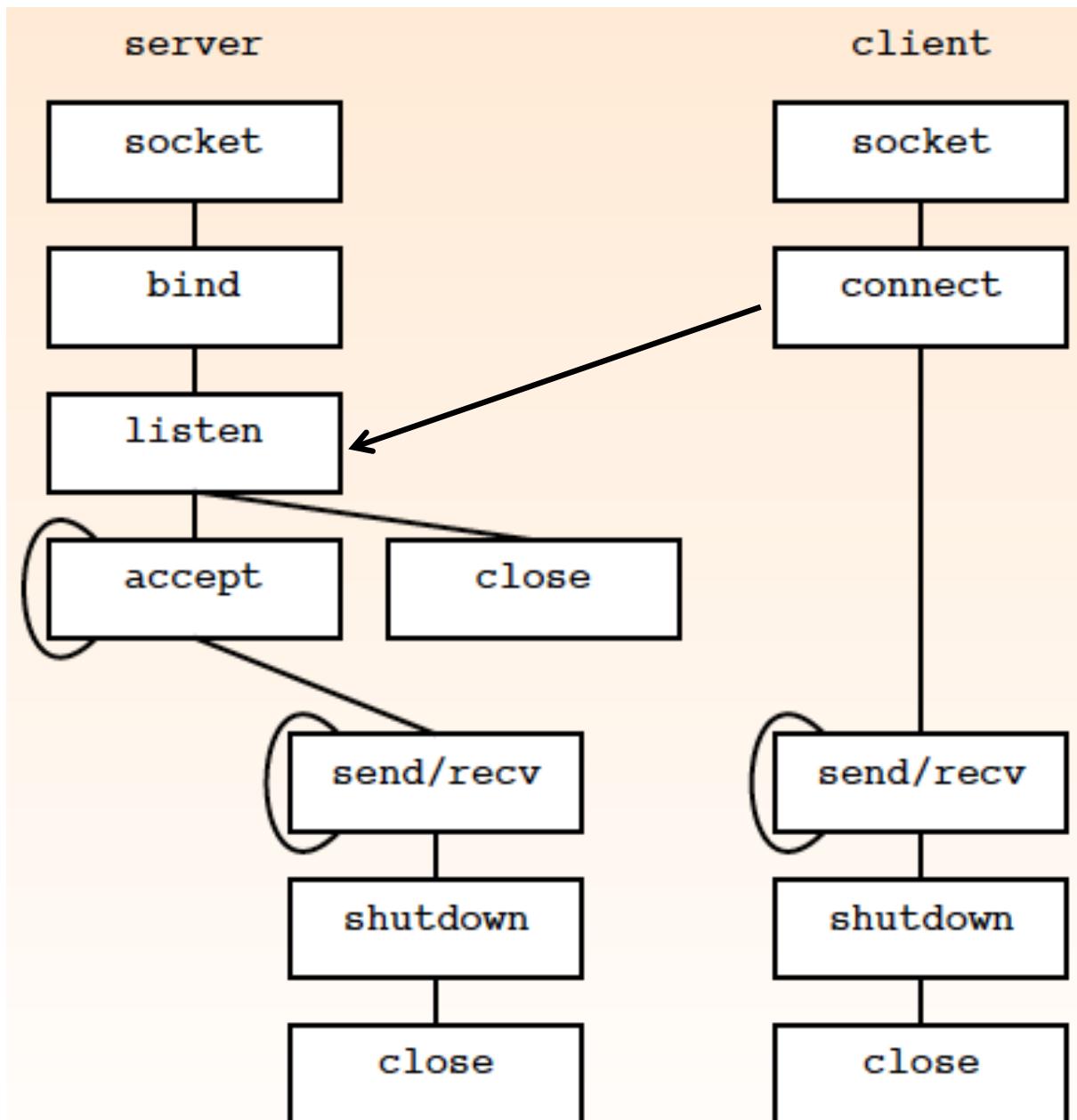
Oreste trimitere sau/și receptie de date

`shutdown (s, SHUT_RD / SHUT_RDWR / SHUT_WR)`

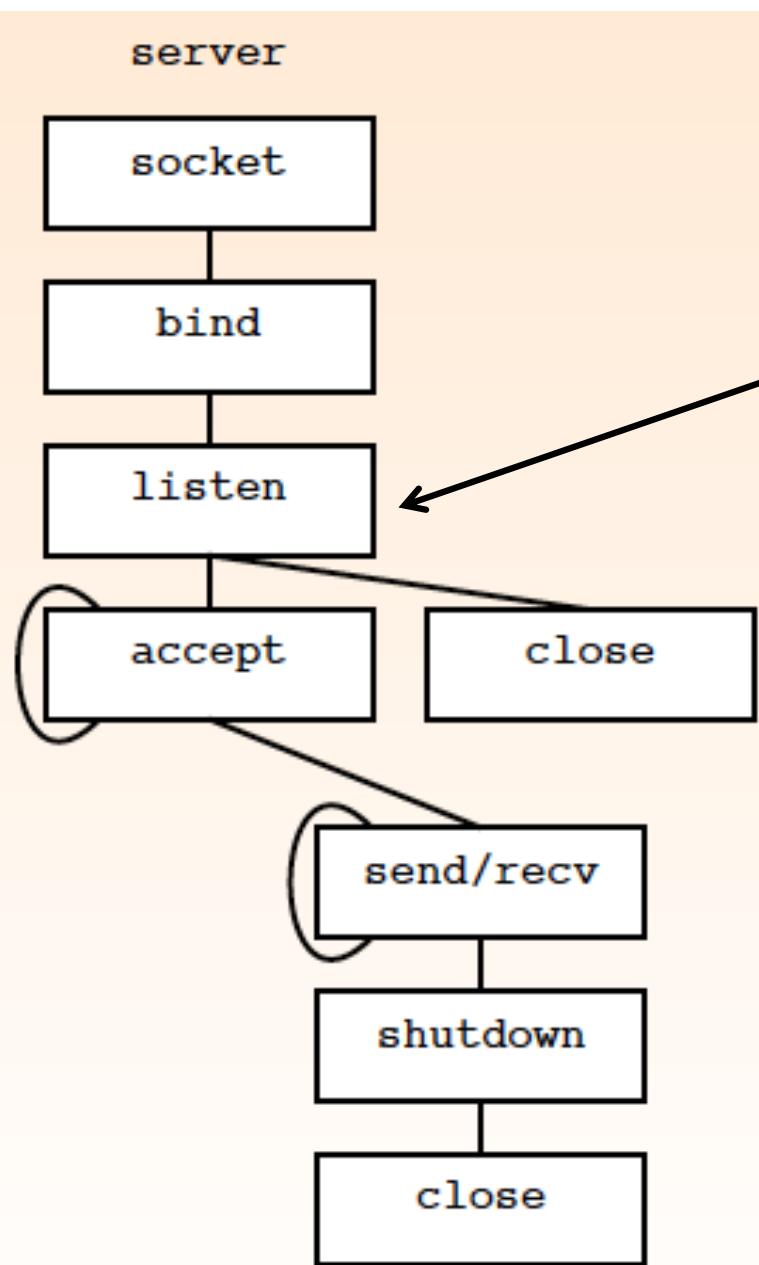
Termina utilizarea socket si elibereaza resursele alocate

`close (s)`

# Serviciu orientat pe conexiune



# Serviciu orientat pe conexiune - server



1. Creaza socket:

```
int ls = socket (AF_INET, SOCK_STREAM,  
IPPROTO_TCP);
```

2. Asociaza un socket cu <port, adresa\_IP>:

```
bind(ls, &addr, sizeof(addr));
```

3. Asteapta cereri de conectare la socket de ascultare (declara nr max cereri asteapta):

```
int listen (int socket_descriptor, int queue_size)
```

```
listen(ls, 5);
```

4. repetat - Accepta o cerere de conectare de la un client; creaza un nou socket pentru conexiune:

```
int accept (int listen_socket_descriptor, struct sockaddr*  
client_socket_addr, int* client_addrlen)
```

```
int s = accept(ls,NULL,NULL);
```

5. repetat - Trimit / primește pe s etc.



# Serviciu orientat pe conexiune - client

1. Creaza socket:

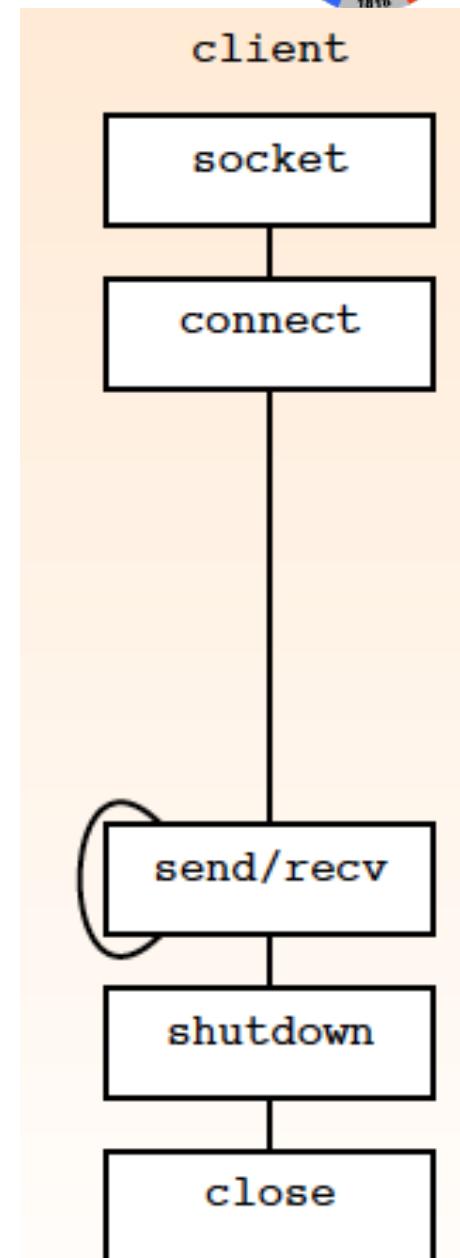
```
int s = socket (AF_INET, SOCK_STREAM,  
IPPROTO_TCP);
```

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 / primește 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 – trimitre/primire 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)



# Protocole de Transport

Două protocole majore

- UDP
- TCP

Aspecte discutate

- Formatul datelor
- Adresare
- Funcționare



# UDP - User Datagram Protocol

- UDP livrează datagrame utilizator - *user datagrams*
  - Livrare “Best effort” – datagramele pot fi pierdute, primite în alta ordine etc.
  - Sume de control pentru integritate
- Puncte de capăt UDP = *protocol ports* sau *ports*
- UDP identifică adresa Internet și *număr port* pentru sursă și destinație
- *Destination port* și *source port* pot差别.



# Antet UDP



**checksum** (calculat la fel ca la TCP) dar nefolosit

Nu control flux

Nu control erori

Nu retransmisie



# Aplicații care folosesc UDP

- DNS
- Voice over IP
- Online Games
- Se foloseste atunci cand:
  - Latența este foarte importantă
  - Livrarea tuturor datelor nu e necesara
  - Retransmisiile sunt implementate de aplicații 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

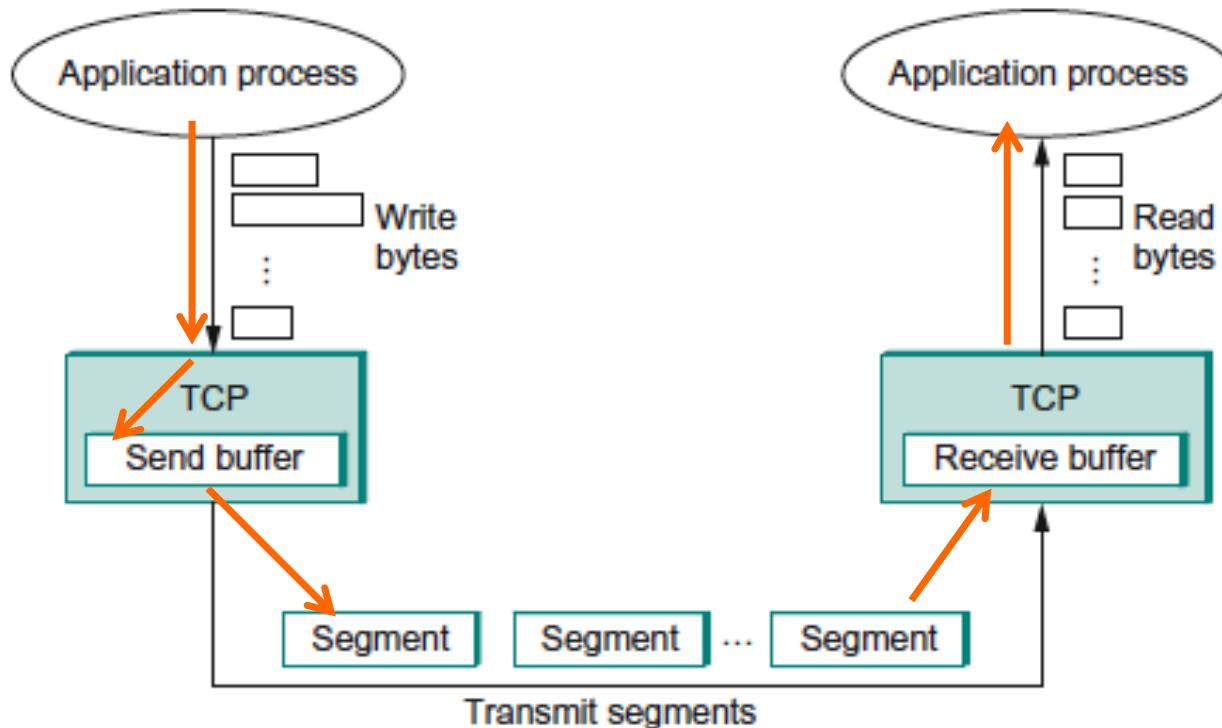
Peer-to-peer



# Cateva porturi standard

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

# TCP este orientat pe flux de octeti



- 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 (cât vrea!) din conexiunea TCP

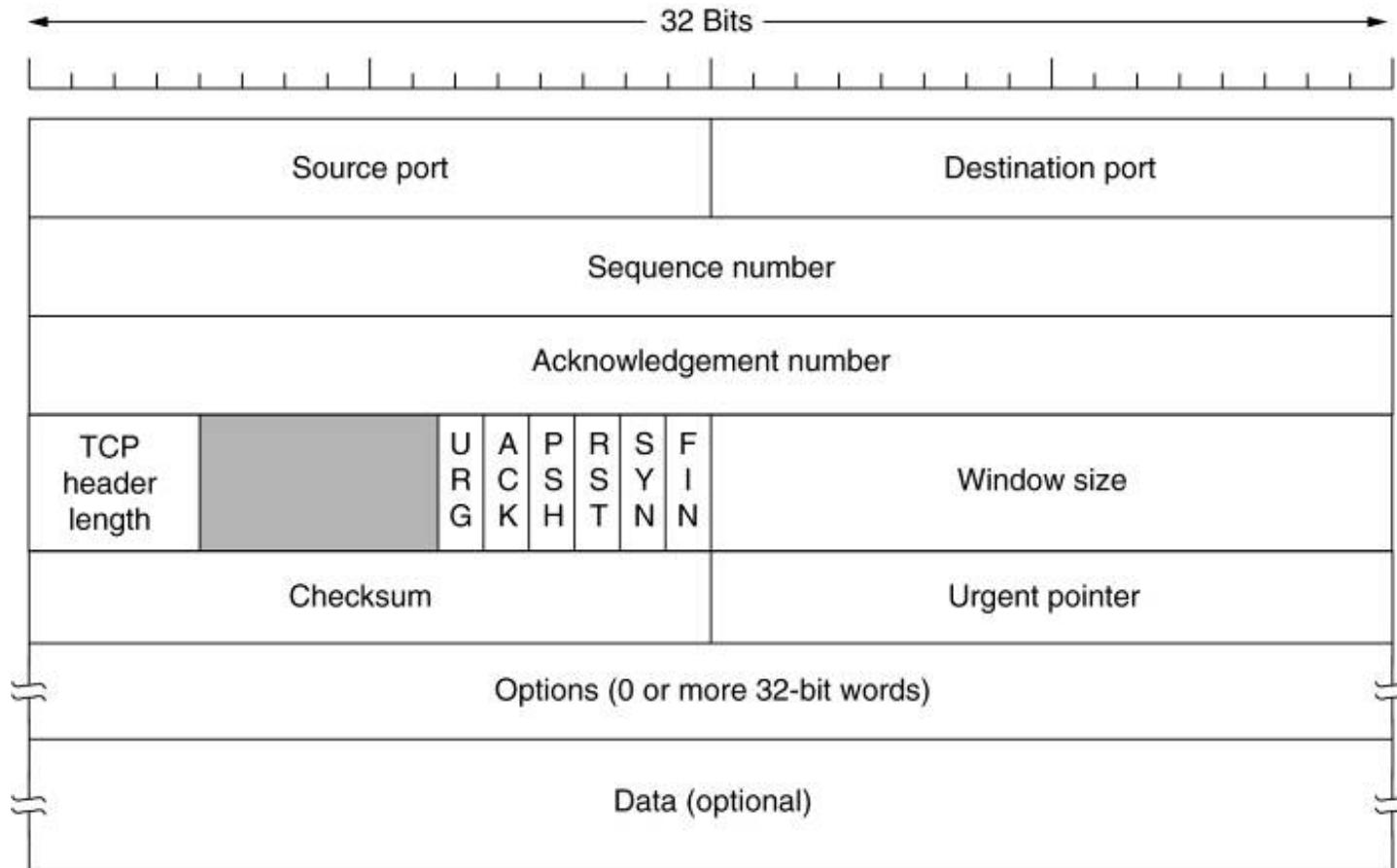


# Caracteristici

- Orientat pe conexiune
- Interfață **flux** (Stream)
  - transmisie și receptie siruri de octeti
- Face **controlul congestiei** adaptând viteza de transmisie la condițiile rețelei
- Garantează **transmisie în ordine și sigură** a datelor pe o conexiune
- **Full duplex**
- Stabilire sigură a conexiunii - three-way handshake
- Eliberare lină a conexiunii – fără pierdere de date



# Antet segment TCP



**Sequence number** – numarul **primului** octet din segment

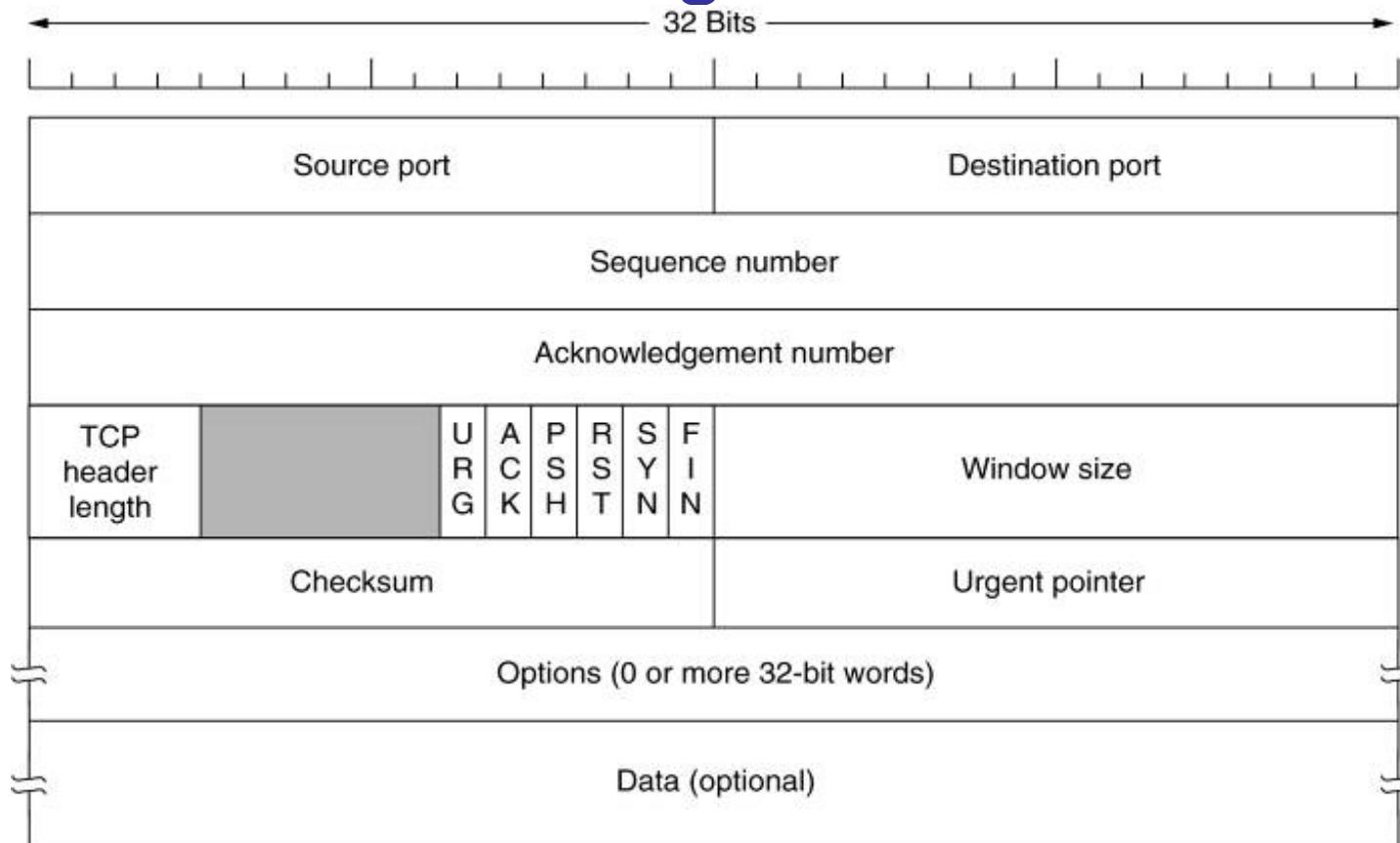
**Acknowledgement number** – numarul **urmatorului** octet asteptat

**Window size** - numărul de **octeți** care pot fi trimiși, începând cu octetul confirmat

**Urgent Pointer** – deplasamentul, față de **Sequence number**, ptr. info. urgentă



# Antet segment TCP



**URG** Urgent pointer valid

**ACK** Acknowledge Number valid

**PSH** - push information to user

**RST** - close a connection due to an error

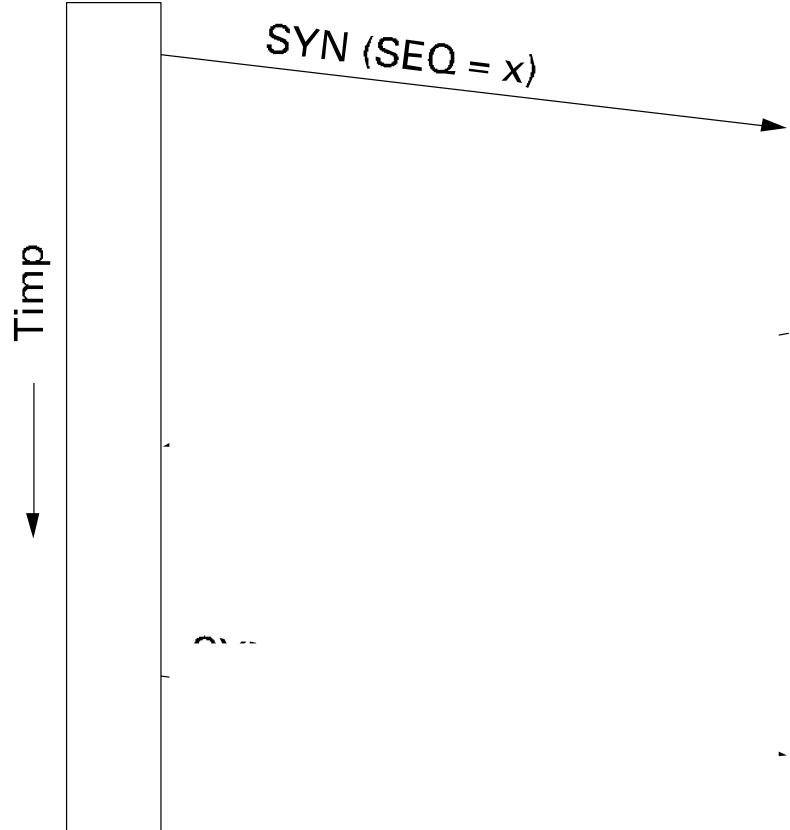
**SYN** - open connection

**FIN** - close a connection

**Options**: e.g. max TCP payload (implicit **536 octeti**), selective repeat

# Stabilirea conexiunii - Three way handshaking

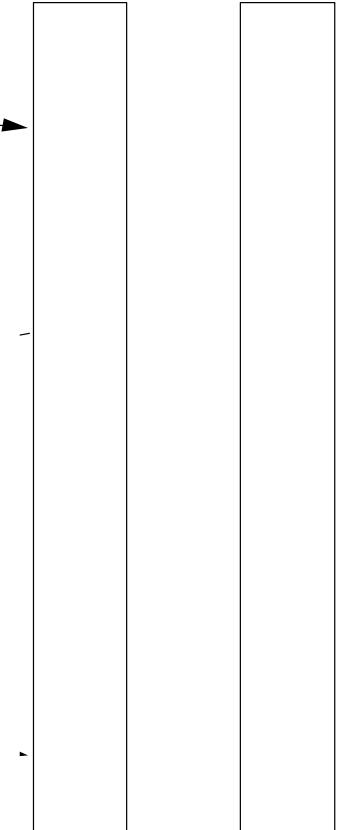
Gazdă 1



(a)

(a) Cazul normal.

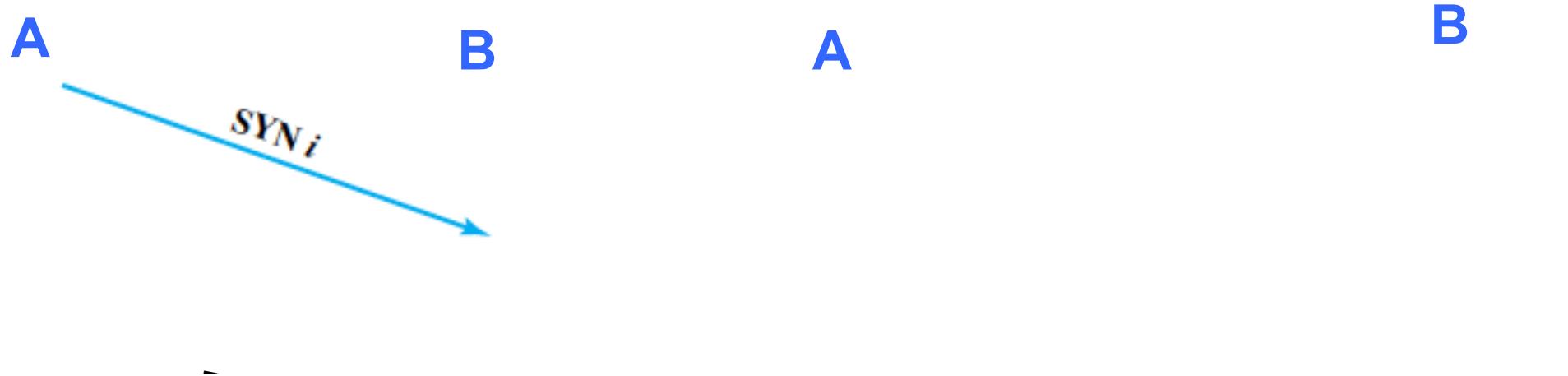
Gazdă 2



(b)

(b) Coliziune.

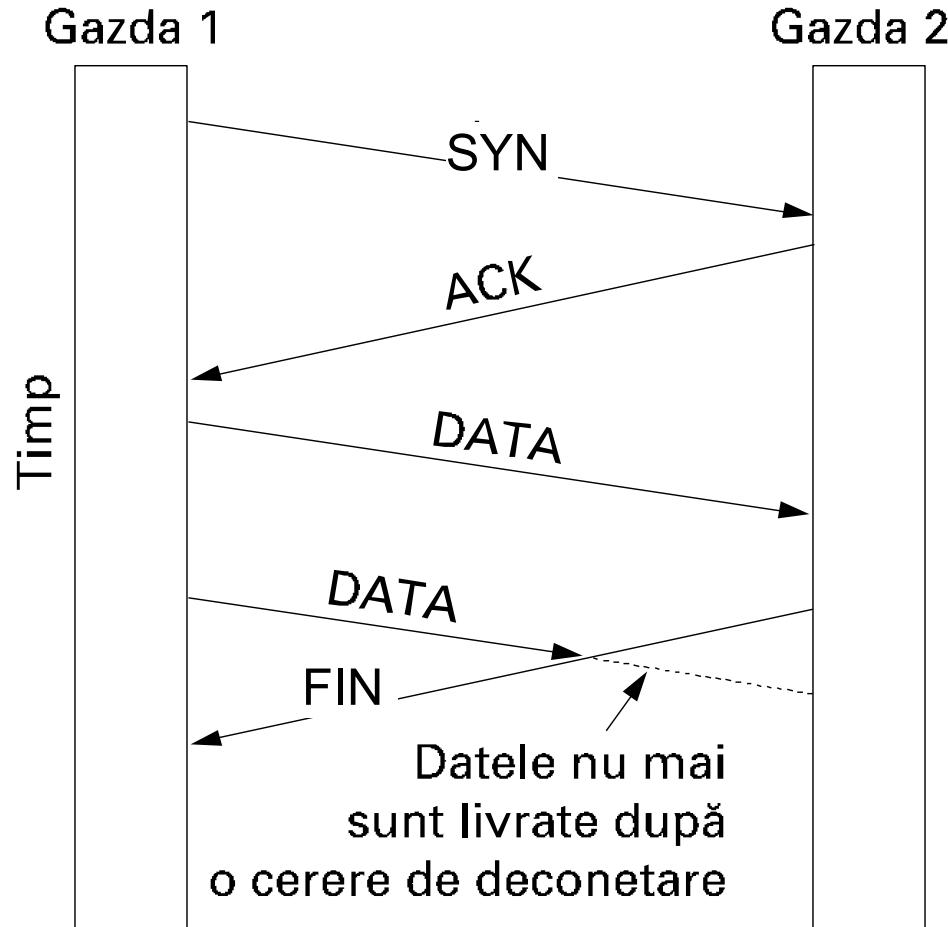
# Rejectarea conexiunii



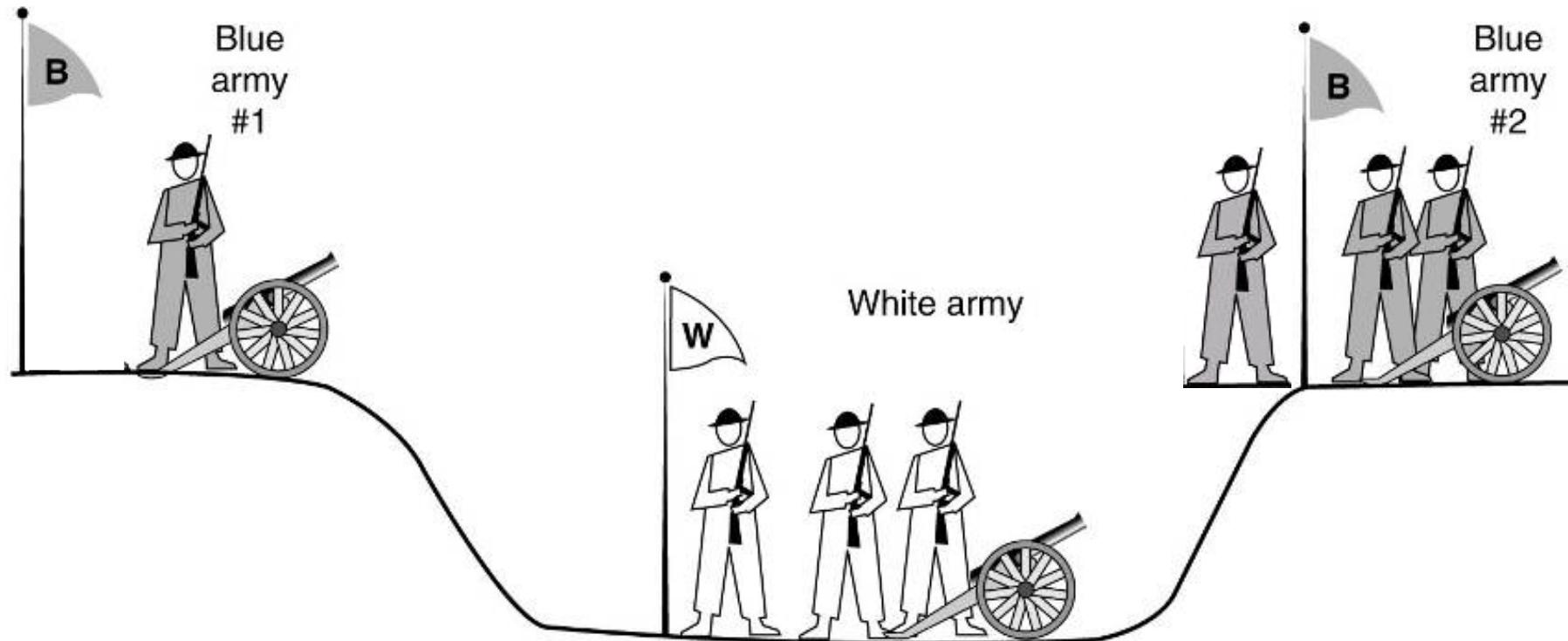
SYN intarziat  
B accepta  
A rejecteaza

A initiaza conexiune ( $SYN_i$ ) +  $SYN_k$  intarziat  
A refuza (RST)  
B accepta  $SYN_i$  – raspunde cu  $SYN_j$

# Deconectare abruptă cu pierdere de date

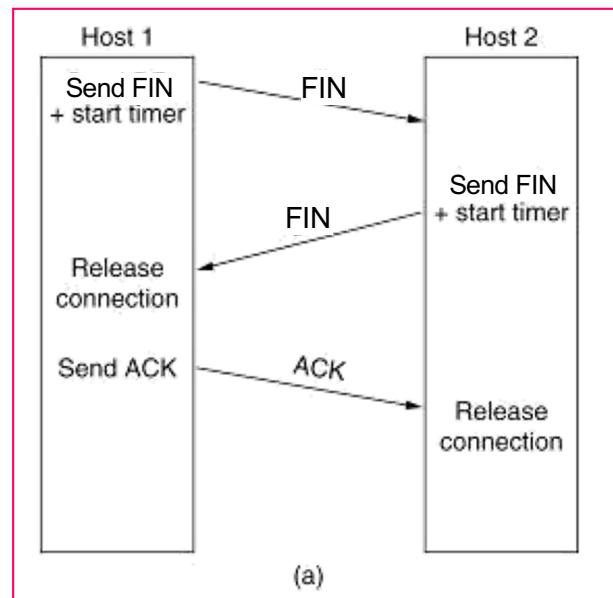


# Problema celor două armate



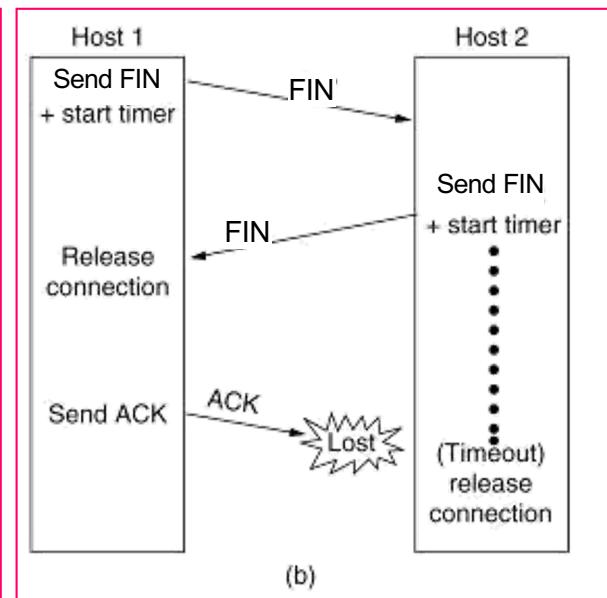
## **Patru scenarii de eliberarea conexiunii**

normal



ACK final  
pierdut

**raspuns  
pierdut**

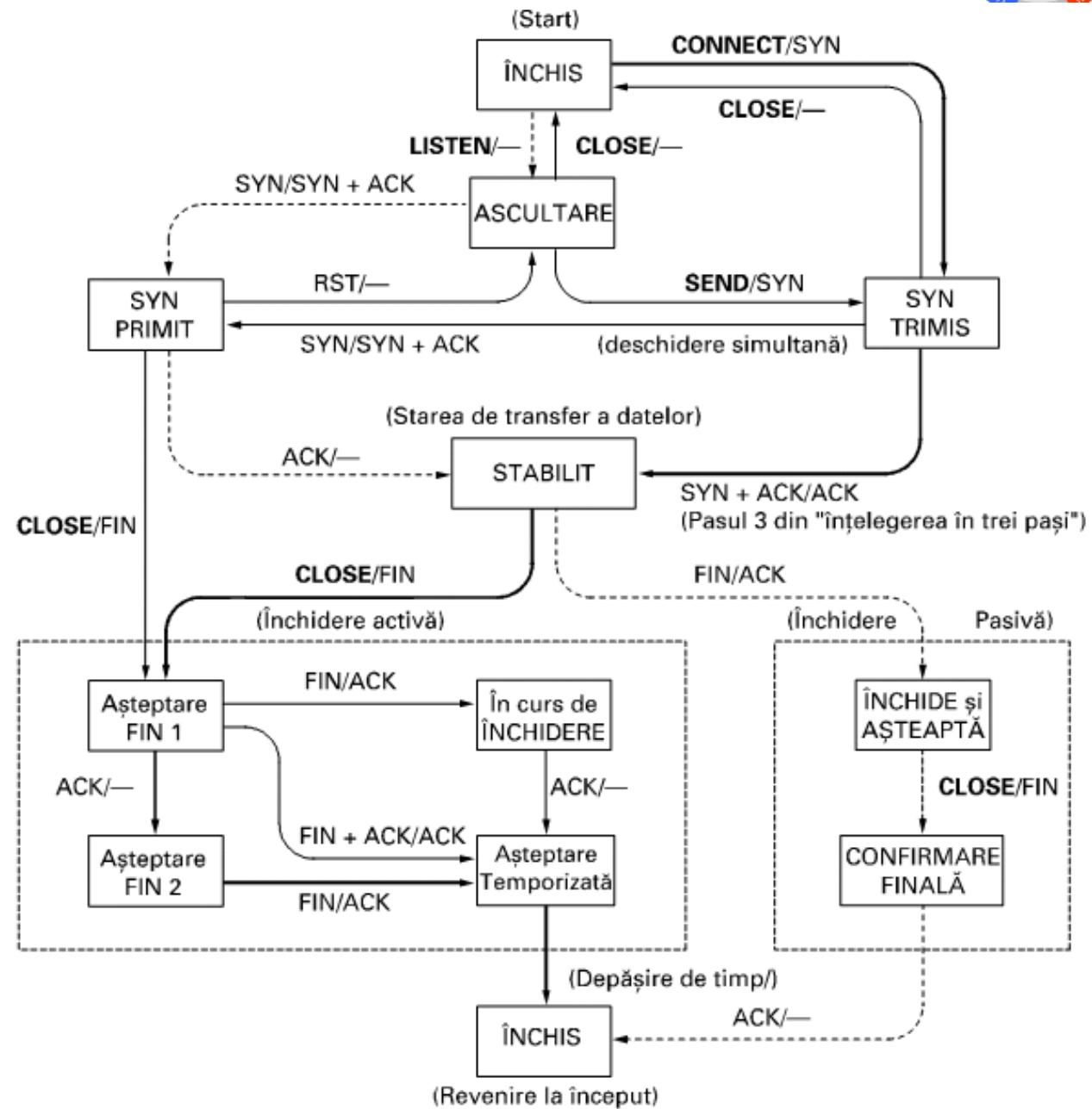


# Raspuns si FIN pierdute



# Gestiunea Conexiunilor TCP (mașina de stări)

- client (**activ**) cere conectarea
- server (**pasiv**) raspunde
- Linie groasa continua = cale normala pentru client
- Linie intrerupta = cale normala server
- Linii subtiri = evenimente exceptionale
- Fiecare tranzitie are eticheta **eveniment / actiune**

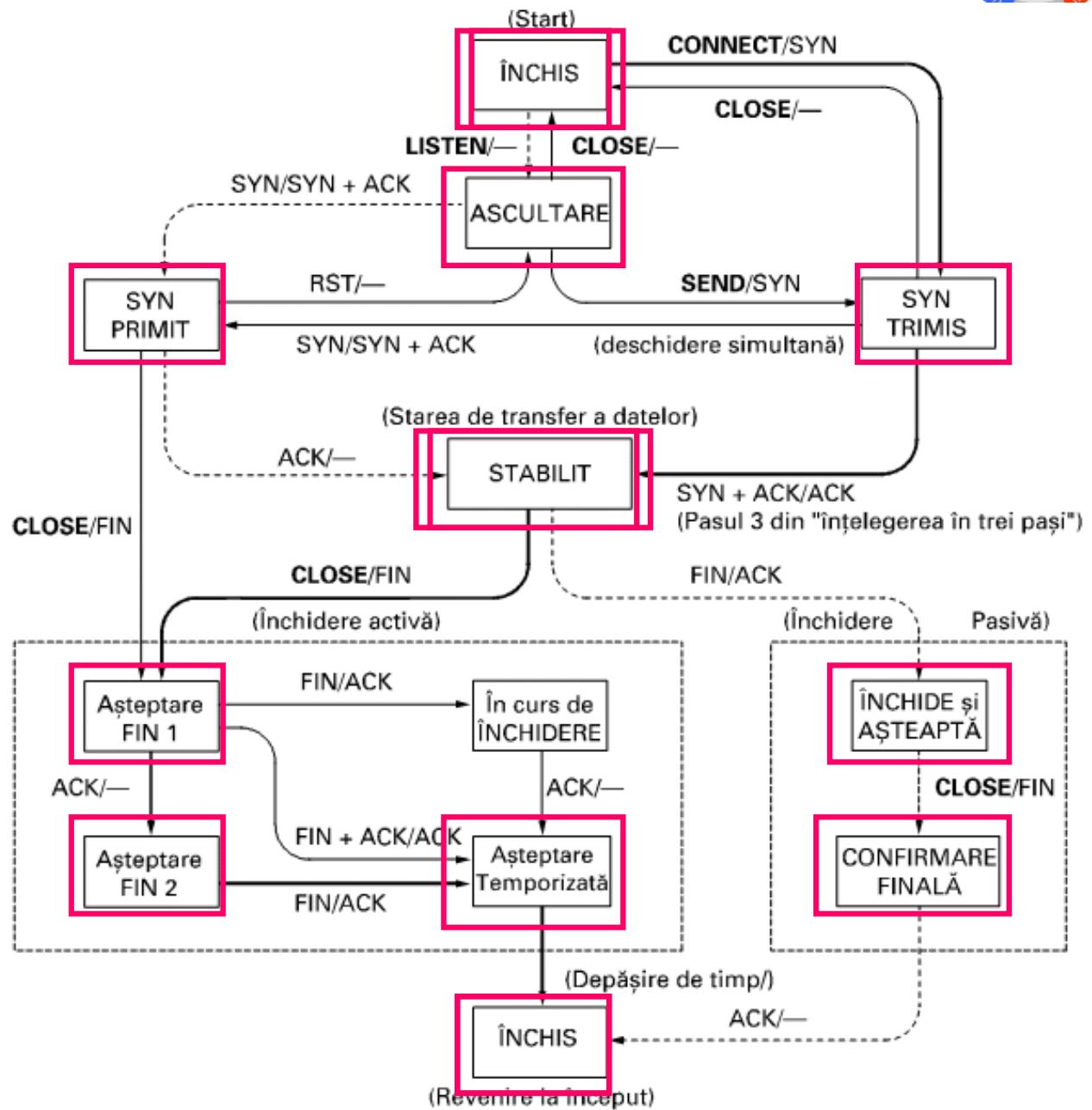


# Client

**Linie groasa  
continua**

# Server

# Linie interrupta





# Corectitudinea segmentelor TCP

1 - Suma de control (checksum) din antet are 16 biti si include

1. antet
2. incarcatura segment TCP (date)
3. un pseudo-antet

adresele IP sursa si destinatie

protocolul (6 pentru TCP)

lungime segment TCP (include antetul)

## Algoritm: la transmisie

aduna cuvinte de 16 biti in complement fata de 1  
complementeaza rezultatul  
scrie rezultatul in antet

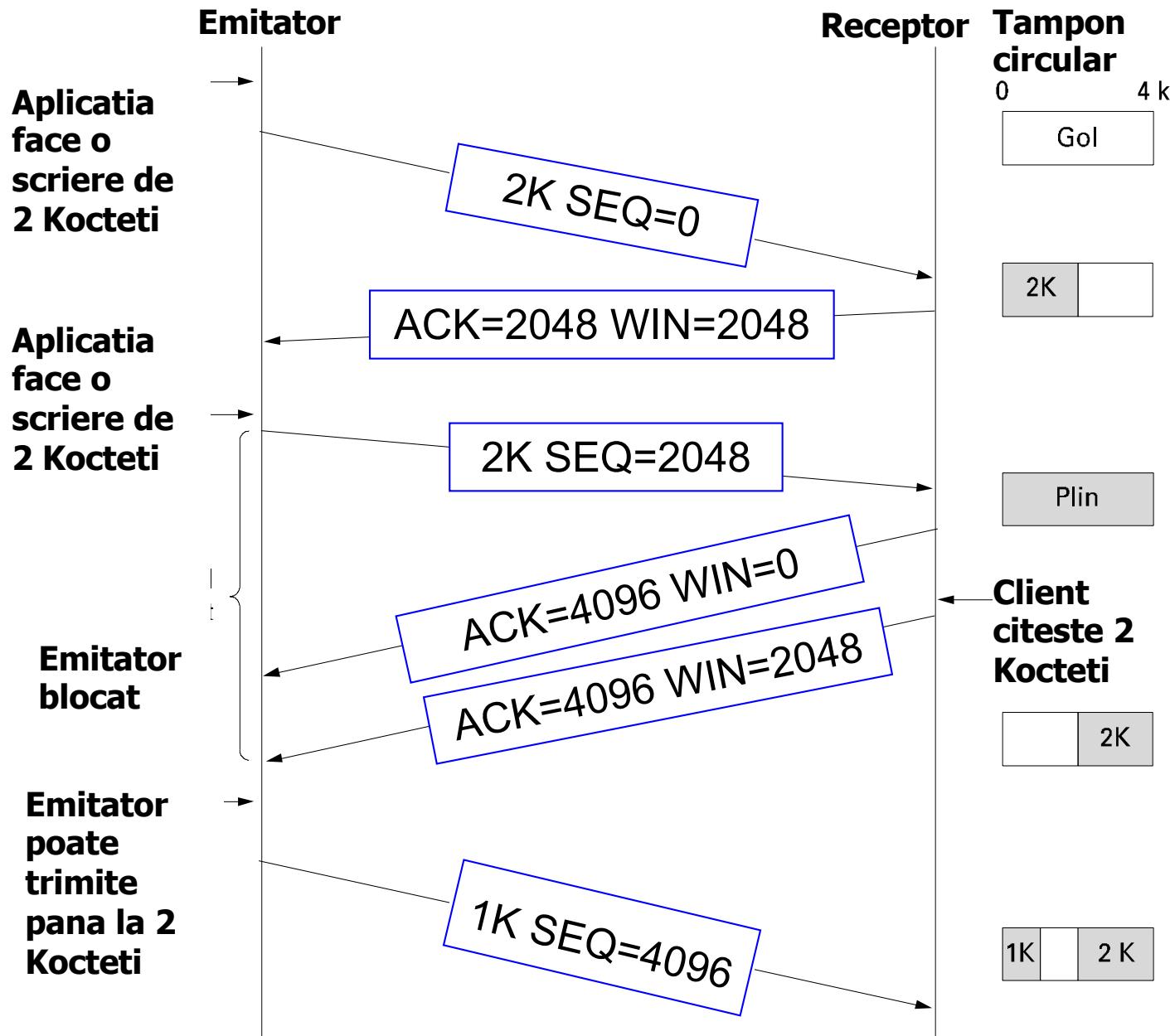
## la receptie

aduna cuvinte de 16 biti – rezultatul trebuie sa fie zero

## 2 - Acknowledgement number

Corectia se face prin retransmisie

# Controlul fluxului de date



Receptorul specifică fereastra de receptie disponibila

Un emitor blocat poate trimite:

- date urgente
- un segment de 1 octet pînă a afla fereastra (daca anuntul precedent al receptorului s-a pierdut)



# Probleme dimensiuni câmpuri antet

## Numere secvență de 32-bitii

- Durata ciclu de numarare – depinde de viteza transmisie
  - 1 saptamana pentru 56kbps
  - 57 min pentru 10Mbps
  - 34 sec pentru 1Gbps (**sub 120 sec care este timp viata maxim in Internet**)

**Problema:** pot apare segmente diferite, cu acelasi numar de secventa?

## Soluție

- Folosire opțiuni TCP (RFC 1323)
  - **TCP Timestamps**
    - asociaza o amprenta de timp fiecarui segment
    - rezolva numere de secventa duplicate



## Probleme dimensiuni câmpuri antet (2)

**Fereastra receptor** (camp **Window size** de 16 biti – echivalent 64 KB)

Transmitere **500 Kb** pe legatura **1 Gbps** ocupa  $500 \mu\text{sec}$

La intarziere de **20 ms** pe sens confirmarea se primește după **40 ms** =>  
**ocupare canal** pe un ciclu complet este mică - **1.25%**

Ocupare completă în ambele direcții: produs **bandwidth\*delay** =  
= **1 Gbps \* 40 ms** = 40 milioane biti

**Condiții** - fereastra receptor ar trebui să fie  $\geq$  **bandwidth\*delay**

- camp **Window size** nu se poate mari

### Soluție

- Folosire **opțiuni TCP** (RFC 1323)
  - **Window Scale** – factor de scalare a câmpului **Window size** de până la  **$2^{14}$**  ori
    - ➔ ferestre de până la  **$2^{30}$**  octeți



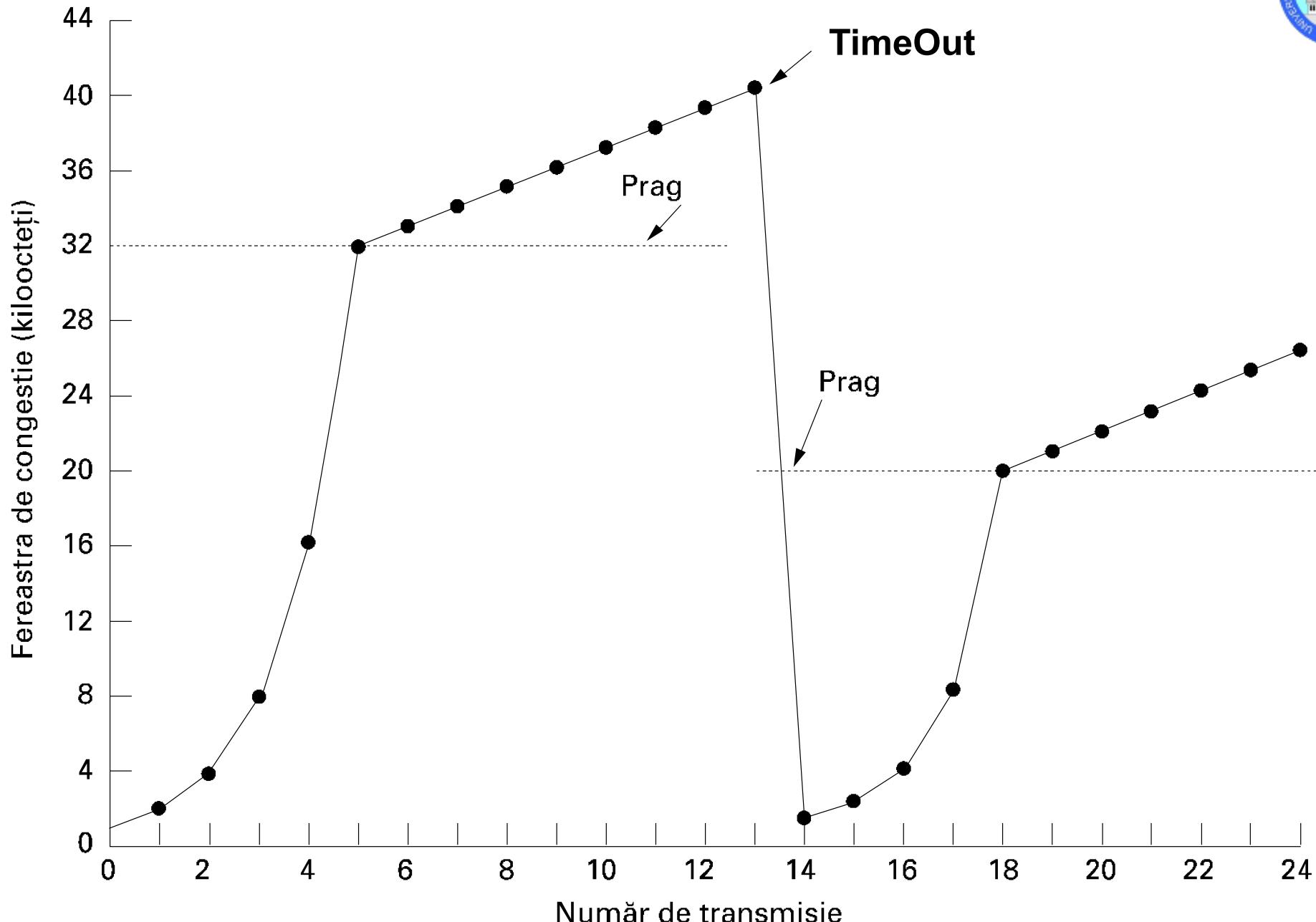
# Controlul congestiei

- Fluxul de date transmis pe o conex. TCP limitat de minimul dintre:
  - dimensiunea fereastrei receptorului
  - capacitatea retelei (**fereastra de congestie**)
- Algoritm de **stabilire fereastra de congestie**
  - transmite un segment de **dimensiune maximă** pe conexiunea stabilită
  - dubleaza volumul de date – rafală de segmente - (creștere exponențială) la fiecare transmisie confirmată la timp
  - la **primul timeout** opreste procedeul si fereastra ramane la valoarea ultimei transmisii confirmate la timp (fara timeout)

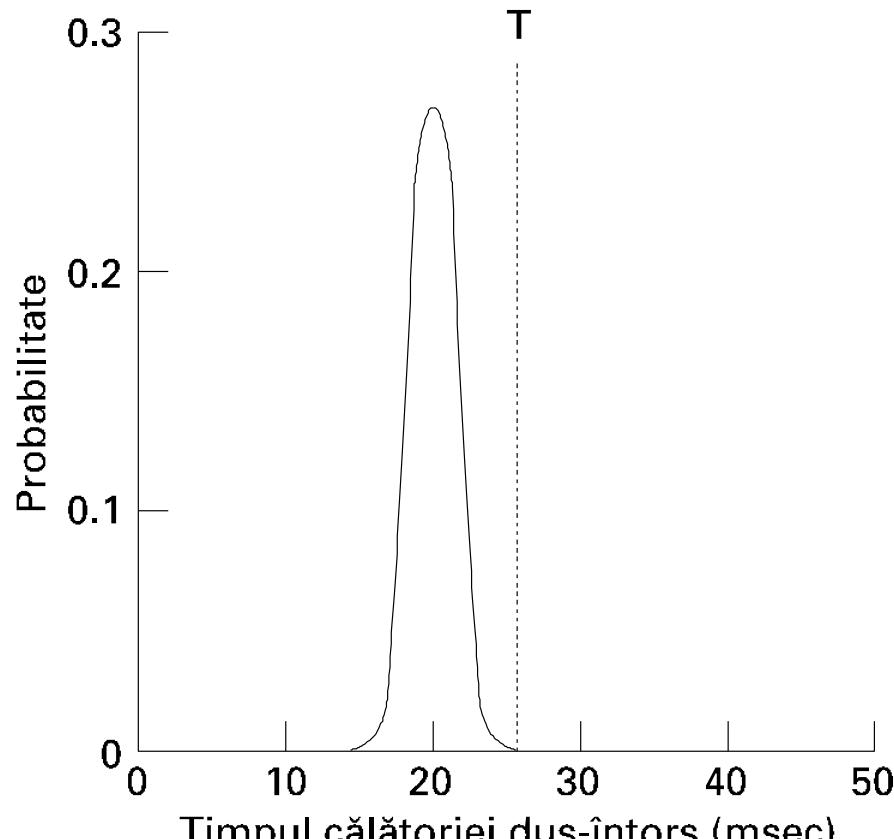


# Controlul congestiei

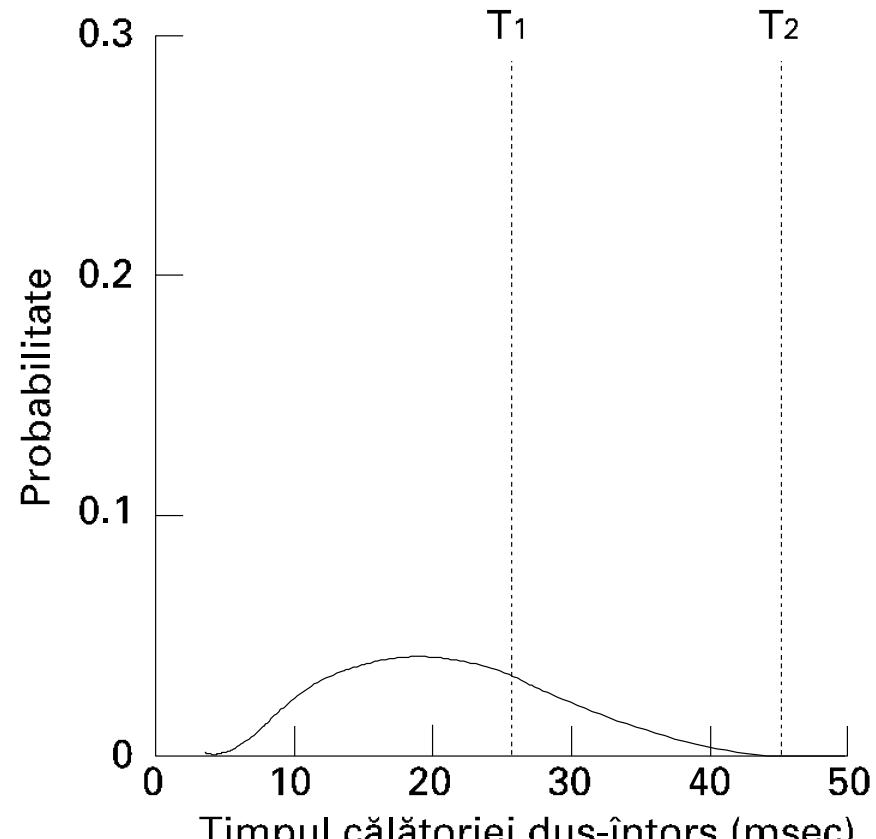
- Algoritmul de control al congestiei
  - foloseste un **prag** (threshold)
  - la un timeout pragul setat la jumate din fereastra de congestie
  - se aplica procedeul de crestere (exponentiala) a fereastrei de congestie pana se atinge pragul
  - peste prag se aplica o crestere liniara (cu cate un segment de dimensiune maximă o dată)



# Gestiunea ceasurilor în TCP



(a)



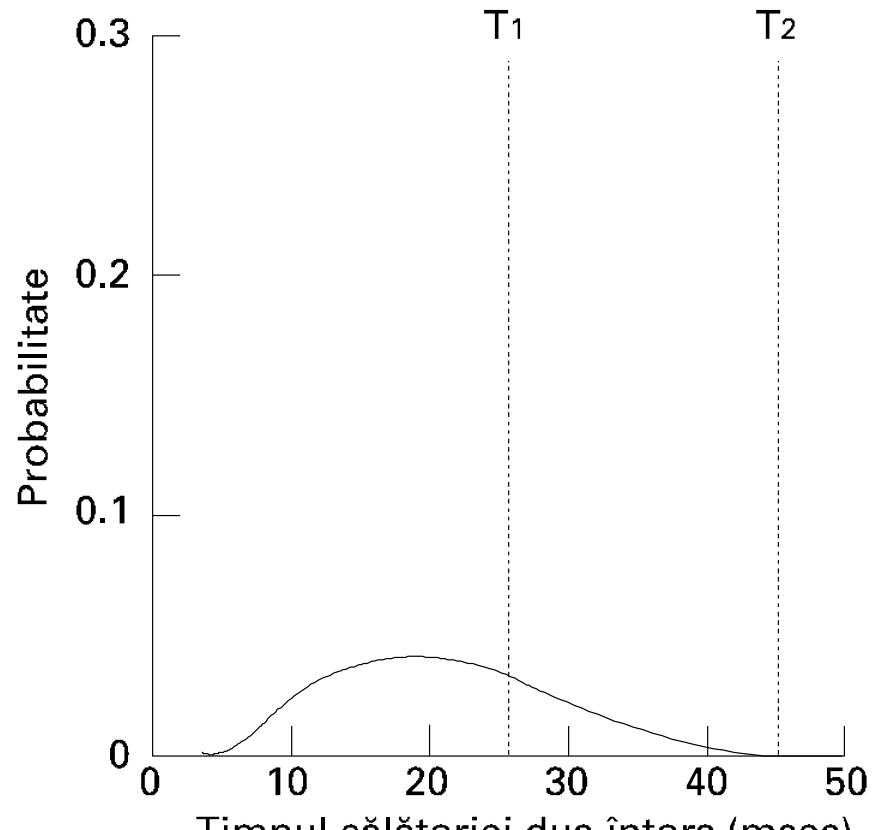
(b)

- (a) Densitatea de probabilitate a timpilor de sosire ACK în nivelul **legatura de date**.  
(b) Densitatea de probabilitate a timpilor de sosire ACK pentru **TCP**.

## Gestiunea ceasurilor în TCP (2)

Configurare incorectă – performante slabe:

- Prea lung ( $T_2$ ) – transmitatorul asteapta mult ptr retransmisie
- Prea scurt ( $T_1$ ) – trafic inutil generat de transmitator



(b)



## Stabilire time-out

- Timeout diferit la fiecare conexiune - setat dinamic
- Se folosesc metode empirice
- Transmitatorul alege *Retransmission TimeOut (RTO)* pe baza *Round Trip Time (RTT)*

$M$  este timpul masurat pana la primirea ack

$$RTT = \alpha * RTT + (1 - \alpha) * M \quad \text{cu } \alpha = 7/8$$

$$RTO = \beta * RTT \quad \text{cu } \beta = 2$$

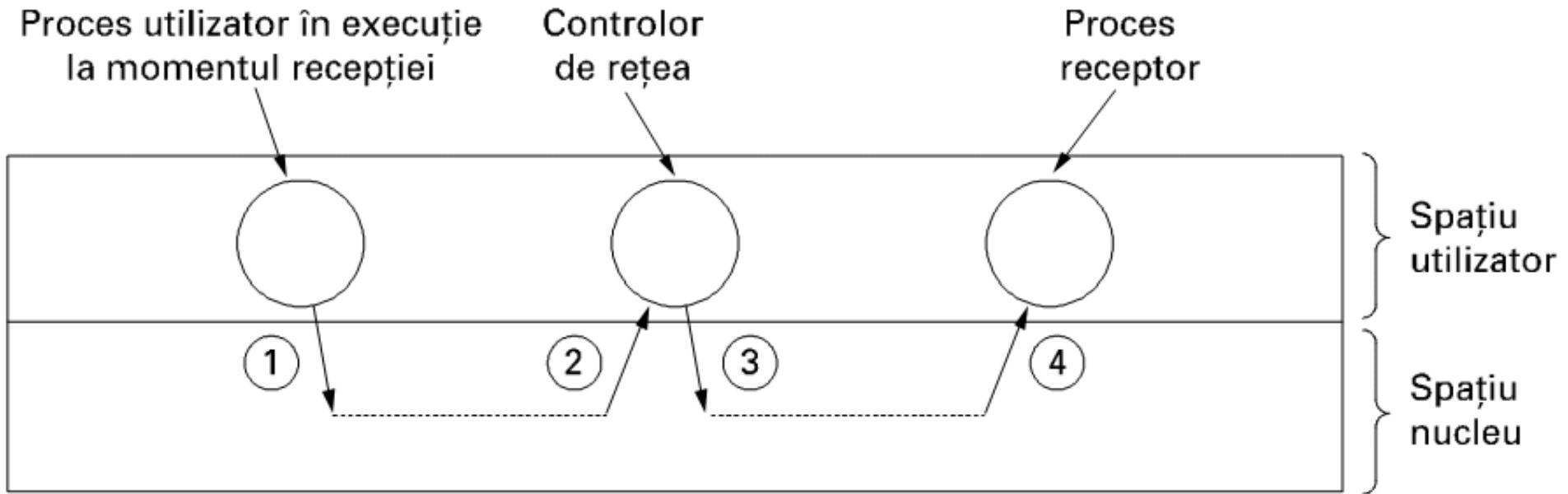
- Alegere dupa *deviatia standard (DS)*;

$D$  aproximeaza DS

$$D = \alpha * D + (1 - \alpha) * |RTT - M|$$

$$RTO = RTT + 4*D$$

# Proiectarea pentru performanta



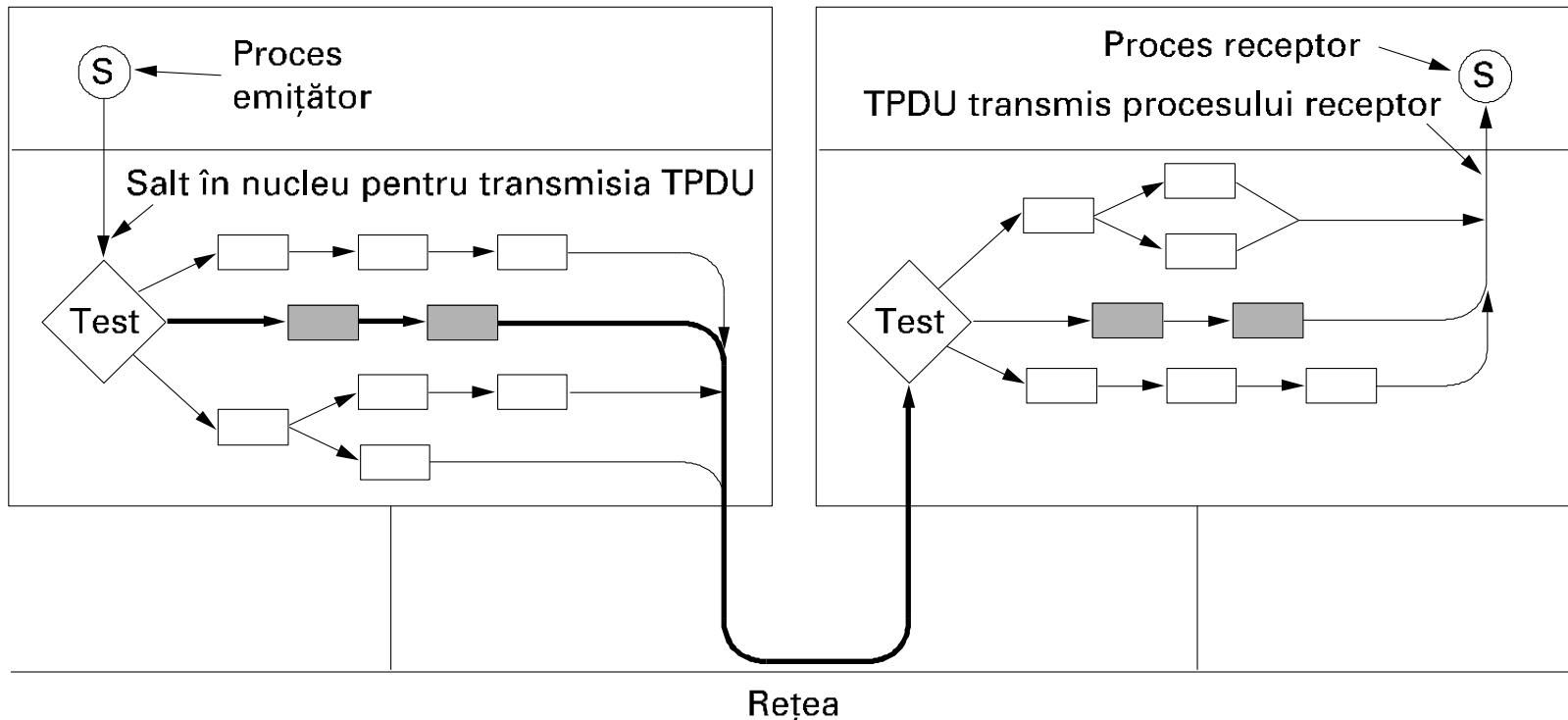
**Problema:** Multe schimbari de context user – kernel pentru a manevra pachete cu un Controlor de retea in spatiul utilizatorului.

Controlorul asambleaza segmentele pentru a le furniza apoi procesului receptor

**Solutia:** reducerea numar schimbari de context - acumularea mesajelor sosite, in memorie tampon si livrarea in grup catre utilizator.

Similar, gruparea mesajelor de transmis, in memorie tampon si trimiterea lor grupata

# Prelucrare rapida TPDU



Calea rapida intre transmitator si receptor este cu line groasa. Pasii sunt reprezentati cu gri.

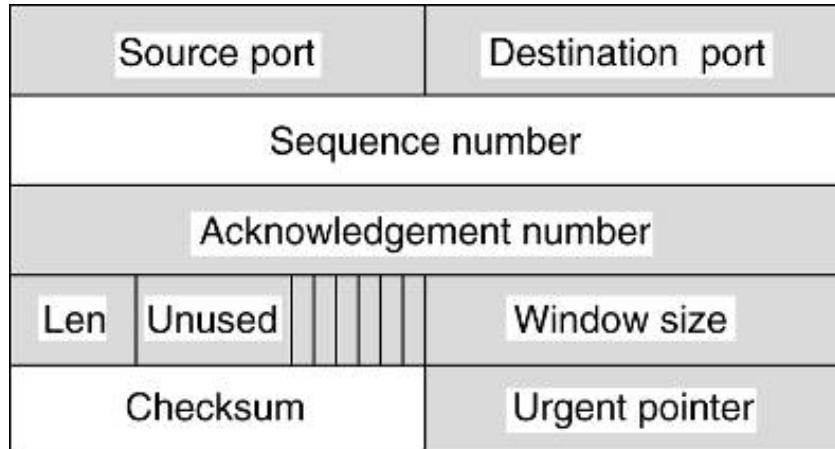
## Test caz normal:

starea = ESTABLISHED  
 nu se incearca inchiderea conexiunii,  
 TPDU normal (nu out-of-band),  
 suficient spatiu la receptor

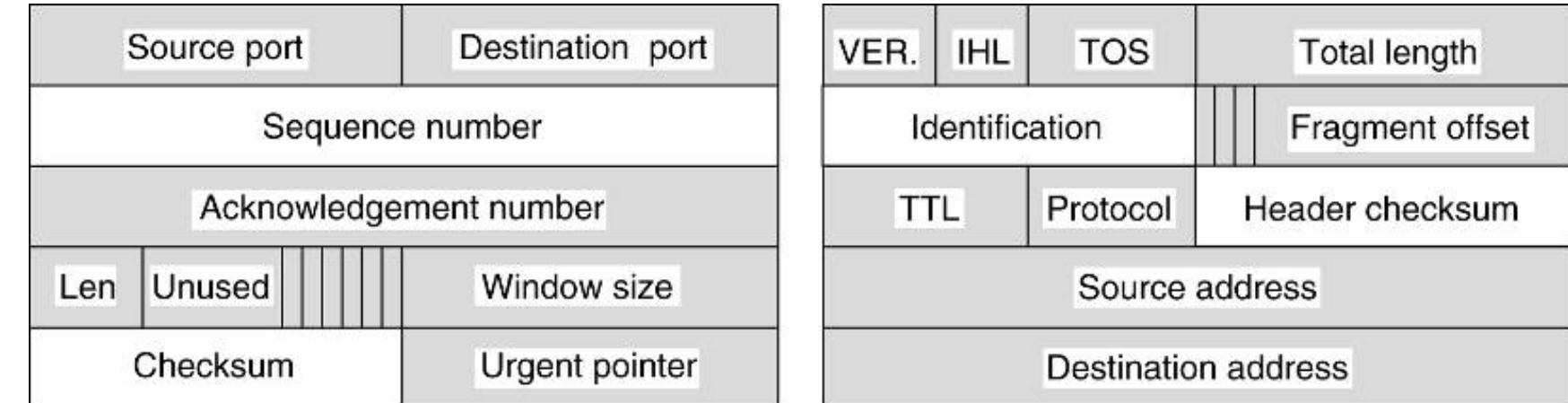
} → alege **fast path**



# Prelucrare rapida TPDU (2)



(a)



(b)

(a) Antet TCP.

(b) Antet IP.

## Transmitator

Pastreaza un **prototip** de mesaj in entitatea de transport – campuri nemodificate in unitati de date consecutive; la fel pentru pachet IP

Campurile gri sunt luate din **prototip** fara modificari. Celelalte se calculeaza pentru fiecare segment

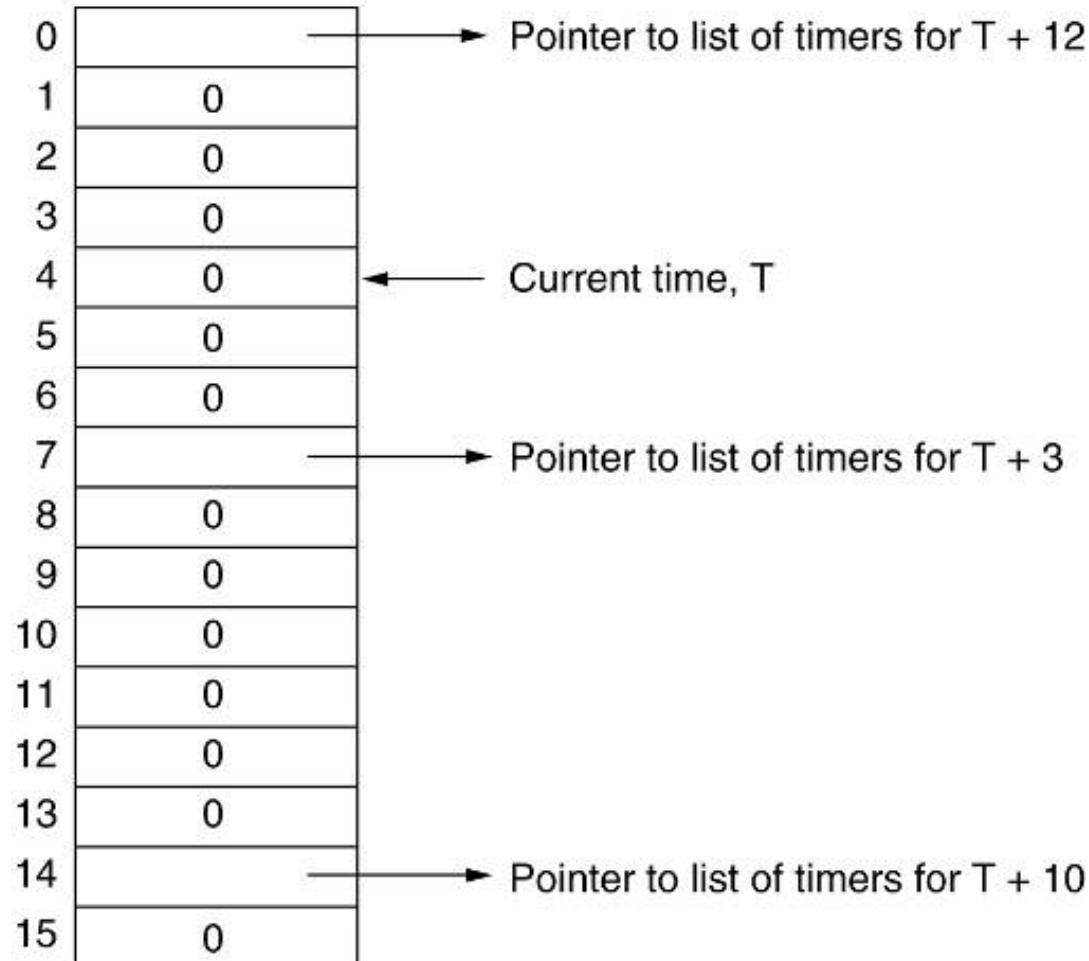
## Receptor

- Localizeaza inregistrarea conexiunii din TPDU intr-o **tabela hash**
- Testeaza pentru cazul normal (similar cu transmisia)
- Actualizeaza inregistrarea conexiunii (starea curenta)
- Copiaza datele la utilizator si calculeaza suma de control
- Transmite confirmarea



# Prelucrare Fast TPDU (timer management)

Slot



“Timing wheel.”

1 slot = 1 clock tick

Timp curent T=4

Programare time-out peste 7 tick-uri  
→ insereaza eveniment in lista de la slot 11

Anulare → cauta in lista de la slot corespunzator si elimina eveniment

La fiecare **Clock tick**, un pointer avaneaza cu un slot, circular

Daca slot nevid, proceseaza toate evenimentele



# The Real-Time Transport Protocol

Folosit pentru aplicatii multimedia de timp real

- muzica sau video la cerere, videoconferinte etc.

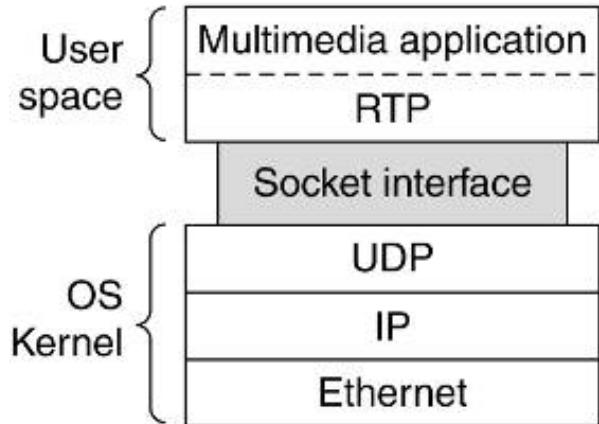
**Functie principala:** multiplexare fluxuri RTP si transmiterea lor ca un **singur sir** de pachete **UDP**

La receptie, RTP livreaza aplicatiei datele multimedia

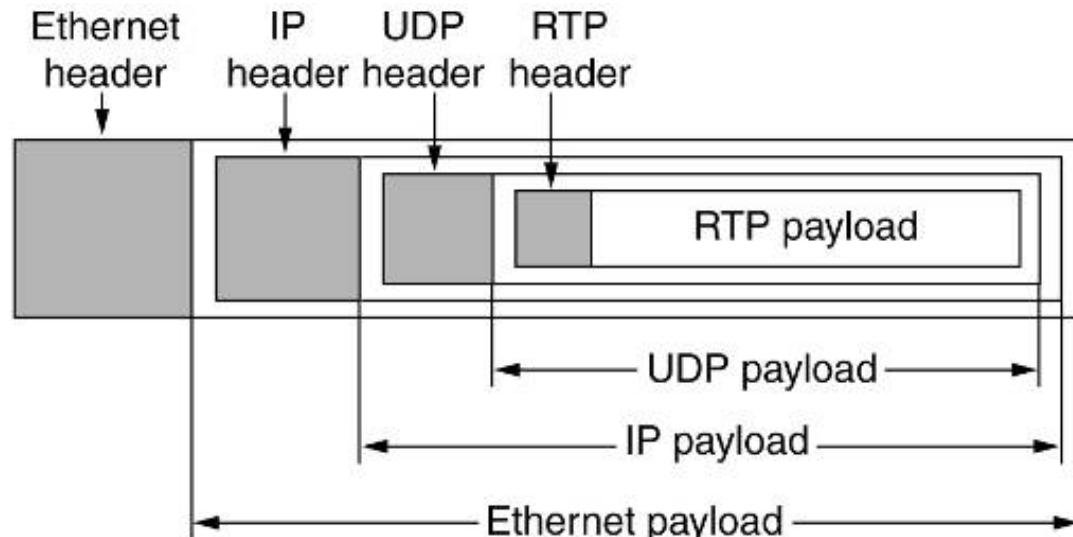
**Fara retransmiterea pachetelor**

- UDP nu asigura corectitudinea transmisiei
- RTP atribuie un numar de **seventa** fiecarui pachet
- **tratarea** unui pachet pierdut se face **la aplicatie**: receptorul poate “interpolă” pachetul absent sau ignora eroarea

# The Real-Time Transport Protocol (2)



(a)



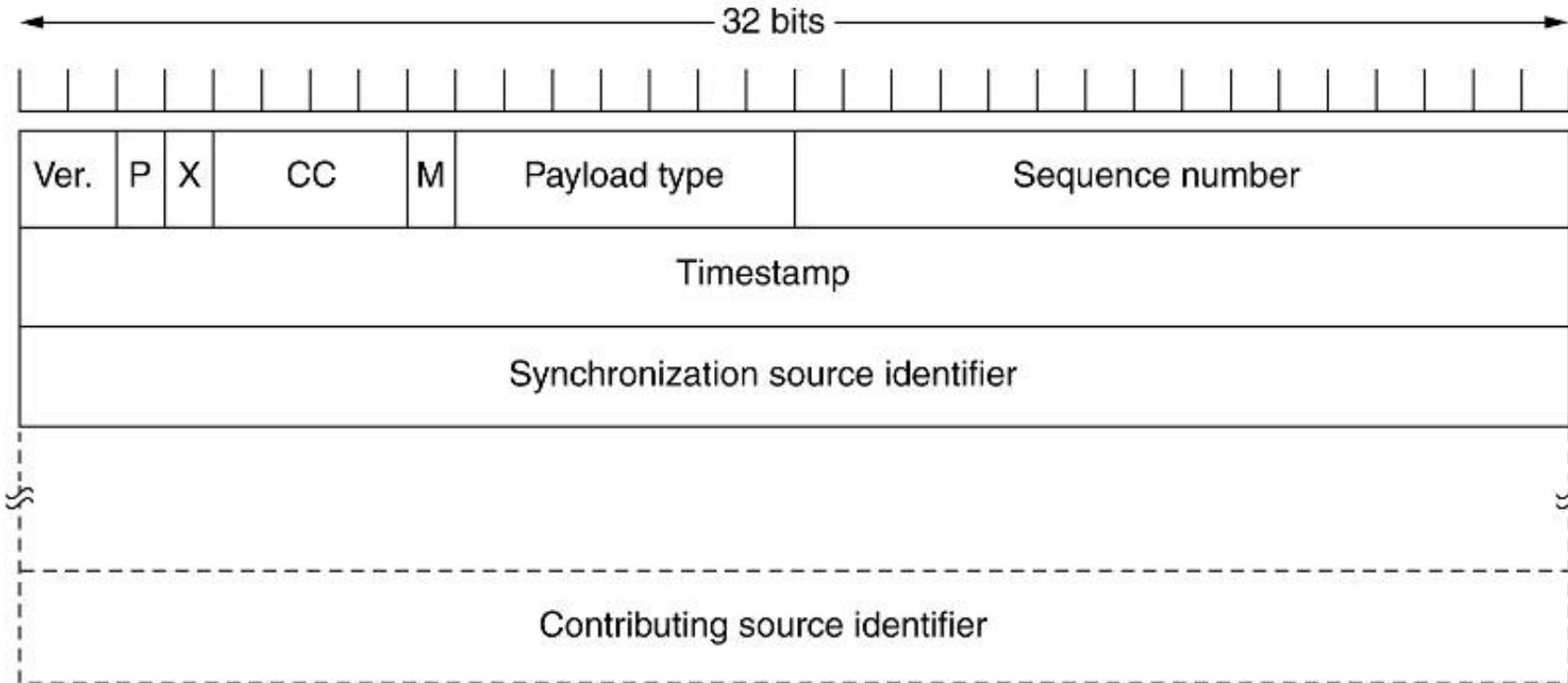
(b)

(a) Pozitia RTP în stiva de protocoale.

(b) Format pachet.

**Formatul** datelor din "RTP payload" este specific aplicatiilor

RTP permite definirea unor **profile** (ex. single audio stream) si, pentru fiecare profil, a mai multor **formate**



## Antet RTP

**Timestamp** – amprenta de timp relativa la inceputul fluxului

- reduce efectul intarzierilor variabile
- permite sincronizarea intre stream-uri

**Synchro** – identifica sursa de sincronizare (ex. microfon, camera video)

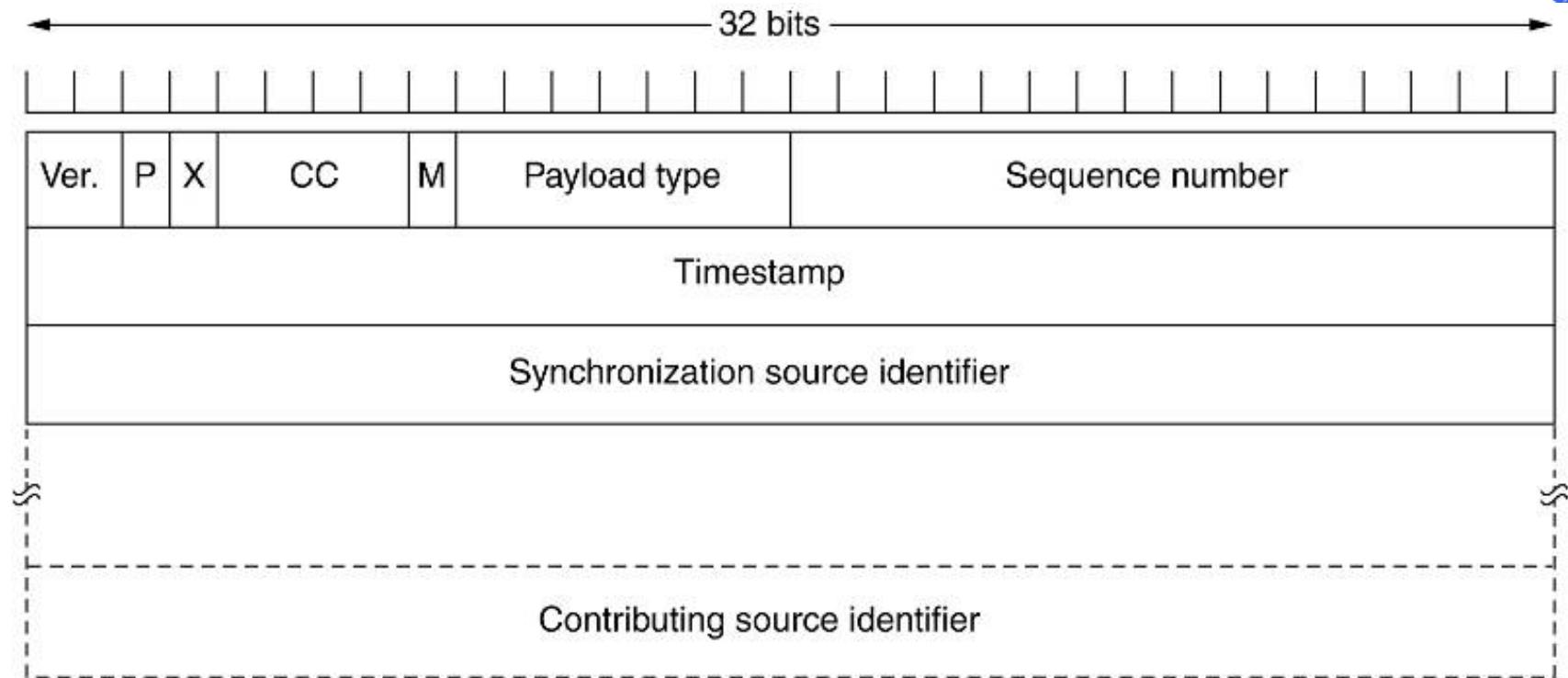
- receptorul grupeaza pachetele dupa sursa, pentru redare

**Contrib** – lista surselor care au contribuit la continutul curent; folosit pentru mixere;

- ex. la audio-conf, lista vorbitorilor ale caror discursuri sunt in pachet curent



# Antet RTP



**P** – padding - pachet extins la multiplu de 4 octeti

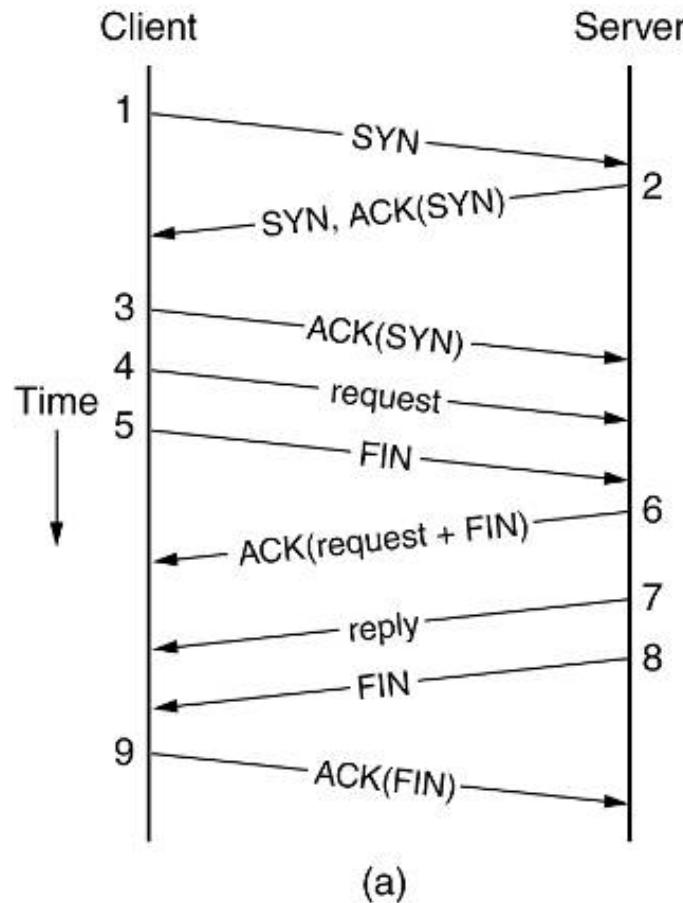
**X** – extension - antet extins (primul cuvant da lungimea)

**CC** – no. antete surse contributoare

**M** – mark (specific aplicatiei. Ex. start [video frame](#))

**Payload type** – e.g. MP3

# TCP Tranzactional



(a) RPC – Remote Procedure Call - folosind TPC normal.

(b) RPC folosind T/T  
**Permite transfer de date odata cu stabilirea conexiunii**



# Nivelul Aplicație

DNS



# Cuprins

- De ce este nevoie de DNS
- Spatiul de nume DNS – structura logica arborescenta
- Component DNS
- Inregistrari de resurse
- Serverul de nume
- Protocolul DNS
- Rezolvarea recursiva si iterativa a numelor
- Cereri inverse
- Replicarea serverelor DNS



# Protocole de aplicații și Servicii Middleware

**Aplicatii** (procese client si server)

**Servicii de obiecte distribuite**  
(CORBA, OLE/ActiveX)

**Servicii speciale**  
(Wireless, multimedia, groupware, legacy)

**Gestiunea datelor distribuite**  
si procesarea distribuita a tranzactiilor

**World Wide Web**  
(**HTTP**, HTML, Web browsers, Java, servere, motoare de cautare)

**Servicii client/server de baza**  
(RPC, RDA, MOM, Securitate,  
**Directoare**, Timp)

**Servicii primitive**  
(Telnet, **E-mail**, **FTP**)

**Servicii de programare a retelei**  
(Sockets, LU6.2, NetBIOS, TLI)

**Servicii de retea**  
*Transport* (TCP/IP, SNA, SPX/IPX, NetBIOS)  
*Conexiune fizica* (Ethernet, Token Ring, FDDI, ISDN, X.25, ATM, Frame Relay)



## De ce este nevoie de DNS ?

- Protocoalele **client – server** folosesc nivelul transport (TCP, UDP) pentru schimb de mesaje

Ex.: descarcarea unei pagini Web cand utilizatorul cunoaste adresa IP si portul serverului

- Browser - deschide o conexiune TCP la port 80 pe 18.23.0.23
- Browser - trimit o comanda GET indicand adresa IP, portul si calea la **fisierul care contine pagina TheProject.html**
- Serverul trimit **fisierul TheProject.html**
- Conexiunea TCP este inchisa
- Browser - afisează conținutul din **TheProject.html**



# Port

**Se folosesc porturi fixe pentru servicii standard**

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

**Adresele IP sunt mai greu de tinut minte !**

**Se folosesc adrese simbolice (nume de domeniu) a caror translatare in adrese IP este facuta de DNS**



# Adrese simbolice: nume de domeniu

Continute în URL – Uniform Resource Locator

<b>schema</b>	protocol (http, ftp etc.)
<b>host</b>	ptr. web <b>nume</b> / adresa IP a serverului Web
<b>port#</b>	numar port server Web (80 pentru http)
<b>path</b>	calea de la radacina serverului la resursa

Schema	Utilizat pentru	Exemple
http	Hipertext (HTML)	http:// <span style="border: 1px solid red; padding: 0 2px;">www.cs.vu.nl</span> /~ast
ftp	FTP	ftp:// <span style="border: 1px solid red; padding: 0 2px;">ftp.cs.vu.nl</span> /pub/minix/README
mailto	Trimitere de poșta electronică	mailto: <span style="border: 1px solid red; padding: 0 2px;">JohnUser@adm.org</span>
telnet	Conectare la distanță	telnet://www.w3.org:80



# Descarcarea unei pagini Web

Utilizatorul cunoaste numele simbolic al serverului si calea spre fisierul ce contine pagina

- Browser - determina URL <http://www.w3.org/TheProject.html>
- Browser - cere unui server DNS adresa IP pentru www.w3.org
  - server DNS - raspunde cu 18.23.0.23
- Browser - deschide o conexiune TCP la port 80 pe 18.23.0.23
- Browser - trimit o comanda la server Web

**GET TheProject.html HTTP/1.1**

**Host: www.w3.org**

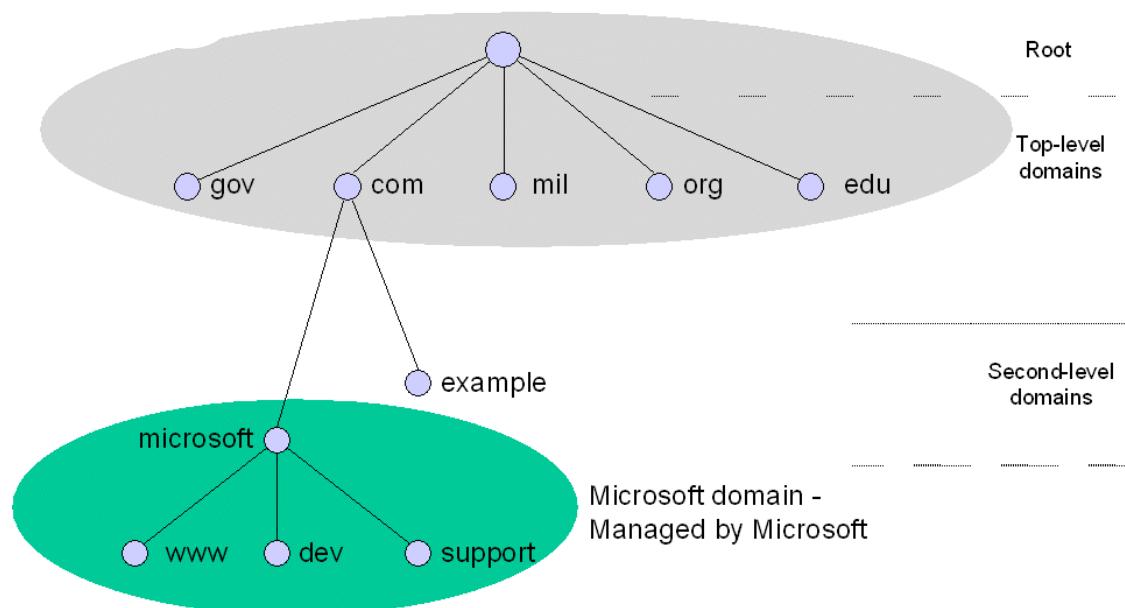
- Server Web [www.w3.org](http://www.w3.org) trimit fisierul TheProject.html
- Conexiunea TCP este inchisa
- Browser - afiseaza continutul din TheProject.html

# DNS – The Domain Name System

**Spatiul de nume** DNS – structura logica arborescenta

Fiecare **nod** din arbore reprezinta un **domeniu**

- Radacina
- De nivel inalt (gov, com,... ) administrate de ICANN - **Internet Corporation for Assigned Names and Numbers**
- De nivel 2 (ex. [microsoft.com](http://microsoft.com)) ... etc.
- **Frunzele** corespund gazdelor

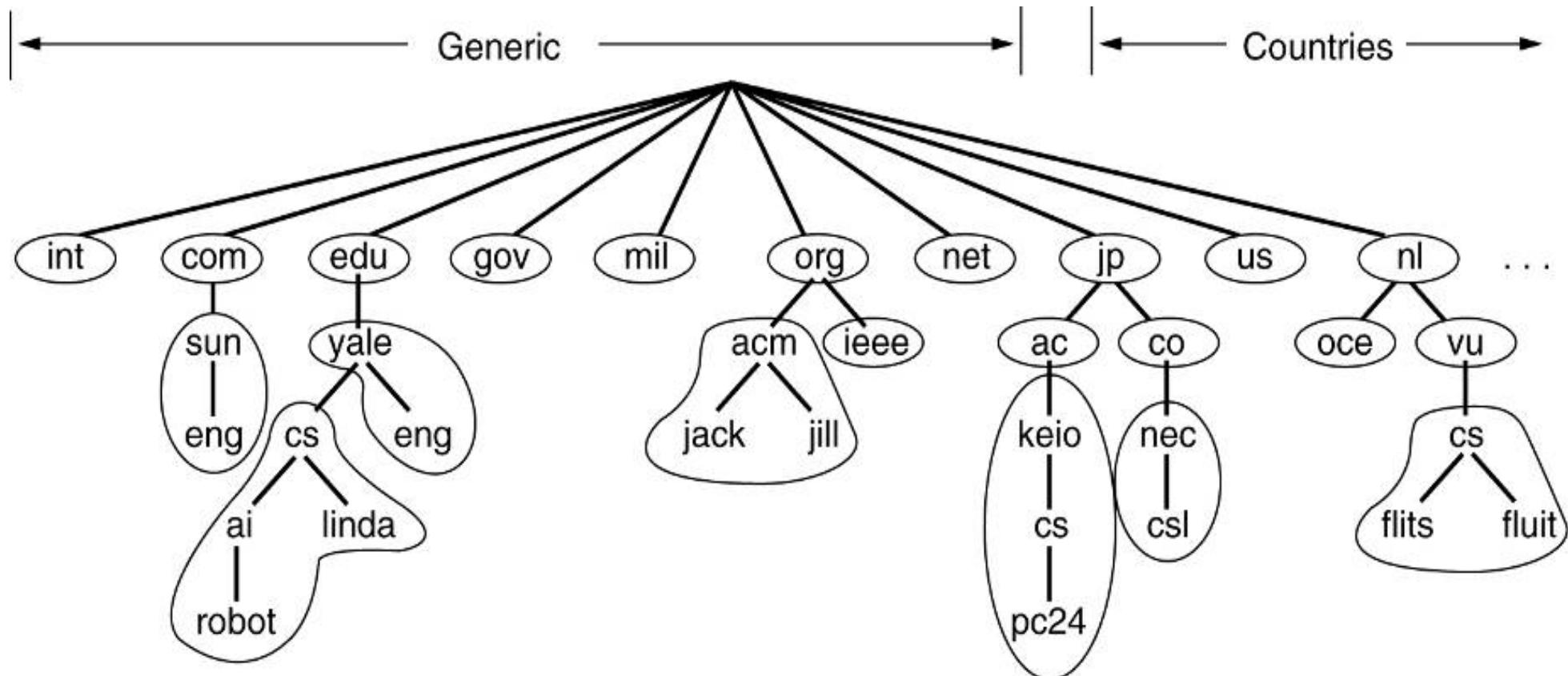


Numele unui domeniu  
foloseste numele nodurilor  
pe calea la radacina

[example.com](http://example.com)  
este diferit de  
[example.mil](http://example.mil)

# Zone DNS si Servere de Nume

- Spațiul de nume DNS este împărțit în **zone** administrate de **servere** de nume distincte (mai multe servere pot răspunde de o zona)
- Serverul de nume
  - pastrează informații pentru **unul** sau mai **multe domenii**
  - cunoaște adresele **alțor servere** (inclusiv de la nivelul inferior)





# Componente DNS

- **Spatiul de nume (namespace)**
  - organizat ierarhic
  - fiecare nod are asociat un set de informatii pastrate in **baze de date** DNS
- **Servere DNS**
  - administreaza **zone** DNS
  - pastreaza BD cu informatii necesare clientilor
  - in inregistrari de resurse (**resource records**)



# Înregistrări de resurse

O BD DNS contine o colectie de Resource Records – RRs in **format text**

Fiecare inregistrare include:

Nume\_domeniu      ex:      `srv1.dev.microsoft.com.`

- ultimul "." in `srv1.dev.microsoft.com.` este **radacina**

Timp\_de\_viata      ex:      `3600`      (in secunde)

Clasa      ex:      `IN`      (pentru Internet)

Tip      ex:      `A`      (adresa)

Valoare      ex:      `157.60.221.205`



# Principalele tipuri de înregistrări DNS

Tip	Semnificație	Valoare
SOA	Start autoritate	<b>Start Of Authority</b> - Parametrii pentru această zonă (ex. adresa E-mail a administratorului de sistem)
A AAAA	Adresa IP a unui sistem gazdă	<b>Address</b> - Întreg pe 32 de biți (A) sau pe 128 de biti (AAAA)
MX	Server de mail	<b>Mail eXchange</b> – Leg. simbolica la server de mail
NS	Server de Nume	<b>Name Server</b> - Nume server pentru acest domeniu
CNAME	Nume canonic	<b>Canonical Name</b> – Legatura simbolica cu numele primar al nodului reprezentat (pseudonim)
PTR	Pointer	<b>Pointer</b> – uzual, numele corespunzator unei adrese IP
HINFO	Descriere sistem gazdă	<b>Host Info</b> – Info ptr. calculatorul reprezentat de nod (Unitate centrală, sistem operare) în format ASCII
TXT	Text	<b>Text ASCII</b> – orice informatie utilă despre entitate



# Exemplu Resource Records

O parte a unei baze de date DNS pentru [cs.vu.nl](http://cs.vu.nl)

; Authoritative data for cs.vu.nl

cs.vu.nl.	86400	IN	SOA	star boss (9527,7200,7200,241920,86400)
cs.vu.nl.	86400	IN	MX	1 zephyr
cs.vu.nl.	86400	IN	MX	2 top
cs.vu.nl.	86400	IN	NS	star

info despre domeniu  
2 servere de mail si  
unul de nume

star	86400	IN	A	130.37.56.205
zephyr	86400	IN	A	130.37.20.10
top	86400	IN	A	130.37.20.11
www	86400	IN	CNAME	star.cs.vu.nl
ftp	86400	IN	CNAME	zephyr.cs.vu.nl

3 adrese IP si 2  
pseudonime pentru  
Web si FTP

flits	86400	IN	A	130.37.16.112
flits	86400	IN	A	192.31.231.165
flits	86400	IN	MX	1 flits
flits	86400	IN	MX	2 zephyr
flits	86400	IN	MX	3 top

secțiune ptr server  
flits cu 2 adrese si 3  
servere mail

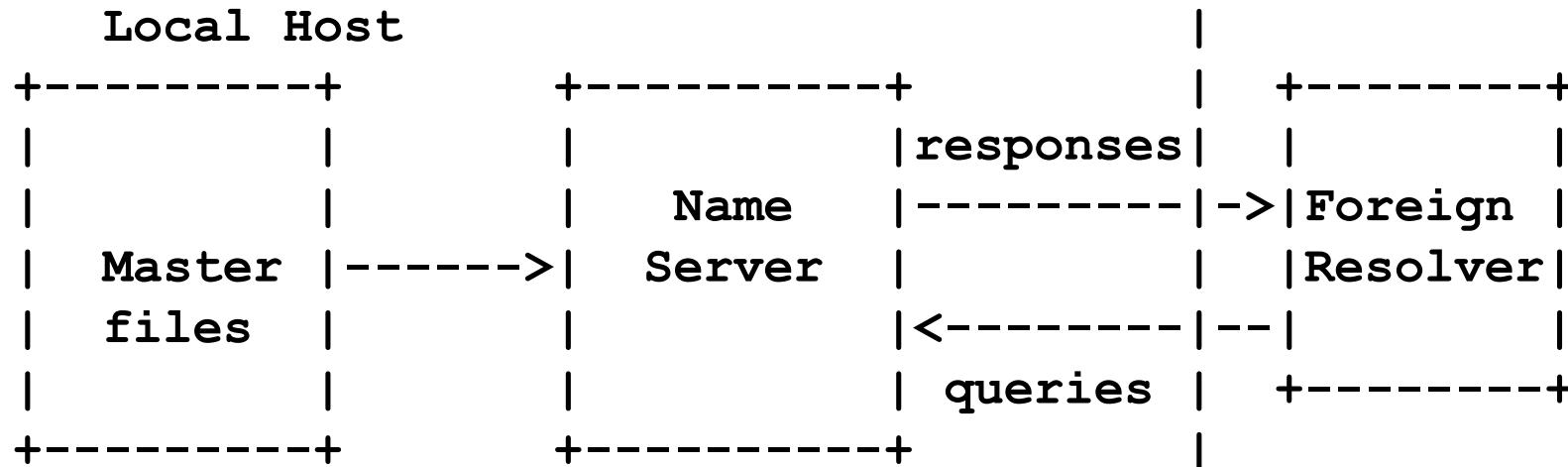
rowboat		IN	A	130.37.56.201
		IN	MX	1 rowboat
		IN	MX	2 zephyr

RRs statie de lucru  
cu 2 adrese mail

little-sister		IN	A	130.37.62.23
laserjet		IN	A	192.31.231.216



# Serverul de nume (DNS)



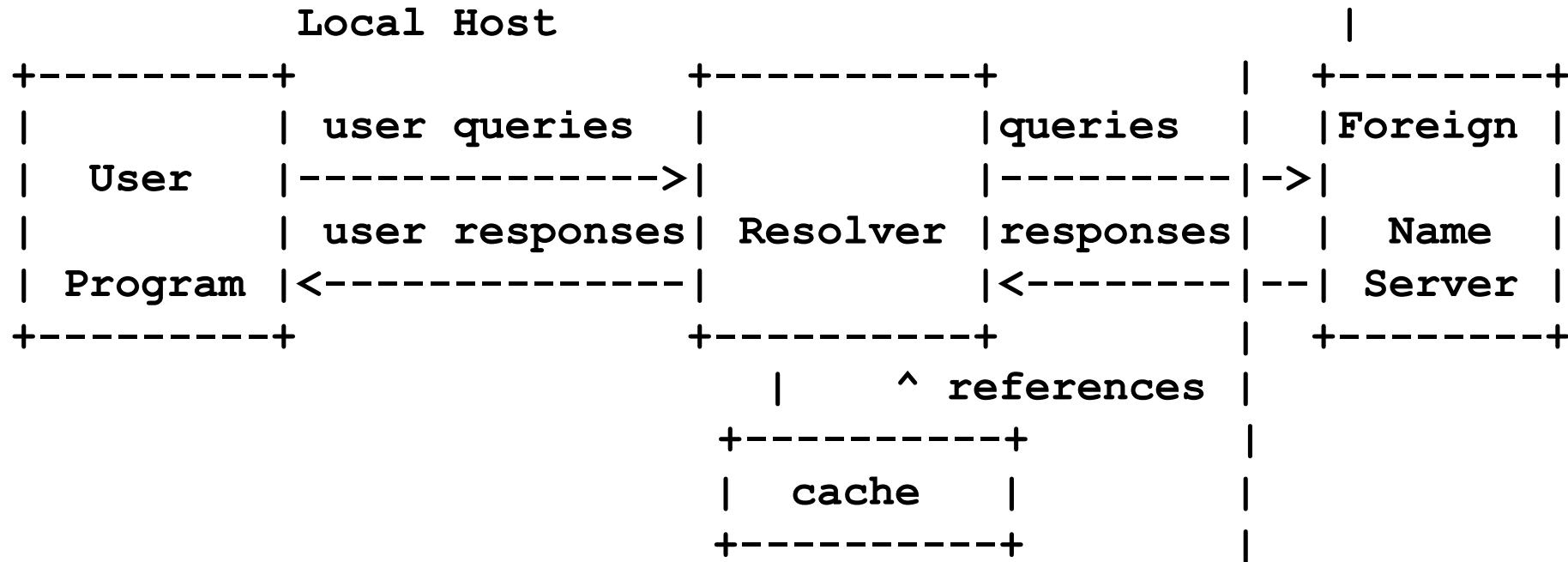
Informația **primara** despre zone este pastrată în **fisiere master** (Master files) aflate în sistemul de fisiere local serverului DNS

Un **server** de nume **primar**

- Folosește **Master files** pentru a defini sau actualiza BD pentru una sau mai multe **zone**
- Raspunde întrebarilor resolverelor



# Translatarea de la nume domeniu la adresa IP



Programul apeleaza un **Resolver** local

Formatul mesajelor **user** ↔ **resolver** este specific sistemului gazda (apeluri SO). Ex. - UNIX [gethostbyname](#)

**Resolver** apeleaza un server DNS local ([ii cunoaste adresa IP!](#))

Format mesaje **resolver** ↔ **name server** este standard (protocol DNS)

Resolver poate pastra in **cache** numele si adresele IP recent rezolvate

Perioada cache este data de [time-to-live](#) din Resource Record



# Protocolul DNS

- Software de rezolvare disponibil ca proceduri de biblioteca
  - Exemplu - UNIX `gethostbyname`
- La apelul unui *client*, Resolverul
  - Construiește un mesaj *DNS request*
  - Trimite mesajul serverului DNS local
- *Serverul* DNS rezolva numele
  - Construiește un mesaj *DNS reply*
  - Trimit mesajul Resolverului si așteaptă următoarea cerere



# Format mesaje DNS

+-----+	
	<b>Header</b>
+-----+	
	<b>Question</b>
	the question for the name server
+-----+	
	<b>Answer</b>
	RRs answering the question
+-----+	
	<b>Authority</b>
	RRs pointing toward an authority
+-----+	
	<b>Additional</b>
	RRs holding additional information
+-----+	

RR = Resource Record

**Header** contine info despre

- ce sectiuni sunt prezente in mesaj
- mesajul este **intrebare** sau **raspuns**
- sau **alta operatie** (se specifica cod operatie)



# Format mesaje DNS (2)

+-----+	
	<b>Header</b>
+-----+	
	<b>Question</b>
	the question for the name server
+-----+	
	<b>Answer</b>
	RRs answering the question
+-----+	
	<b>Authority</b>
	RRs pointing toward an authority
+-----+	
	<b>Additional</b>
	RRs holding additional information
+-----+	

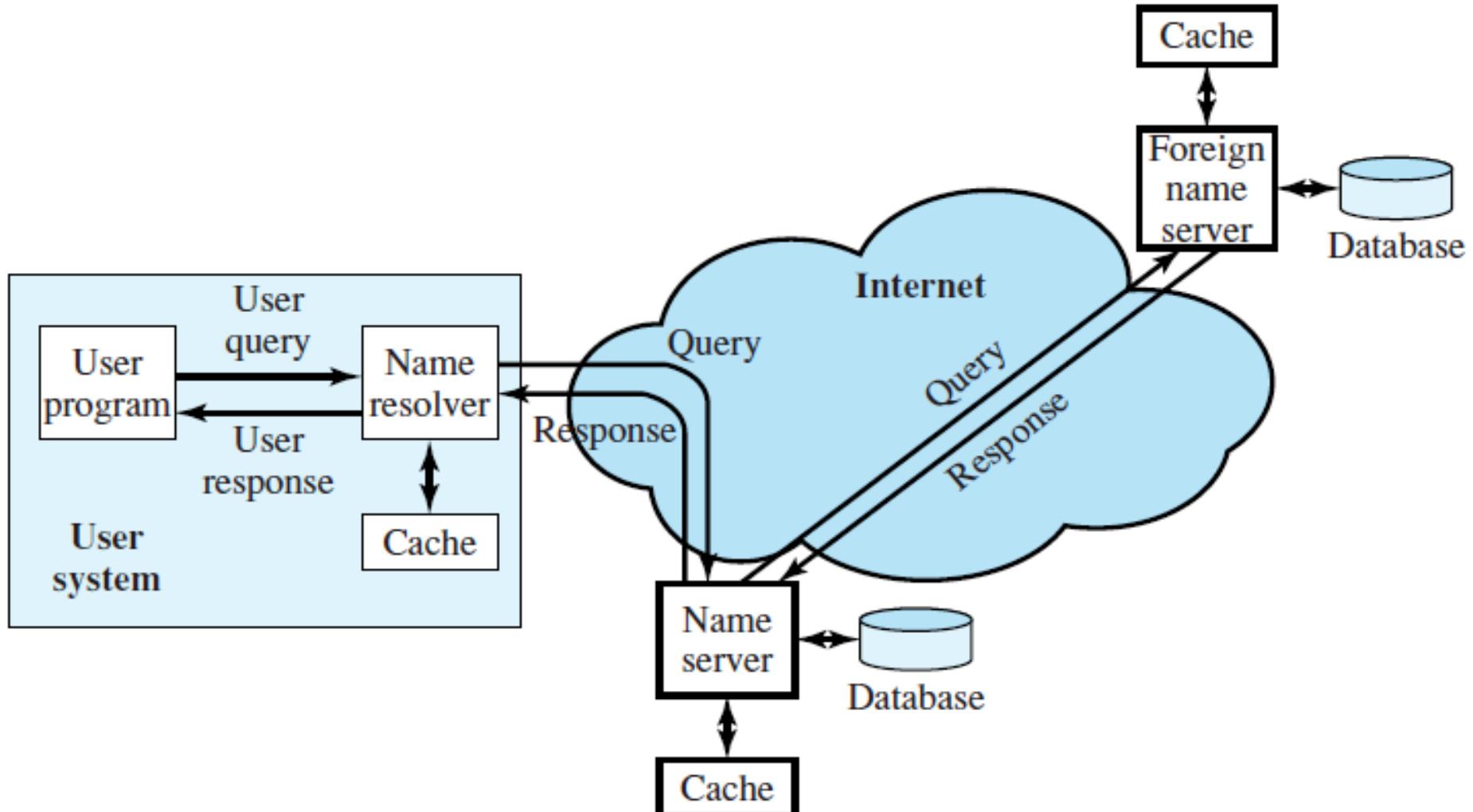
## Question – intrebarea

- tuplu Nume-domeniu, tip, clasa
- este singurul camp inclus în întrebare

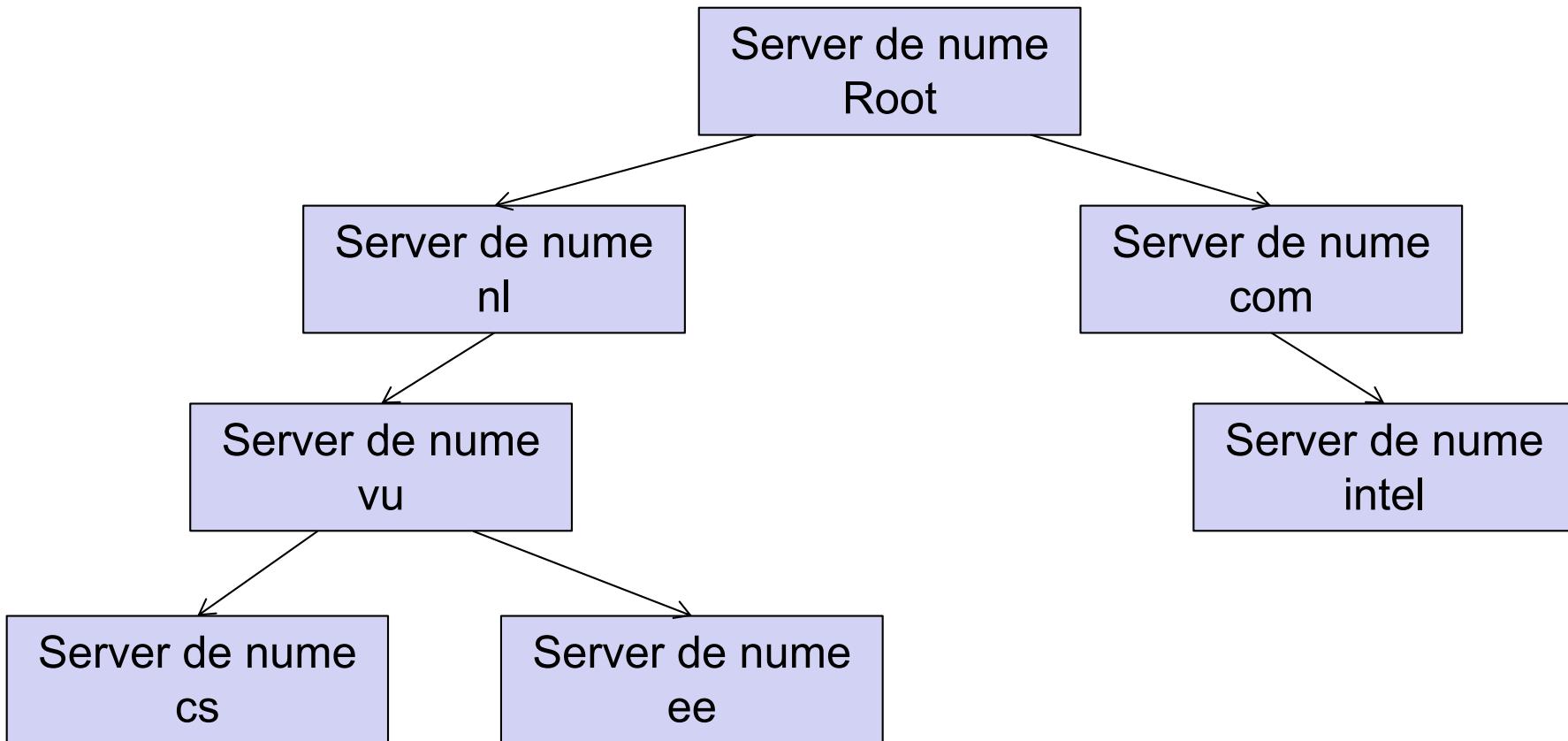
**Answer** include RR-urile care corespund întrebării

Restul – colecție de RR-uri reprezentând răspunsul, autoritatea și informațiile suplimentare

- Un server DNS este *server autoritate* ptr numele gestionate
- Daca cererea contine un nume gestionat de serverul apelat, acesta răspunde direct
- Altfel, cererea trebuie sa ajunga la serverul autoritate pentru acel nume



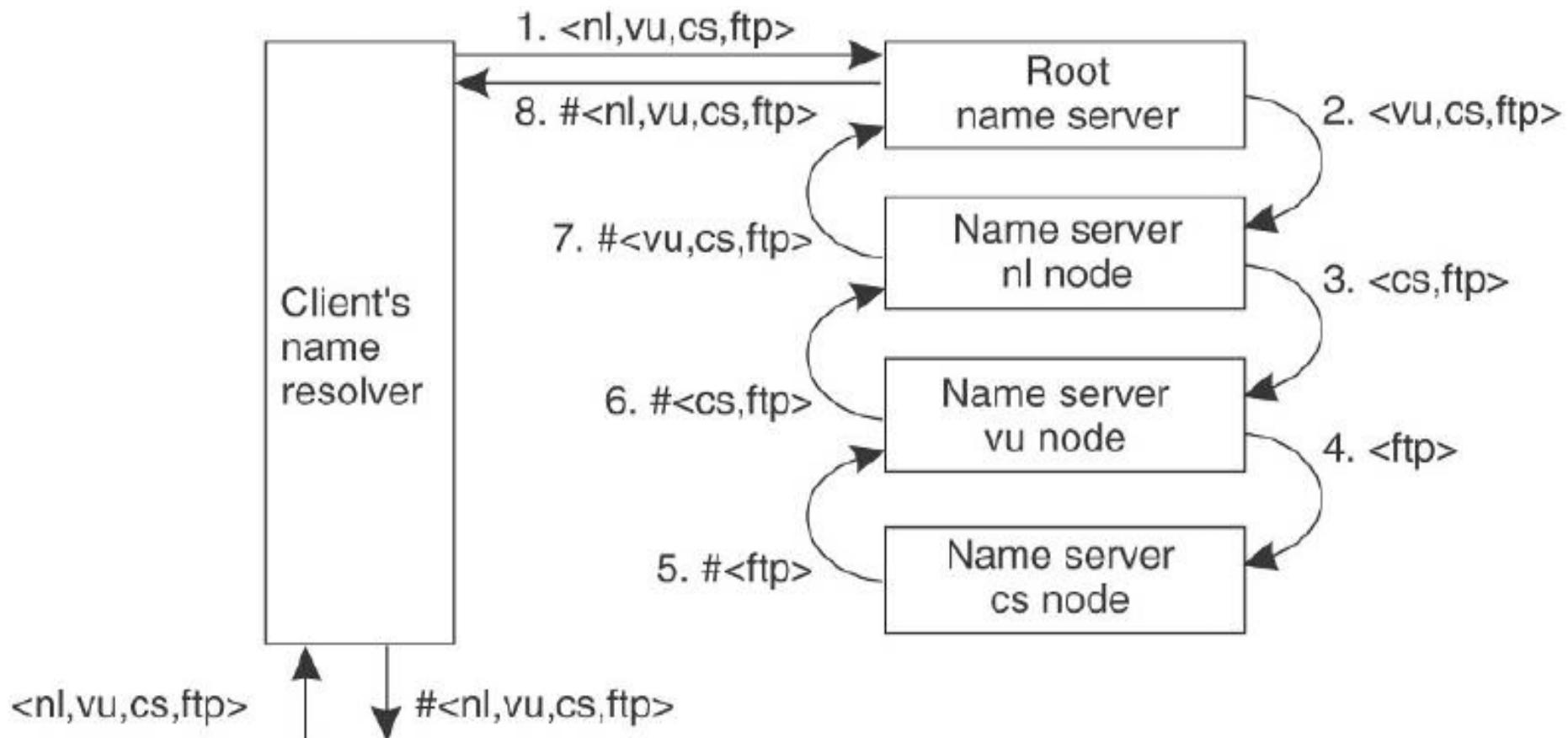
# O posibila ierarhie de servere DNS



- 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 recursivă

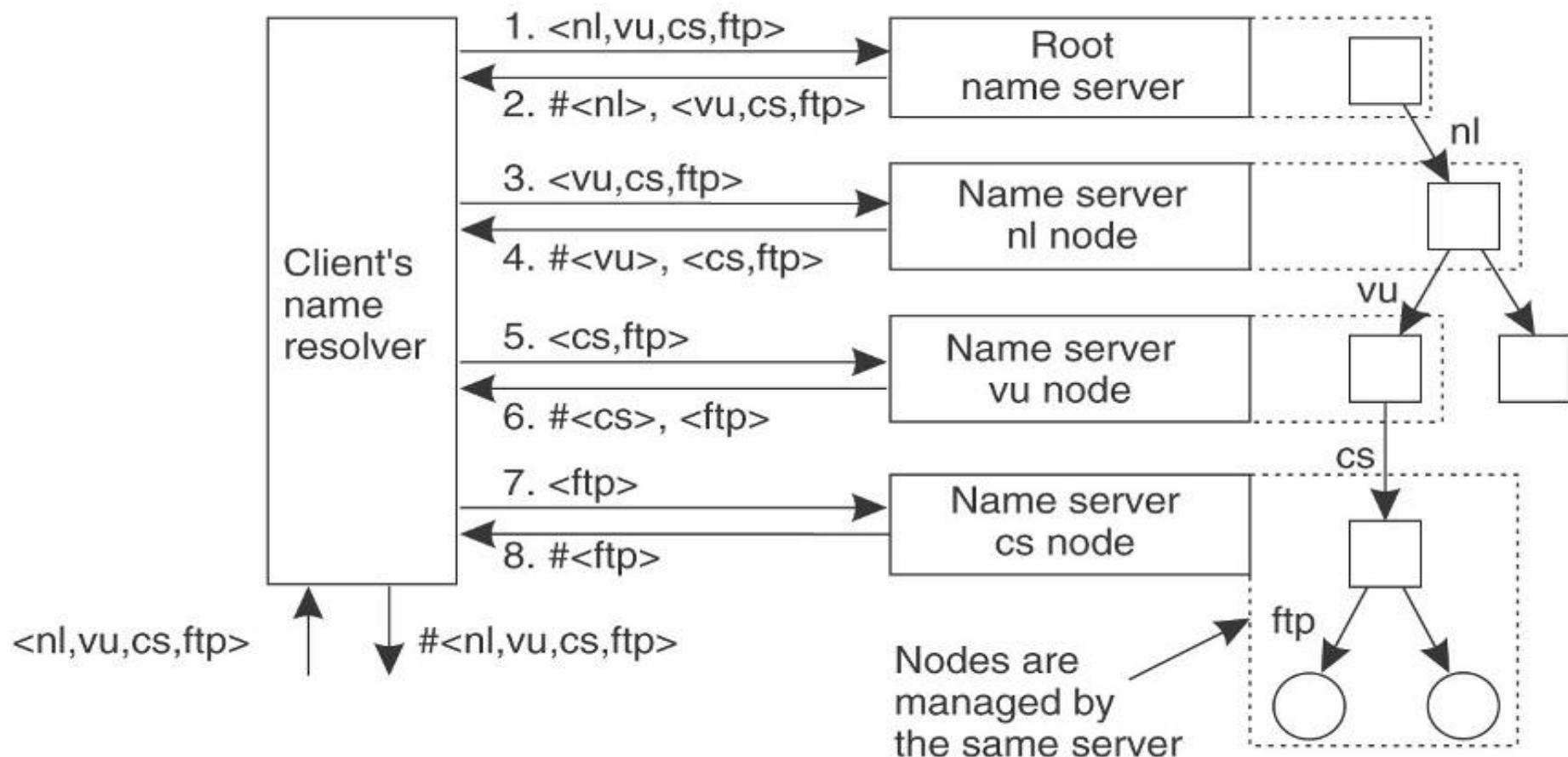
- cererea este pasata de la un server DNS la altul pana ajunge la serverul DNS care rezolva numele din cerere
- raspunsul este trimis pe calea inversa



In exemplu: `#<ftp>` este adresa IP a serverului [ftp.cs.vu.nl](http://ftp.cs.vu.nl)

# Rezolvare Iterativă

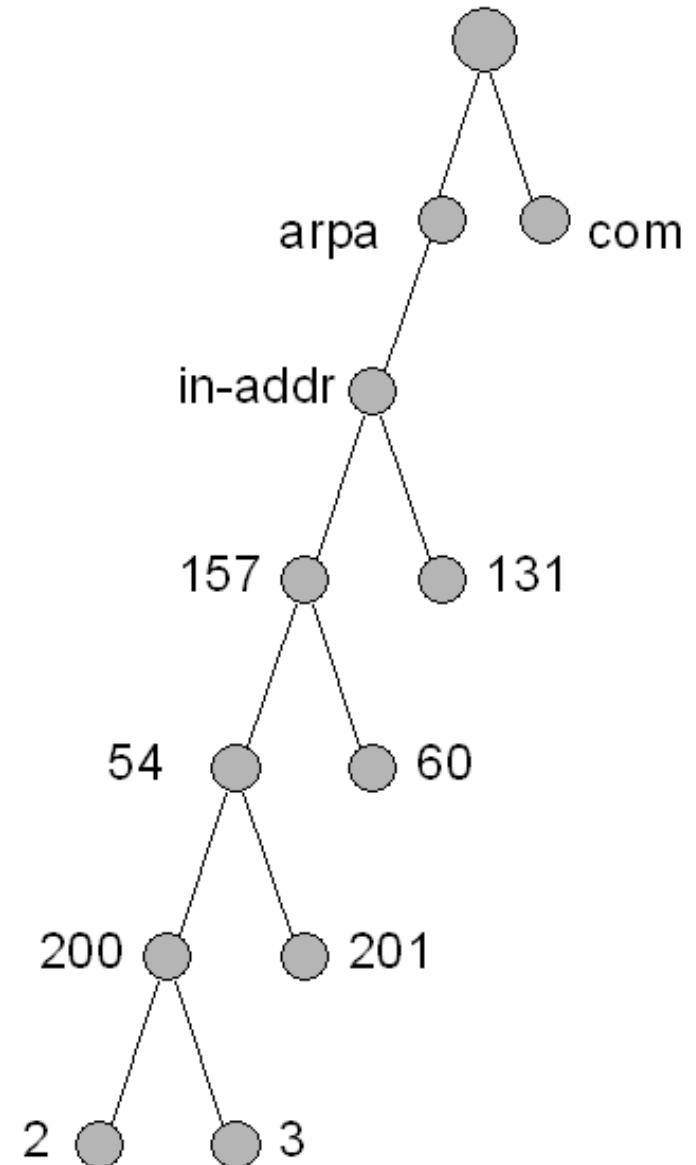
- daca serverul DNS nu poate rezolva intregul nume, el trimit clientului partea nerezolvata si adresa serverului DNS care o poate rezolva
- clientul trimit o noua cerere acestui server DNS





# Cereri inverse

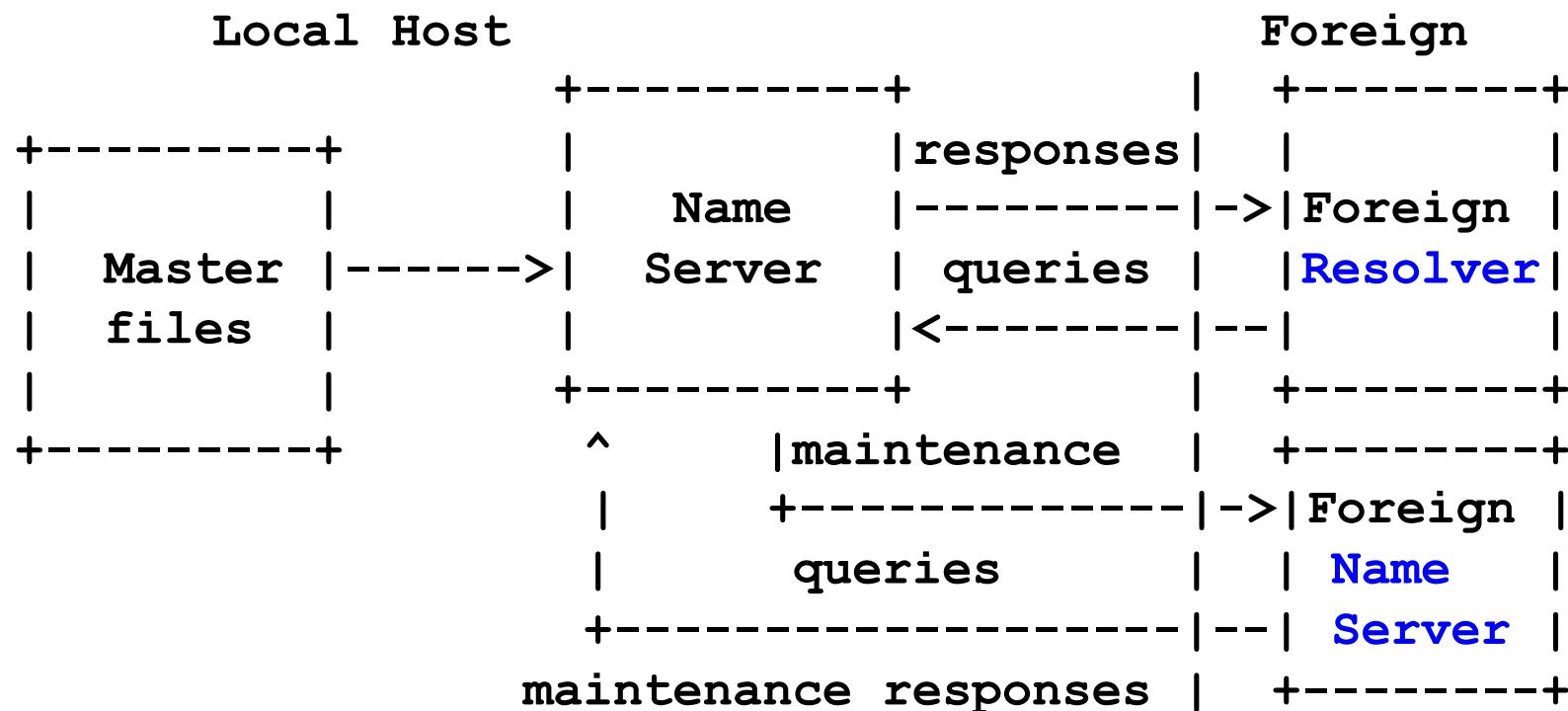
- Cauta nume pentru adresa IP 157.54.200.2
- Organizare - un domeniu special  
**in-addr.arpa**  
in care nodurile sunt numite dupa numerele din adresa IP
- In **in-addr.arpa** se creaza inregistrari PTR, in care numele sunt adrese IP
- Clientul face o cerere PTR pentru numele **2.200.54.157.in-addr.arpa**
- 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





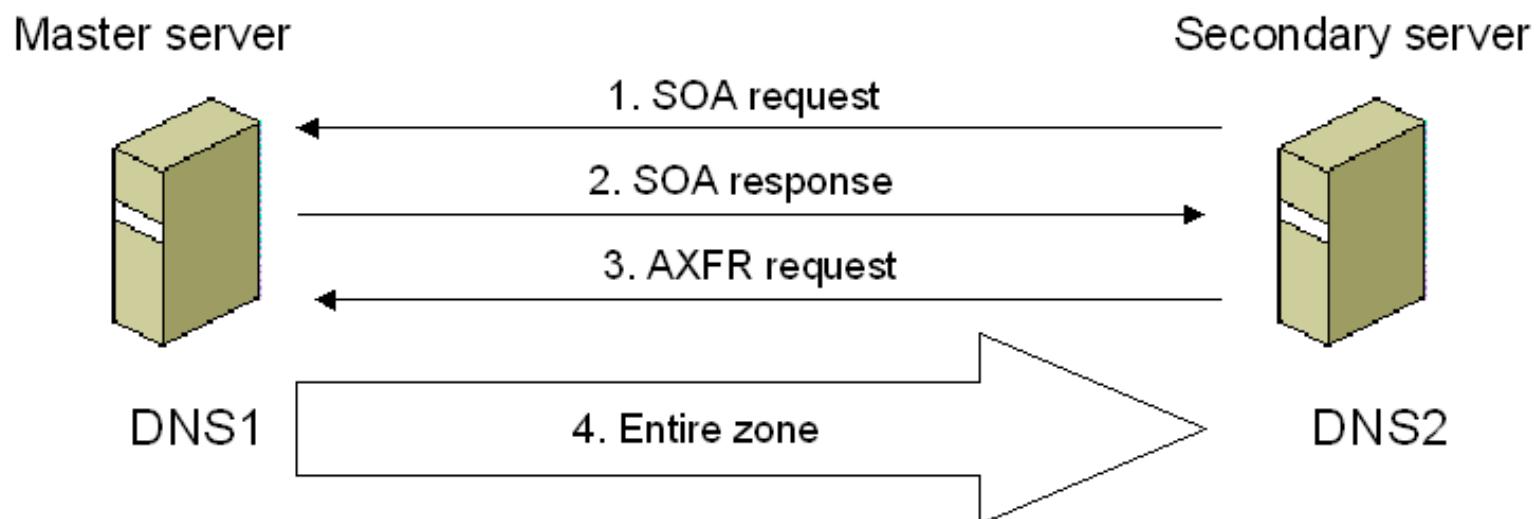
# Replicarea serverelor DNS

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



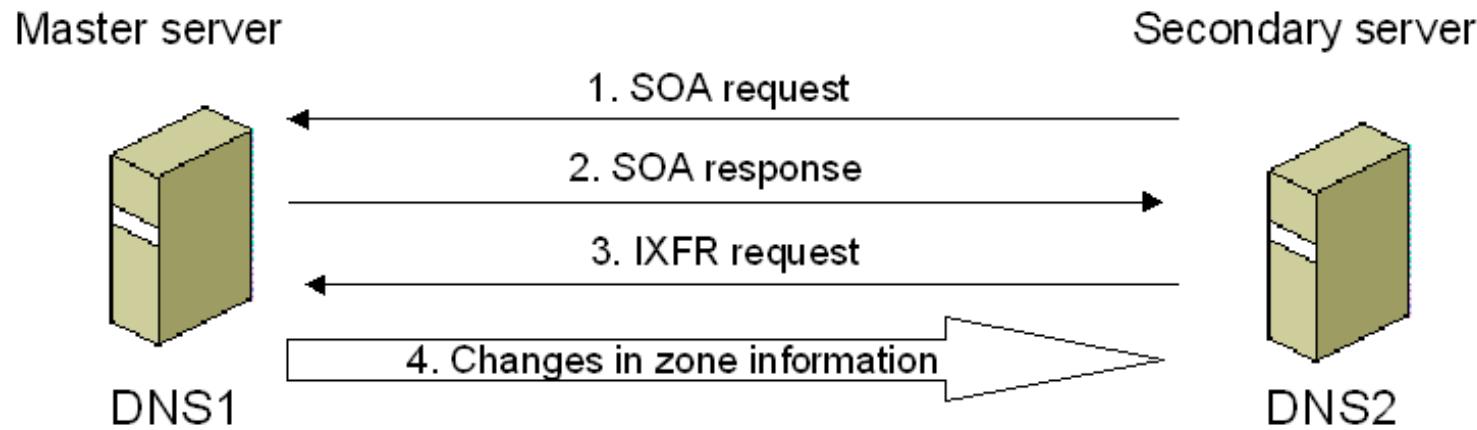
# Facilitati – transfer toata zona

- 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

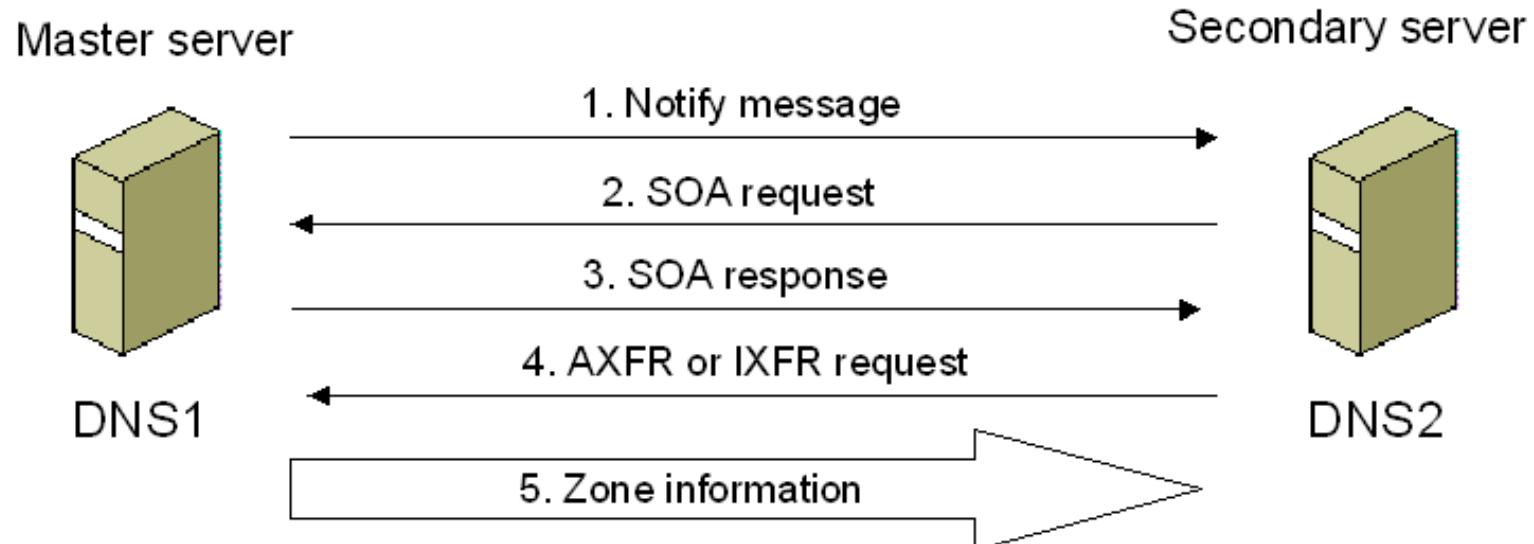




# Transfer incremental (Incremental Zone Transfer)



## Notificari





# Studiu individual

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

## 7.1 DNS - SISTEMUL NUMELOR DE DOMENII

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

## 7.1 DNS—THE DOMAIN NAME SYSTEM



# Nivelul Aplicație

World Wide Web

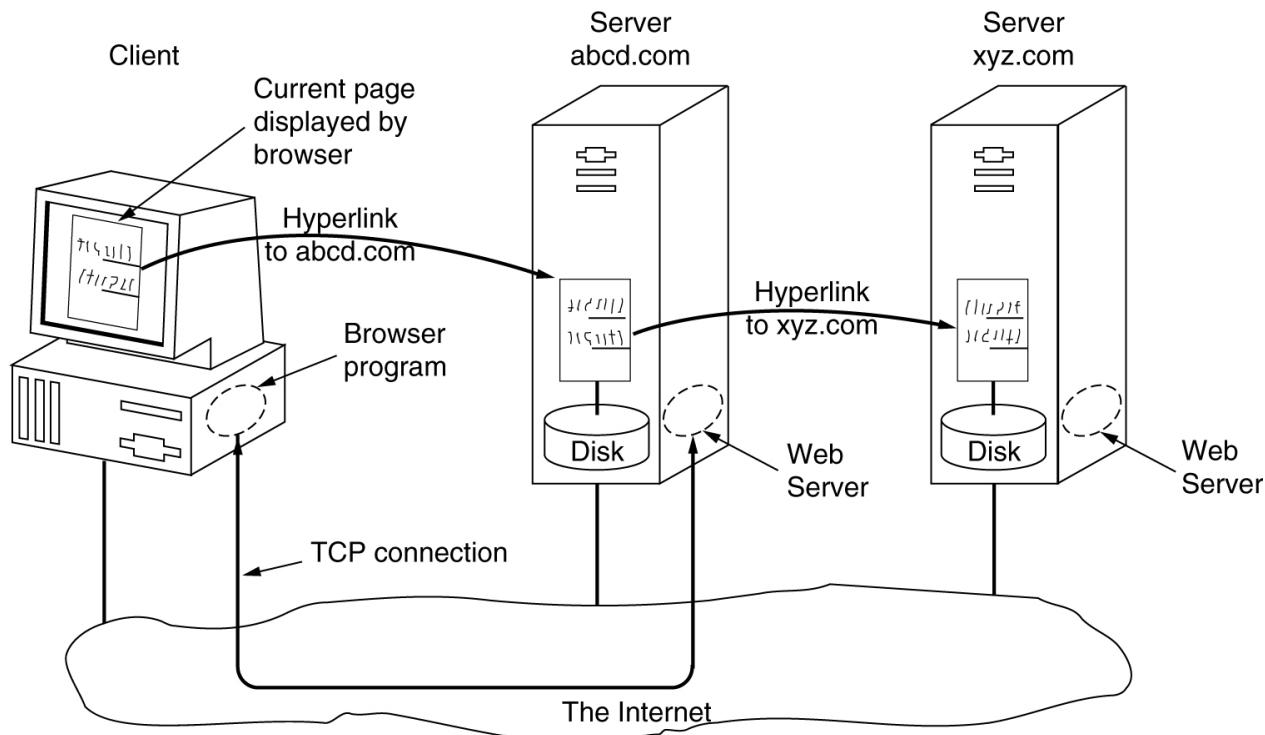


# Cuprins

- Functionarea WWW
- URL
- HTML – marcaje, formulare
- HTTP
- Clientul (Browser)
- Serverul

# World Wide Web

- Set de documente (pagini) cu legături între ele (hyperlinks)
- Distribuite pe mașini diferite
- Include o pagina de referință (home page)
  - pagina initială a unui site Web
  - pagina afisată la pornirea unui browser





# Interacțiunea client – server (reluare!)

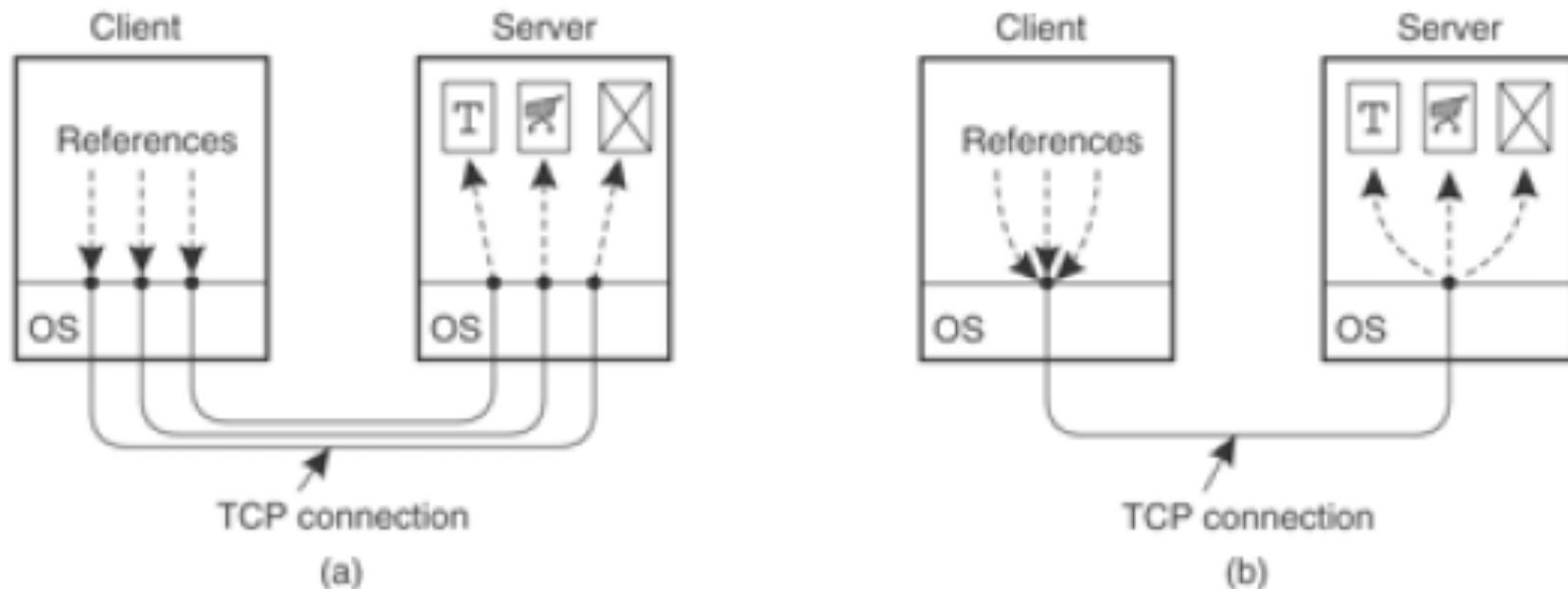
- Browser - determina URL
- Browser - cere DNS-ului adresa IP pentru www.w3.org
  - DNS - raspunde cu 18.23.0.23
- Browser - deschide o conexiune TCP la port 80 pe 18.23.0.23
- Browser - trimitte o comanda

```
GET /hypertext/www/TheProject.html
```

  - Server www.w3.org - trimitte fisierul TheProject.html
- Conexiunea TCP este inchisa
- Browser - afișează conținutul din TheProject.html
- Browser - extrage si afiseaza toate imaginile din TheProject.html (se deschide o noua conexiune TCP pentru fiecare imagine)

# Conexiuni persistente

- Introduse în HTTP 1.1
- O singură conexiune persistentă poate fi folosită pentru mai multe cereri-raspunsuri
- Cerele pot fi transmise și în “pipeline” (fără a aștepta raspunsurile)





## Trei elemente de baza

- O schema de adresare a documentelor in Internet ([URL – Uniform Resource Locator](#))
- Un limbaj de formatare a documentelor ([HTML – HyperText Markup Language](#))
- Un protocol pentru transportul mesajelor specializeze prin retea ([HTTP – HyperText Transfer Protocol](#))



# URL – Uniform Resource Locator

scheme://host[:port#]/path/.../[;url-params][?query-string][#anchor]

<b>scheme</b>	protocol (http, ftp etc.)
<b>host</b>	nume / adresa IP a serverului Web
<b>port#</b>	numar port server Web (80 pentru http)
<b>path</b>	calea de la radacina serverului la document
<b>url-params</b>	pentru identificarea sesiunii
<b>query-string</b>	valori din formular HTML
<b>anchor</b>	referinta la un marcat pozitional din document

## *exemplu*

<http://www.situlmeu.ro/cv/test?id=8079?name=valentin&x=true#aici>



# Câteva URL-uri obișnuite

Schema	Utilizat pentru	Exemple
http	Hipertext (HTML)	<a href="http://www.cs.vu.nl/~ast">http://www.cs.vu.nl/~ast</a>
ftp	FTP	<a href="ftp://ftp.cs.vu.nl/pub/minix/README">ftp://ftp.cs.vu.nl/pub/minix/README</a>
File	Fișier local	<a href="file:///usr/suzanne/prog.c">file:///usr/suzanne/prog.c</a>
news	Grup de știri	<a href="news:comp.os.minix">news:comp.os.minix</a>
news	Articol de știri	<a href="news:AA0134223112@cs.utah.edu">news:AA0134223112@cs.utah.edu</a>
gopher	Gopher	<a href="gopher://gopher.tc.umn.edu/11/libraries">gopher://gopher.tc.umn.edu/11/libraries</a>
mailto	Trimitere de poșta electronică	<a href="mailto:JohnUser@acm.org">mailto:JohnUser@acm.org</a>
telnet	Conectare la distanță	<a href="telnet://www.w3.org:80">telnet://www.w3.org:80</a>



# HTML - HyperText Markup Language

## Structura unei pagini

```
<html>
  <head>
    <title>
      Prima incercare
    </title>
  </head>
  <body>
    Prima incercare: Nu este greu sa
    construiesti un text urat in html,
    mai complicat este sa construiesti
    un text care sa arate bine.
  </body>
</html>
```

Ce afiseaza browser-ul

**Prima incercare: Nu este greu sa  
construiesti un text urat in html, mai  
complicat este sa construiesti unul care  
sa arate bine.**



# O selecție de marcaje uzuale

Marcaj	Descriere
<html> ... </html>	Delimitază textul scris în HTML
<head> ... </head>	Delimitază zona de antet
<title> ... </title>	Definește titlul (nu este afișat de programul de navigare)
<body> ... </body>	Delimitază zona de corp
<hn> ... </hn>	Delimitază un titlu de nivel <i>n</i>
<b> ... </b>	Text îngroșat
<i> ...</i>	Text cursiv
<center> ... </center>	Centrat pe orizontală
 	Trecere la linie nouă
<p>	Început de paragraf
<ul> ... </ul>	Delimitază o listă neordonată
<ol> ... </ol>	Delimitază o listă ordonată (numerotată)
<li> ... </li>	Delimitază un element într-o listă ordonată sau neordonată
<hr>	Linie orizontală
	Afișează o imagine în acest loc (sau text-ul specificat)
<a href="URL">text</a>	Hiper-legătură la o pagină
<a name="ancora">text</a>	Declară o ancoră într-un document



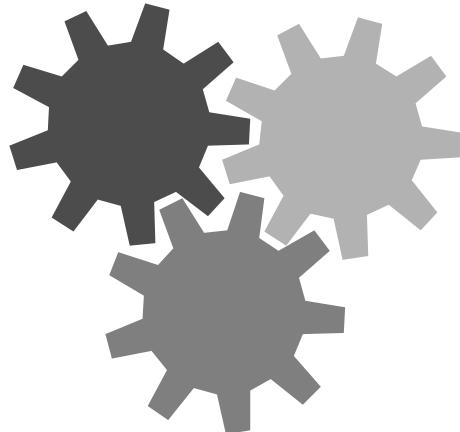
# HTML – un exemplu

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title></head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img SRC="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p> Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by FAX. </p>
<hr>
<h2> Product Information </h2>
<ul>
    <li> <a href="http://widget.com/products/big" > Big widgets </a>
    <li> <a href="http://widget.com/products/little" > Little widgets </a>
</ul>
<h2> Telephone Numbers </h2>
<ul>
    <li> 1-800-WIDGETS
    <li> 1-415-765-4321
</ul>
</body>
</html>
```



# Pagina formatată

## Welcome to AWI's Home Page



We are so happy that you have chosen to visit **Amalgamated Widget's** home page. We hope you will find all the information you need here.

Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by FAX.

---

### Product Information

- [Big widgets](#)
- [Little widgets](#)

### Telephone numbers

- 1-800-WIDGETS
- 1-415-765-4321



# Formulare – marcaje specifice

element HTML	Parametri	Semnificație
<b>&lt;INPUT&gt;, TYPE=text</b>	<b>NAME, SIZE, MAXLENGTH</b>	câmp de intrare <b>(tipul implicit)</b>
<b>&lt;INPUT&gt;, TYPE=radio</b>	<b>NAME, VALUE</b>	buton radio
<b>&lt;INPUT&gt;, TYPE=checkbox</b>	<b>NAME, CHECKED</b>	casetă de selecție
<b>&lt;INPUT&gt;, TYPE=password</b>	<b>NAME, SIZE, MAXLENGTH</b>	câmp de parolă
<b>&lt;INPUT&gt;, TYPE=reset sau submit</b>		buton de acțiune
<b>&lt;INPUT&gt;, TYPE=image</b>	<b>NAME, ALIGN, SRC</b>	hartă (imagine) activă
<b>&lt;INPUT&gt;, TYPE=hidden</b>	<b>NAME,</b>	element ascuns
<b>&lt;SELECT&gt;</b>	<b>NAME, OPTION, MULTIPLE</b>	listă de selecție
<b>&lt;TEXTAREA&gt;</b>	<b>NAME, COLS, ROWS, WRAP</b>	zonă de editare



# Formulare – un exemplu

```

<html>
<head><title> AWI CUSTOMER ORDERING FORM </title></head>
<body>
    → <h1> Widget Order Form </h1>
    <form ACTION="http://widget.com/cgi-bin/widgetorder" method=POST>
        → <p> Name <input name="customer" size=46> </p>
        → <p> Street Address <input name="address" size=40> </p>
        → <p> City <input name="city" size=20> State <input name="state" size=4> Country
            <input name="country" size=10> </p>
        → <p> Credit card # <input name="cardno" size=10> expires <input name="expires"
            size=4> M/C <input name="cc" type=radio value="mastercard"> VISA <input
            name="cc" type=radio value="visacard"> </p>
        → <p> Widget size Big <input name="product" type=radio value="expensive"> Little
            <input name="product" type=radio value="cheap"> Ship by express courier
            <input name="express" type=checkbox> </p>
        → <p> <input type=submit value="Submit order"> </p>
    Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>

```

→ **Widget Order Form**

→ Name

→ Street address

→ City  State  Country

→ Credit card #  Expires  M/C  Visa

→ Widget size Big  Little  Ship by express courier

→

Thank you for ordering an AWI widget, the best widget money can buy!



# Formulare

Un text cu informațiile complete de utilizator

## Widget Order Form

Name

Street address

City  State  Country

Credit card #  Expires  M/C  Visa

Widget size Big  Little  Ship by express courier

Thank you for ordering an AWI widget, the best widget money can buy!

customer=John+Doe&address=100+Main+St.&city=White+Plain&state=NY&country=USA&cardno=1234567890&expires6/98&cc=mastercard&product=cheap&express=on

(împărțit aici în trei linii din motive de aliniere in pagină)



# HTTP

- **Protocol “stateless”**
- **Foloseste paradigma request/response**
  - clientul și serverul comunică direct sau prin proxy-uri
  - structura mesajelor:
    - linia de comanda / răspuns
    - linii de antet
    - linie blank
    - corp mesaj

## Structura mesaj **request**

**METHOD /path-to-resource**  
**HTTP/version-number**  
**Header-name-1: value**  
**Header-name-2: value**  
**...**  
**[ optional request body ]**

## Exemplu

**GET /sj/index.html HTTP/1.1**  
**Host: www.mywebsite.com**

## Structura mesaj **response**

**HTTP/version-number status-code message**  
**Header-name-1: value**  
**Header-name-2: value**  
**...**  
**[ response body ]**

## Exemplu

**HTTP/1.1 200 OK**  
**Content-Type: text/html**  
**Content-Length: 9934**  
**...**  
**<HTML>**  
**<HEAD> ... </HEAD> ...**  
**... </HTML>**



# Metode HTTP

Metoda	Descriere
GET	Cerere de citire a unei pagini Web
HEAD	Cerere de citire a antetului unei pagini de Web
POST	Adăugarea la resursa specificată (de exemplu o pagină de Web)
PUT	Cerere de memorare a unei pagini de Web
DELETE	Ștergerea unei pagini de Web
TRACE	Transmite în ecou cererea care a sosit
OPTIONS	Interrogarea anumitor opțiuni
CONNECT	Folosit pentru conectare prin proxy server pe conexiune tunel



# Exemplu GET

- Formular HTML

```
<HTML>
```

```
<HEAD><TITLE>Formular  
simplu</TITLE></HEAD>
```

```
<BODY>
```

```
<H2>Formular simplu</H2>
```

```
<FORM ACTION="http://financiar.yahoo.com/q"  
METHOD="get">
```

Ticker: <INPUT SIZE="25" NAME="s">

<INPUT TYPE="submit" VALUE="Get Quote">

</FORM>

</BODY>

</HTML>

## Formular simplu

Ticker:

**Get Quote**

URL construit de browser pentru intrarea  
**YHOO**

**http://financiar.yahoo.com/q?**

Cerere HTTP

**GET /q?s=YHOO HTTP/1.1**

**Host: financiar.yahoo.com**

**User-Agent: Mozilla/4.75 [en]**



# Raspuns

HTTP/1.1 200 OK

Date: Sat, 03 May 2005 17:48:35 GMT

Connection: close

Content-Type: text/html

Set-Cookie: B=dfaosiu534qjnfretk&b=2; expires=Thu, 15 Aug 2011 20:00:00 GMT; path=/; domain=.yahoo.com

<HTML>

<HEAD><TITLE>Yahoo! financlar - YHOO</TITLE></HEAD>

<BODY>

...

</BODY>

</HTML>



## Exemplu POST

- Aceeasi cerere, formulata cu metoda POST

**POST /q HTTP/1.1**

**Host: finanziar.yahoo.com**

**User-Agent: Mozilla/4.75 [en] (WinNT; U)**

**Content-Type: application/x-www-form-urlencoded**

**Content-Length: 6**

**s=YHOO**

- Raspunsul este identic



# Exemplu HEAD

## Cerere

**HEAD http://www.cs.pub.ro/~ionescu/ HTTP/1.1**

**Host: www.cs.pub.ro**

**User-Agent: Mozilla/4.75 [en] (WinNT; U)**

## Raspuns

**HTTP/1.1 200 OK**

**Date: Mon, 05 Feb 2005 04:33:19 GMT**

**Server: Apache/1.2.5**

**Last-Modified: Mon, 05 Feb 2005 04:30:19 GMT**

**Content-Length: 2234**

**Content-Type: text/html**



# Coduri de stare

<b>Cod</b>	<b>Semnificație</b>	<b>Exemple</b>
1xx	Informație	100 = serverul acceptă continuarea tratării cererii de la client (asociat cu un antet Expect din cerere)
2xx	Succes	200 = cerere reușită; 204 = nu există conținut
3xx	Redirectare	301 = pagină mutată definitiv; 302 = pagina mutata temporar; 304 = pagina din memoria ascunsă este încă validă
4xx	Eroare la client	400 = cerere incorecta; 401 = ne-autorizat 403 = interzis 404 = pagina nu a fost găsită
5xx	Eroare la server	500 = eroare internă la server; 501 = ne-implementat 503 = încearcă mai târziu



# Antete Mesaje HTTP

Antet	Tip	Descriere
User-Agent	Cerere	Informație asupra programului de navigare și a platformei
Accept	Cerere	Tipul de pagini pe care clientul le poate trata
Accept-Charset	Cerere	Seturile de caractere care sunt acceptabile la client
Accept-Encoding	Cerere	Codificările de pagini pe care clientul le poate trata
Accept-Language	Cerere	Limbajele naturale pe care clientul le poate trata
Host	Cerere	Numele DNS al serverului (folosit pentru virtual hosting)
Authorization	Cerere	O listă a drepturilor clientului
Cookie	Cerere	Trimite (la server) un cookie setat anterior
Set-Cookie	Răspuns	Serverul vrea să salveze un cookie la client
Server	Răspuns	Informație despre server (ex. Server: Apache/1.2.5)



# Antete Mesaje HTTP (2)

Antet	Tip	Descriere
<b>Content-Encoding</b>	Răspuns	Cum este codat conținutului (de exemplu, gzip)
<b>Content-Length</b>	Răspuns	Lungimea paginii în octeți
<b>Content-Type</b>	Răspuns	Tipul/subtipul MIME al paginii
<b>Last-Modified</b>	Răspuns	Ora și data la care pagina a fost ultima dată modificată
<b>Location</b>	Răspuns	O indicatie pentru client pentru redirectarea cererii
<b>Accept-Ranges</b>	Răspuns	Serverul va accepta cereri în anumite limite de octeți
<b>Date</b>	Ambele	Data și ora la care mesajul a fost trimis
<b>Connection</b>	Ambele	Intentia de a pastra sau nu conexiunea (ex. <b>Connection: Close</b> )



# Antete referitoare la tipul continutului

- Sistem de tipuri imprumutat din MIME (Multipurpose Internet Mail Extensions)
- Doua niveluri (reprezentate de doua antete in raspuns)
  - Content-Encoding
    - gzip (GNU zip)
    - compress (UNIX)
    - deflate (zlib format definit in RFC 1950 si 1951)
  - Content-Type
    - Tip, subtip si (optional) perechi *atribut = valoare*
    - *Exemple*  
`Content-Type: text/plain; charset = 'us-ascii'`  
`Content-Type: text/xml`  
`Content-Type: application/pdf`  
`Content-Type: video/x-mpeg`



# Exemplu mesaje multipart

## Cerere

```
GET /cgi-bin/doit.cgi HTTP/1.1  
Host: cgi-bin.netscape.com  
Date: Sun, 18 Feb 2004 06:22:33 GMT
```

## Raspuns

```
HTTP/1.1 200 OK  
Server: Netscape-Enterprise-3.6 SP1  
Date: Sun, 18 Feb 2004 06:22:35 GMT  
Content-Type: multipart/x-mixed-replace;  
boundary="ThisRandomString"
```

```
--ThisRandomString  
Content-Type: image/gif
```

```
...
```

```
--ThisRandomString  
Content-Type: image/gif
```

```
...
```

```
--ThisRandomString  
Content-Type: image/gif
```

```
...
```



# Antete pentru control caching

Trei tipuri de caching:

- la client – cache privat
- la proxy, server – cache-uri partajate

Control caching – introdus in HTTP/1.1

- se face de server prin antet **Cache-Control** cu valorile
  - **public** - nici o restrictie pentru caching
  - **private** – nu in *shared caches*
  - **no-cache** – nici in browser, nici in proxy

Exemplu

**HTTP/1.1 200 OK**

**Date: Mon, 05 Feb 2005 04:33:19 GMT**

**Server: Apache/1.2.5**

**Last-Modified: Mon, 05 Feb 2005 04:30:28 GMT**

**Cache-Control: private**

**Content-Length: 2289**

...



# Consistenta cache-urilor (1)

- Asigura ca documentul din cache este acelasi cu cel din server
- **Solutie 1:** Folosind comanda **HEAD**
  - clientul transmite **HEAD**
  - primeste raspuns si verifica antet **Last-Modified**
  - transmite **GET** daca document din server este mai nou decat copia din cache
- Cerere

**HEAD** `http://www.cs.pub.ro/~ionescu/` **HTTP/1.1**

**Host:** `www.cs.pub.ro`

**User-Agent:** `Mozilla/4.75 [en] (WinNT; U)`

- Raspuns

**HTTP/1.1 200 OK**

**Date:** `Mon, 05 Feb 2005 04:33:19 GMT`

**Server:** `Apache/1.2.5`

**Last-Modified:** `Mon, 05 Feb 2005 04:30:19 GMT`

• • •



## Consistenta cache-urilor (2)

- **Solutie 2:** Folosind comanda GET cu antet **If-Modified-Since**

**GET /~ionescu/ HTTP/1.1**

**Host: www.cs.pub.ro**

**If-Modified-Since: Mon, 04 Feb 2005 04:30:28 GMT**

- **serverul transmite**

**HTTP/1.1 304 Not Modified**

**Date: Mon, 05 Feb 2005 04:33:19 GMT**

**Server: Apache/1.2.5**

- **sau**

**HTTP/1.1 200 OK**

**. . .**

**Last-Modified: Mon, 05 Feb 2005 04:30:28 GMT**

**Content-Length: 2289**



# Solutie pentru performanta

- Clientul nu contacteaza serverul pentru orice cerere
  - Raspunsul unui server poate include **data expirarii**, care este memorata de client

HTTP/1.1 200 OK

Date: Mon, 05 Feb 2005 04:33:20 GMT

. . .

Cache-Control: private

Expires: Tue, 06 Feb 2005 04:33:20 GMT

Last-Modified: Mon, 05 Feb 2005 04:33:18 GMT

- Clientul verifica existenta paginii in cache
  - **Nu exista** – cere resursa neconditionat
  - **Există expirata** - adauga la cerere antet If-Modified-Since
    - daca server raspunde cu **304 Not Modified** foloseste intrarea din cache
  - **Există ne-expirata** – foloseste intrarea din cache



# Autentificare si autorizare

- Autentificare de baza
  - permite accesul la pagini protejate
  - prin antet de autorizare
  - nume si parola transmise codat Base64 (nu criptat)
    - atentie, trebuie folosit HTTPS
- Secvența de actiuni
  - Clientul cere resursa restrictionata
  - Server raspunde cu 401

**HTTP/1.1 401 Authenticate**

**Date:** Mon, 05 Feb 2005 04:33:19 GMT

**Server:** Apache/1.2.5

**WWW-Authenticate:** Basic realm="Capitol3"

– **realm** defineste domeniul protejat



## Autentificare si autorizare (2)

- Browser retrimite cererea cu antet suplimentar de autorizare

`GET /carte/capitol3/index.html HTTP/1.1`

`Date: Mon, 05 Feb 2005 04:33:20 GMT`

`Host: www.cs.pub.ro`

`Authorization: Basic eNCoDED-userID:PaSSwoRd`

- Server verifica credentialele de autorizare si satisface cererea (sau refuza cu 403)
- Odata trimise credentialele, browserul retrimite automat antetul de autorizare si credentiale in viitoarele cereri la URL dependente (fisiere din subdirectoare)

Ex. **`http://cs.pub.ro/~popescu/clase/`**

deinde de **`http://cs.pub.ro/~popescu/`**

care este **un prefix** al primului



# Suport sesiune

**Cookie** este un mecanism ce permite transmiterea unor informații de stare prin mesaje HTTP

- ex. info stare - identificatorul unei sesiuni
- Intelegerea este inițiată de server prin antet Set-Cookie

Set-Cookie: <nume>=<valoare>[; expires=<data>]

[;path=<cale>] [;domain=<nume\_domeniu>][; secure]

<nume>=<valoare> pereche atribut/valoare de trimis înapoi de browser

**path, domain** identifică cererile care sunt calificate

- pentru domeniul .pub.edu domeniile **calificate** au forma \*.pub.edu
- pentru calea /test/ caile **calificate** sunt de forma /test/\*

**secure** browser-ul trebuie să transmită info pe legătura securizată



## Suport sesiune (2)

- Intelegerarea este acceptata de client prin **antet Cookie**

Cookie: <nume>=<valoare>

inclus de browser in cererile referitoare la URL in care **domeniul** si / sau **calea** sunt calificate

Exemplu:

**HTTP/1.1 200 OK**

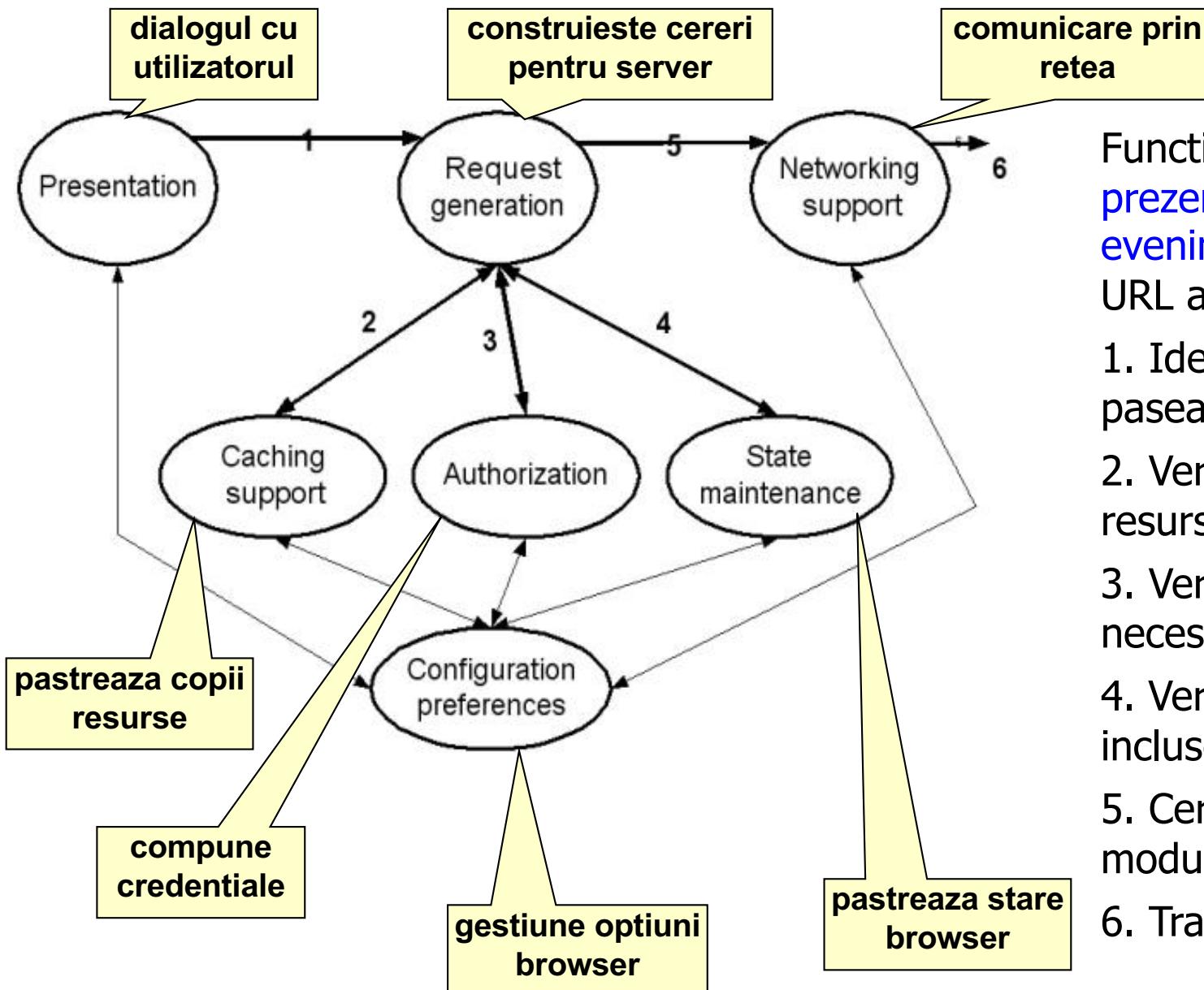
**Set-Cookie: client=Ion; path=/carte/capitol3/; domain=.pub.edu**

**GET /carte/capitol3/index.html HTTP/1.1**

**Host: www.cs.pub.edu**

**Cookie: client=Ion**

# Schema browser - Generarea unei cereri



Functionarea **modul prezentare** este bazata pe evenimente (ex. selectie URL afisat)

1. Identifica evenimentul si paseaza legatura (URL)
2. Verifica daca o copie a resursei este in cache
3. Verifica daca sunt necesare credentiale
4. Verifica daca trebuie incluse antete Cookie
5. Cererea este pasata modulului de retea
6. Transmite prin retea



# Functiile modulelor din browser

## Interfața utilizator

- Afiseaza pe ecran rezultatul primit de la modulul Interpretare continuu (din grupul de procesare a răspunsului)
- Executa acțiunile initiate de utilizator, prin meniu, taste speciale etc.
  - Selectare/introducere URL
  - Completare formulare
  - Activare butoane de navigare (ex. Back)
  - Vizualizare sursa paginii, info resurse etc.
  - Setare opțiuni configurare
    - Nu descarca imagini referite în pagina HTML
    - Rejecteaza cookies
- Paseaza informația de cerere la Generator cereri



# Generator cereri

- Primeste legatura (URL) la pagina care va fi ceruta
- URL poate fi:
  - **absolut** (ex. introdus manual) – este complet <http://domeniu/cale> → nu trebuie prelucrat
  - **relativ** (ex. preluat din pagina curenta de la modulul Interpretare continut) → trebuie rezovat!

Sunt **două cazuri**:

(1) - URL **relativ** la **directorul curent** al paginii afisate (calea din HREF **nu incepe cu /**)

**Ex:**

URL curent: <http://www.myserver.com/mydirectory/index.html>

Link în pagina: <A HREF ="**altdirector/pag2.html**">...</A>

Rezolvat la: <http://www.myserver.com/mydirectory/altdirector/pag2.html>



## Generator cereri (2)

(2) - URL **relativ** la directorul **radacina** al serverului Web cu numele din URL-ul paginii curente (calea din HREF **incepe cu /**)

**Ex:**

URL curent: <http://www.myserver.com/mydirectory/index.html>

Link in pagina: <A HREF ="/rootdirector/homepage.html">...</A>

Rezolvat la: <http://www.myserver.com/rootdirector/homepage.html>



- Construiește linia de cerere, care are 3 componente

## METODA

Implicit (la activare hyperlink) GET

În formular (specificat explicit) GET sau POST

## /cale-resursa

Numai calea în HTTP/1.1

Tot URL în HTTP/1.0

## HTTP-versiune



## – Construiește antetele de baza

Host: [www.cs.pub.ro](http://www.cs.pub.ro)

User-Agent: Mozilla/4.75 [en] (WinNT; U)

Referer: <http://www.cs.pub.ro/~ionescu/index.html>

Accept: text/html, text/plain, type/subtype

Accept-Charset: ISO-8859-1

...

Content-Type: mime-type/mime-subtype

Content-Length: xxx

Date:

Referer – pagina în care se află link-ul activat de utilizator



- Intreaba **Suport caching** daca exista intrare in cache
  - Nu exista – cere resursa neconditionat
  - Exista expirata - adauga la cerere antet **If-Modified-Since**
    - daca server raspunde cu **304 Not Modified**
      - » paseaza intrarea din cache la **Interpretare continut**
  - Exista ne-expirata – intoarce intrarea din cache
- Intreaba **Autorizare** daca e nevoie de autorizare pentru **domain/path**
  - Exista credentiale – adauga antet **Authorization**
- Intreaba **Management stare** despre cookies (**domain/path**)
  - Da – adauga antet **Cookie**
- Construieste corp cerere (vezi slide-uri urmatoare)
- Paseaza intreaga cerere la **Suport retea**
- Preferintele utilizatorului (**Configurare**) pot modifica fluxul cererii
  - nu se cer imaginile referite in pagina
  - nu se includ Cookies



- Construiește corp cerere
  - se aplică pentru POST, PUT
  - POST
    - parametrii din formulare în corp comanda

**Content-Type: application/x-www-form-urlencoded**

**Content-Length: 6**

**s=YHOO**



- PUT sau POST
  - folosind MIME

**Content-Type: multipart/multipart\_subtype;  
boundary="ThisRandomString"**

**--ThisRandomString**

**Content-Type: tip/subtip partea 1**

**Content-Transfer-Encoding: schema codificare partea 1**

**continut partea 1**

**--ThisRandomString**

**Content-Type: tip/subtip partea 2**

**Content-Transfer-Encoding: schema codificare partea 2**

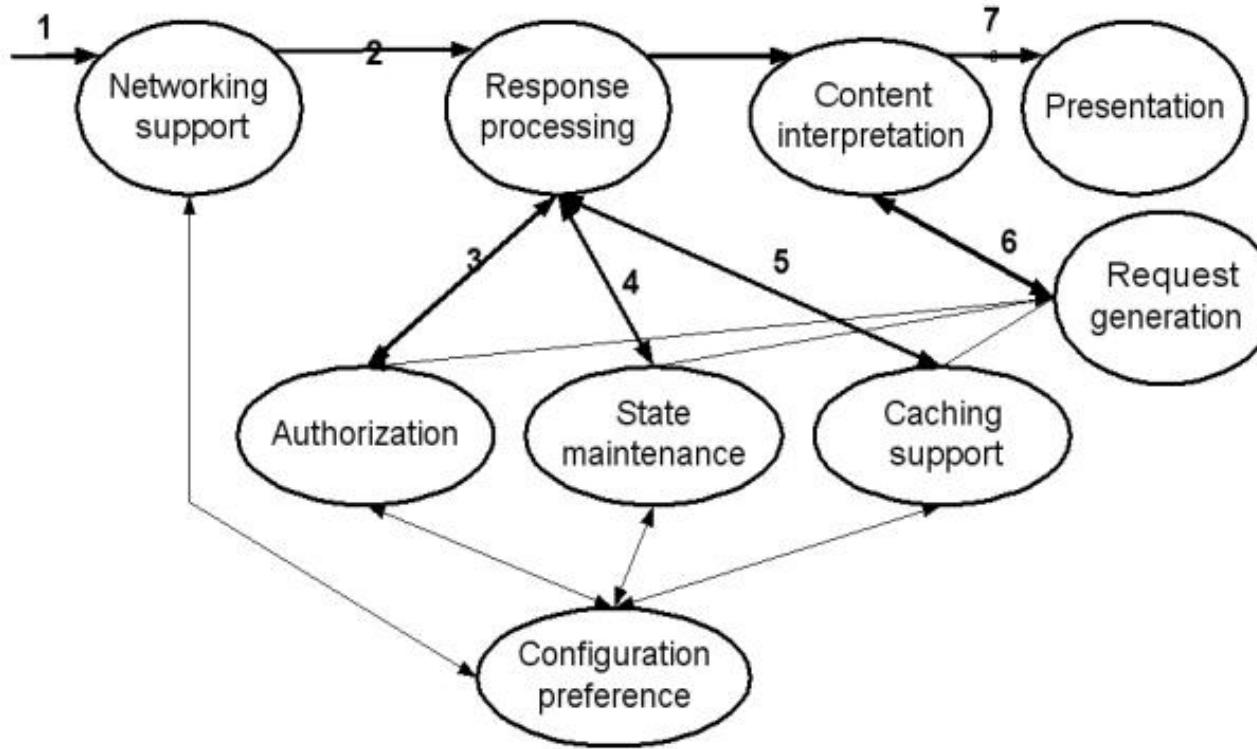
**continut partea 2**



# Suport retea

- **Trimite cererea**
  - Primeste cereri de la **Generator cereri** si le pune in coada transmisie
  - Intreaba **Configurare** ptr a determina daca tinta este un proxy si alte optiuni retea
  - Deschide socket pentru a trimite cereri din coada
    - trimite mai multe cereri la o conectare
- **Trateaza raspuns**
  - Asteapta raspunsuri la cereri
  - Paseaza la **Procesare raspuns**

# Procesarea raspunsului



1. Primeste raspuns
2. Paseaza raspuns la modulul de procesare
3. Cererea a fost **rejectata**  
– verifica daca pot fi folosite **credentiale**  
– verifica **redirectare**
4. Daca se cere info **cookie**, contacteaza modulul management stare

5. Contacteaza **suport caching** pentru memorarea raspunsului;  
apoi paseaza raspuns la **interpretare continut**
6. **Decodifica** corp raspuns, **proceseaza** diferite tipuri MIME si **parseaza** continut ptr  
eventuale referinte la resurse aditionale
7. Continut pasat la modul prezentare



# Procesare raspuns

- Verifica stare 401 (ne-autorizat)
  - Cere modulului de **Autorizare** credentiale ptr domeniul din antet **WWW-Authenticate**
    - Exista – retransmite cerere cu credentiale adaugate
    - Nu – cere credentiale de la utilizator (prin **Interfata utilizator**) si retransmite
    - Credentialele sunt memorate pe durata unei sesiuni



## – Verifica stare redirectare (301/302/307)

- Daca

**HTTP/1.1 301 Moved Permanently**

**Location:** <http://www.alta-locatie.com/pagina.html>

- Retransmite cerere la URL din antet **Location**

**GET /pagina.html HTTP/1.1**

**Host:** [www.alta-locatie.com](http://www.alta-locatie.com)

- Daca 301, memoreaza in *persistent lookup table* pentru redirectare automata a cererilor urmatoare
- Daca 302 (pagina mutata temporar) - nu memoreaza

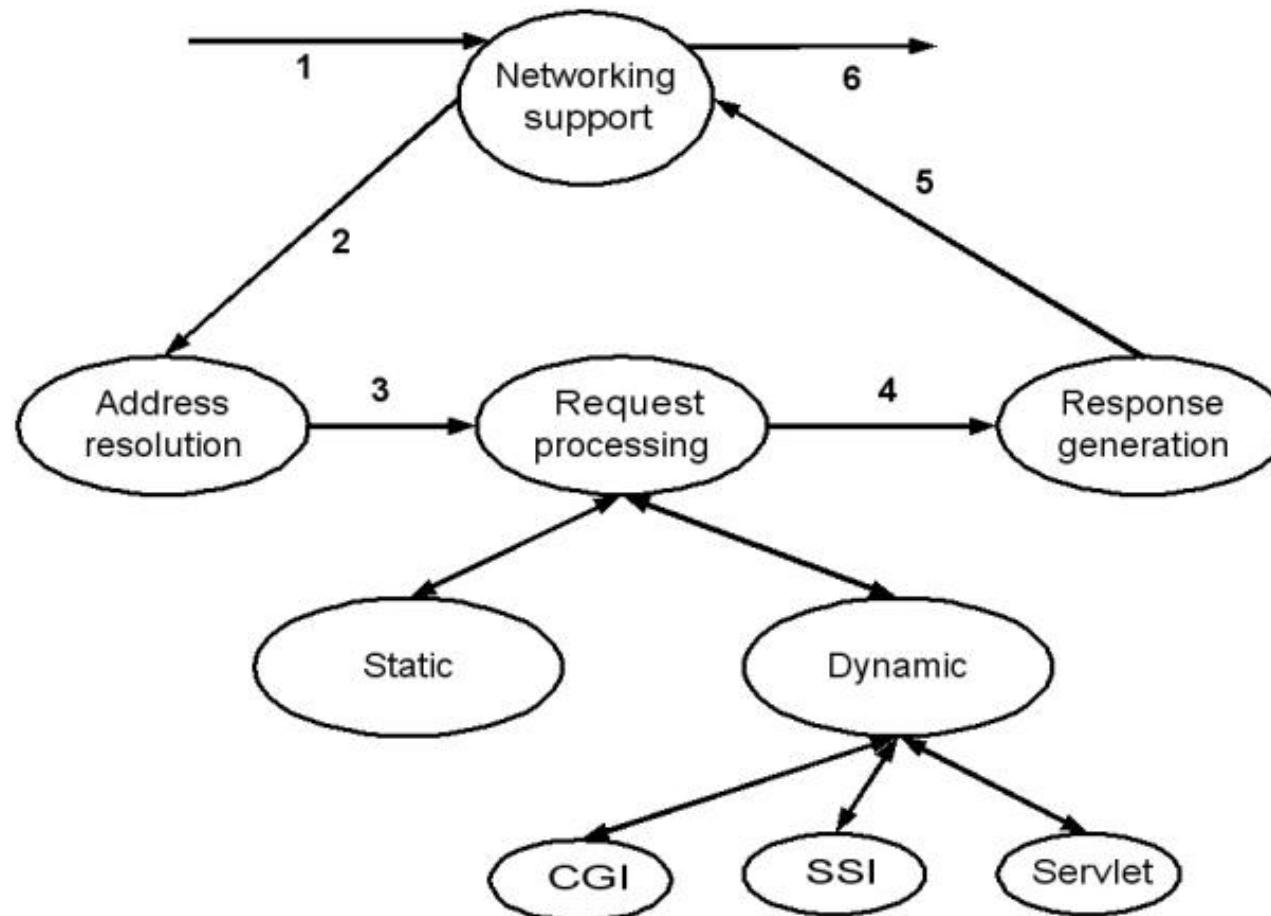


- Verifica antet **Set-Cookie**
  - Cere Management stare sa memoreze cookie in browser
  - Memorarea: pe sesiune / pentru o durata specificata
- Verifica **optiuni caching** si transmite cerere la **Suport caching** pentru a memora resursele obtinute
  - raspunsul poate include data expirarii  
**HTTP/1.1 200 OK**  
• • •  
**Cache-Control: private**  
**Expires: Tue, 06 Feb 2005 04:33:20 GMT**  
**Last-Modified: Mon, 05 Feb 2005 04:33:18 GMT**  
...
- Paseaza rezultat la **Interpretare continut**



- Interpretare continut
  - Primeste continut de la Procesare raspuns (Uneori de la Suport caching)
  - Examineaza antete codificare si, daca e cazul, decodifica continut
    - Content-Transfer-Encoding: chunked
    - Content-Encoding: compress | gzip
  - Paseaza continut decodificat la module specifice tipului MIME pe baza antet Content-Type
  - Daca link-uri la alte resurse, paseaza URL la Generator cereri
  - Paseaza fiecare modul prelucrat la Interfata utilizator
- Configurare
  - Foloseste Interfata utilizator pentru setari preferinte
  - Primeste cereri de la alte module, pentru a determina actiunile in functie de preferintele utilizatorilor

# Operații Server



4. Rezultat pasat generatorului de răspuns
5. Pasat modulului suport rețea
6. Transmite clientului

1. serverul primește o cerere
2. Paseaza la modulul de rezolutie a adresei care
  - (a) determina **server-ul**;
  - (b) determina daca cerere continut **static / dinamic**;
  - (c) examineaza **credentiale** autorizare.
3. Paseaza la modul procesare cerere, care apeleaza sub-module necesare



## Rezolvarea adresei

- selecteaza **virtual host**
  - nu există antet Host: -> eroare 400 Bad request
  - există -> determină domeniul
    - > determină parametrii de config. logica (proprietăți virtual host)

**Obs.** fragment din fisier configurare Apache

<VirtualHost www.ceva.com>

ServerAdmin	webmaster@calculatoare.com	
Alias	/test	/servlet/test
Alias	/images	/static/images
DocumentRoot	/www/docs/ceva	
ServerName	www.ceva.com	
ErrorLog	logs/ceva-error-log	
CustomLog	logs/ceva-access-log common	

</VirtualHost>



- rezolva alias-uri folosind info de configurare logica

Alias                    /test                    /servlet/test

Alias                    /images                  /static/images

– **<http://www.ceva.com/test?a=1&b=2>**

**/test -> /servlet/test**

– **<http://www.ceva.com/images/nou.gif>**

**/images -> /static/images**



- **mapeaza adresa**
  - **pagina statică** – adauga calea la radacina serverului la calea din URL
    - URL **http://www.ceva.com/pagini/cucu.html**
    - in configurare **DocumentRoot /www/docs/ceva**
      - calea devine → **/www/docs/ceva/pagini/cucu.html**
  - **pagina dinamica** – este creata de un program
    - mecanismul folosit se determina pe baza:
      - **prefix cale URL** **/servlet/** target = servlet Java
      - **sufix nume** **.cgi** **.php**
- **verifica autentificare**
  - cod eroare daca resursa ceruta este protejata



- **Procesare cerere**

- regaseste continut (sau primeste de la programul care-l genereaza)
- seteaza tipul MIME conform configurare server

**text/css**      **css**

**text/html**      **html htm**

**text/plain**      **asc txt**

**text/xml**      **xml**

**video/mpeg**      **mpeg mpg mpe**

- seteaza alte antete (Content-Length, Last-Modified etc.)
- antet transfer pe bucati (de ex. chunked)

**HTTP/1.1 200 OK**

**Content-Type text/plain**

**Content-Transfer-Encoding: chunked**



- **Conexiunea persistenta**
  - cozi de cereri si de raspunsuri – pastreaza **ordinea**:
    - **raspunsurile** din coada de raspunsuri pastreaza ordinea **cererilor** din coada de cereri



# Functionare server

- server HTTP = set de thread-uri care proceseaza cererile clientilor
- **Fisier configurare fizica (exemplu din Apache pentru Windows)**

**ServerName demo**

**ServerRoot "C:/Program Files/Apache Group/Apache"**

**ServerType Standalone** pastrat continuu in executie

**Port 80**

**KeepAlive On** suporta conexiuni persistente

**MaxKeepAliveRequest 100** nr maxim cereri in asteptare

**KeepAliveTimeout 15** taie conex. daca nu cerere noua in 15 sec

**MaxRequestsPerChild 200** nr max cereri procesate fara repornire child

**Timeout 300** timp maxim de procesare a unei cereri



## Studiu individual

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

### 7.3 WORLD WIDE WEB

- 7.3.1 Aspecte arhitecturale
- 7.3.2 Documente Web statice
- 7.3.3 Documente Web dinamice
- 7.3.4 HTTP – HyperText Transfer Protocol

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

### 7.3 THE WORLD WIDE WEB

- 7.3.1 Architectural Overview
- 7.3.2 Static Web Pages
- 7.3.3 Dynamic Web Pages and Web Applications
- 7.3.4 HTTP—The HyperText Transfer Protocol



# Nivelul Aplicație

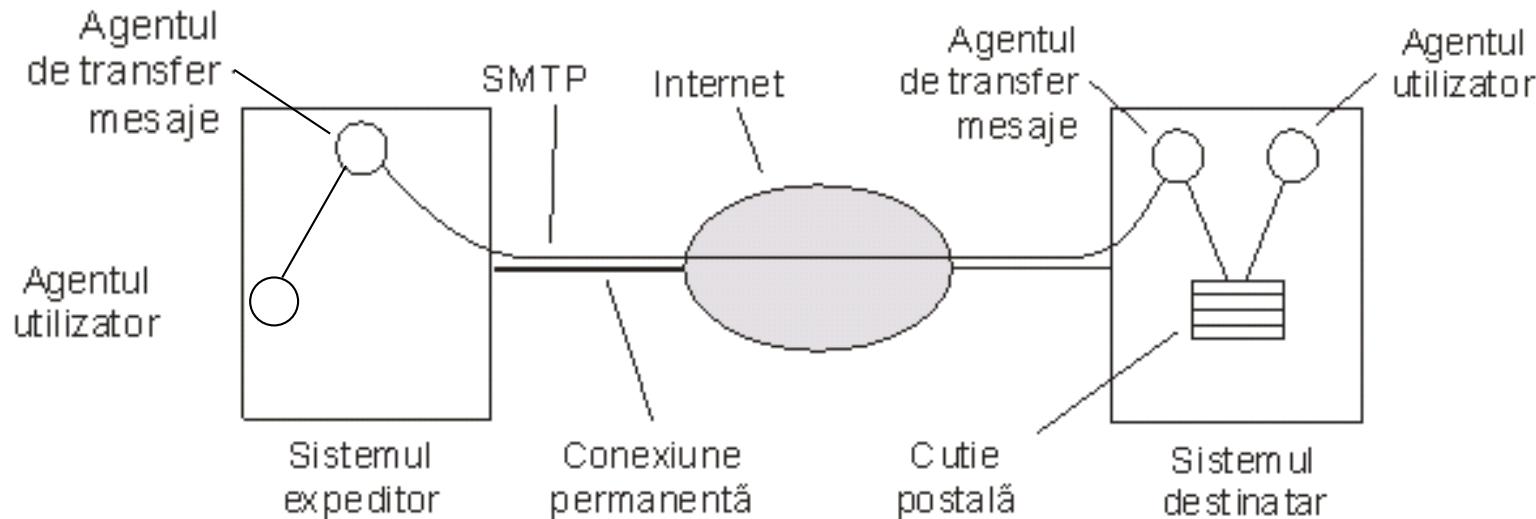
**Poșta electronică (SMTP, POP3, IMAP)**



# Cuprins

- Arhitectura sistemului de e-mail
- Adrese de e-mail
- Formatul mesajelor
- SMTP
- Livrarea finală a mesajelor

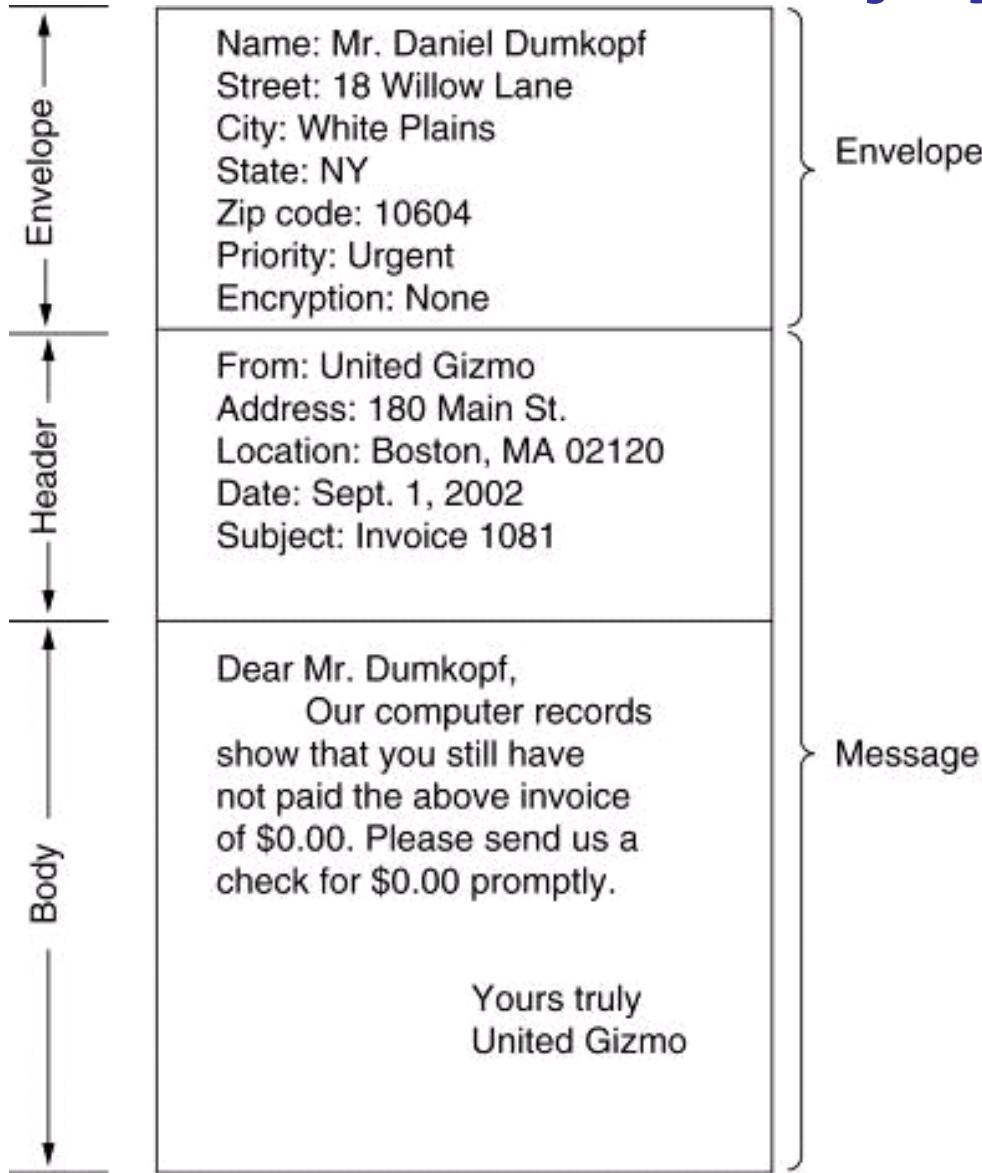
# Arhitectura sistemului de e-mail



- **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 și 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

text sau multimedia



# Posta electronică: Adrese e-mail

- Adresa e-mail  
 $\text{nume\_utilizator@nume\_server\_mail}$
- 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:
    - primește mesajul de la client
    - interpreteaza nume\_utilizator conform cu adresele locale
    - plaseaza mesajul in **cutia postala** corespunzatoare



# Afisarea continutului 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 în cutia postala (mesajul nu este nou)

**A – Answered** – mesaj la care s-a raspuns

**F – Forwarded** – mesaj retransmis altui utilizator



# Formatul mesajelor (RFC 5322)

Antet	Conținut
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 transmițătorului 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 transmițător

Campuri din antet care se referă la [transportul mesajului](#).

Unele campuri sunt folosite de [agentul de transfer](#)

Mesaje auto-continue



# Câmpuri folosite de agentul utilizator sau de utilizator

Antet	Conținut
Date:	Data și 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 referință pentru acest mesaj (identificator)
In-Reply-To:	Identifierul mesajului al cărui răspuns este mesajul curent
References:	Identifică un sir de mesaje dintr-o conversație
Keywords:	Cuvinte cheie alese de utilizator
Subject:	Scurt cuprins al mesajului, afișabil pe o singură linie



# Antete adăugate de MIME – Multi-Purpose Internet Mail Extensions

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



## Antete adaugate de MIME (2)

**Content-Transfer-Encoding** – indica ce codificare a fost folosita pentru a converti datele dintr-un tip MIME in mesaje ASCII, singurele acceptate de e-mail

cerință impusa de sistemele intermediare prin care trec mesajele de mail (ex. gateway)

Pentru SMTP normal

**base64** - 3 valori de 8 biti se transforma in 4 caractere ASCII

**7bit** – cel mult 1000 octeti pe linie cu cod 1..127 din care CR / LF (coduri 13 si 10) la sfarsit de linie.

**quoted-printable** – un octet (orice valoare) este codificat cu 3 caractere: un "=" urmat de doua cifre hexa (0–9 sau A–F) reprezentand valoarea numerica.



# Tipuri si subtipuri MIME definite in RFC 2045

Tip	Subtip	Descriere
Text	Plain	Text neformatat
	Enriched	Text incluzând comenzi simple de formatare
Image	Gif	Imagini fixe în format GIF
	Jpeg	Imagini fixe în format JPEG
Audio	Basic	Sunet
Video	Mpeg	Film în format MPEG
Application	Octet-stream	Secvență neinterpretată de octeți
	Postscript	Un document afișabil în PostScript
Message	Rfc822	Un mesaj MIME RFC 822
	Partial	Mesajul a fost fragmentat pentru transmisie
	External-body	Mesajul în sine trebuie adus din rețea
Multipart	Mixed	Părți independente în ordine specificată
	Alternative	Același mesaj în formate diferite
	Parallel	Părțile trebuie vizualizate simultan
	Digest	Fiecare parte este un mesaj RFC 822 complet

# 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 <b>Carolyn</b>  
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--



# SMTP

- *Simple Mail Transfer Protocol* (SMTP) este protocolul standard de aplicație pentru livrarea mesajelor de posta electronică de la sursă la destinație
- Folosește TCP și un schimb de **mesaje-text** între client și server
  - comenzi (MAIL, RCPT, DATA, QUIT,...)
    - nume comanda urmat de parametri
    - răspunsuri
      - număr din 3 cifre (caractere) urmat de text
- Oferă o livrare sigură a mesajelor
- Alte funcții:
  - Verificarea numelui unui utilizator  
S: VRFY Smith  
R: 250 Fred Smith <Smith@USC-ISIF.ARPA>



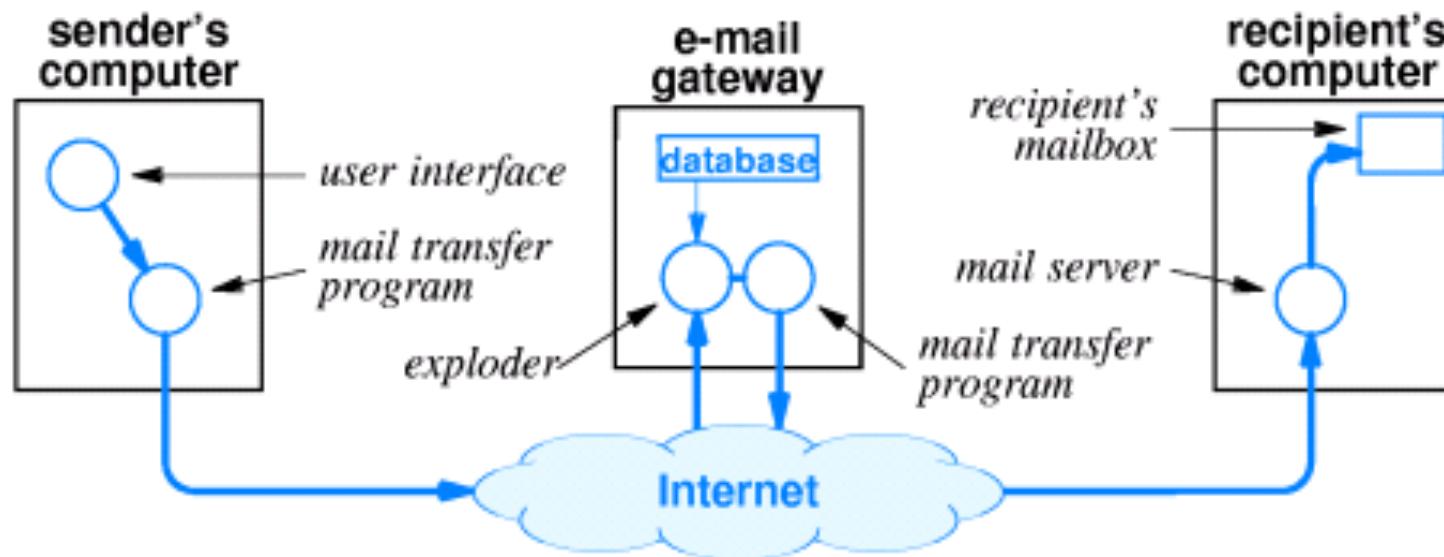
# Transfer e-mail

S: 220 xyz.com SMTP service ready  
**C: HELO abcd.com**  
 S: 250 xyz.com says hello to abcd.com  
**C: MAIL FROM: <elinor@abcd.com>**  
 S: 250 sender ok  
**C: RCPT TO: <carolyn@xyz.com>**  
 S: 250 recipient ok  
**C: DATA**  
 S: 354 Send mail; end with "." on a line by itself  
 C: From: elinor@abcd.com  
 C: To: carolyn@xyz.com  
 C: MIME-Version: 1.0  
 C: Message-Id: <0704760941.AA00747@abcd.com>  
 C: --qwertyuiopasdfghjklzxcvbnm  
 C: .  
 S: 250 message accepted  
**C: QUIT**  
 S: 221 xyz.com closing connection

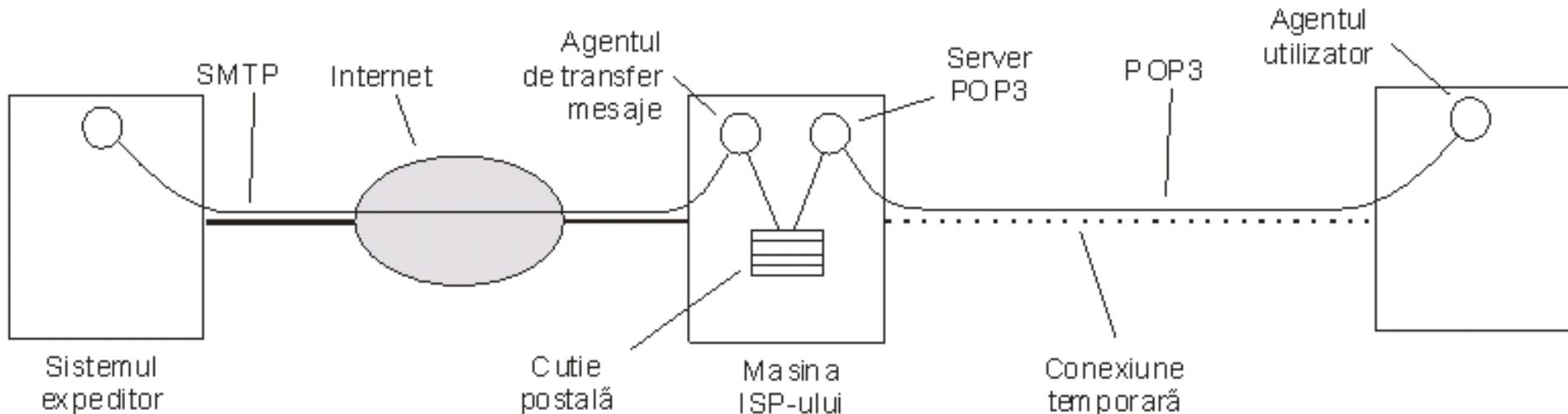
Secventa incepe dupa stabilire conex TCP

# Porti de e-mail (mail gateways)

- Prelucrarea listelor de e-mail poate ocupa resurse importante, în special în marile organizații
- Operatiile pot fi preluate de un server dedicat: *e-mail gateway*
  - Asigura o adresa unică pentru toate mesajele trimise unui grup
  - Un *exploder* cere transmiterea unei copii a mesajului pentru fiecare adresă de destinație din lista



# Livrarea finală



Citire e-mail cand destinatarul foloseste o conexiune **temporara** pentru a accesa cutia postala.

## Protocol

POP – Post Office Protocol

IMAP – Internet Message Access Protocol



# Folosirea POP3 pentru a citi 3 mesaje

AUTHORIZATION

S: +OK POP3 server ready  
C: USER carolyn

S: +OK

C: PASS vegetables

S: +OK login successful

C: LIST

S: 1 2505

S: 2 14302

S: 3 8122

S: .

C: RETR 1

S: (sends message 1)

C: DELE 1

C: RETR 2

S: (sends message 2)

C: DELE 2

C: RETR 3

S: (sends message 3)

C: DELE 3

C: QUIT

S: +OK POP3 server disconnecting

TRANSACTIONS

UPDATE



# IMAP

- **Internet Message Access Protocol (IMAP)**
  - Stil de lucru similar cu POP3
  - Definit în RFC diferit și folosește port diferit de POP3
  - Permite accesul la mail de pe diferite stații de lucru
    - IMAP păstrează mail-uri la server
    - POP3 descarcă mail-uri pe calculatorul client și (de regulă) le sterge de la server



# Studiu individual

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

## 7.2 POȘTA ELECTRONICĂ

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

## 7.2 ELECTRONIC MAIL



# Nivelul Aplicație

## Transferul de fișiere (FTP)



# Cuprins

- Interfata linie-de-comanda si interfata grafica
- Modelul FTP
- O sesiune FTP
- Conexiunea de date
- Transfer FTP intre doua servere
- Securitatea



# File Transfer Protocol

- Standard pentru transfer de fisiere (RFC959)
- Protocol independent de SO si de hardware
- Protocolul permite:
  - Transferul unor fisiere oarecare
    - intre utilizator si server
    - intre doua servere
  - Listarea continutului unui director
  - Modificarea directorului curent
  - Gestionarea restrictiilor de acces
- Poate rula *interactiv* sau *automat*
- Precede TCP/IP, a fost adaptat ulterior la TCP/IP



# Interfata linie-de-comanda cu utilizatorul

!	cr	macdef	proxy	send
\$	delete	mdelete	sendport	status
account	debug	mdir	put	struct
append	dir	mget	pwd	sunique
ascii	disconnect	mkdir	quit	tenex
bell	form	mls	quote	trace
binary	get	mode	recv	type
bye	glob	mput	remotehelp	user
case	hash	nmap	rename	verbose
cd	help	ntrans	reset	?
cdup	lcd	open	rmdir	
close	ls	prompt	runique	

- Nu e standardizata, dar
- Interfata utilizator din BSD UNIX este standard *de facto*
- Nu sunt comenzi de protocol FTP!



# Interfata grafica WS\_FTP

**WS\_FTP Pro hitmillcom.80.centralinfo.net**

Local System				Remote System			
E:\ZONE\html\texture				/users/hitmillcom/html/texture			
	Name	Date	Size		Name	Date	Size
..				..			
0101	bg010401.html	1010401 12:50		0101	bg010401.html	1010401 21:03	1097
0101	bg010401b.html	1010401 13:16		0101	bg010401b.htm~	1010401 21:03	1096
0101	bg_blue1.html	1010403 21:25		0101	bg_blue1.htm~	1010404 04:36	1094
0101	bg_ltblue2.html	1010403 20:31		0101	bg_ltblue2.ht~	1010404 04:36	1095
0101	bgsolar.html	1010401 13:49		0101	bgsolar.html	1010401 21:03	1094
0101	freetexture.html	1010417 12:03		0101	freetexture.h~	1010418 02:29	6137
0101	hitmill2.html	1010410 21:47		0101	hitmill2.html	1010411 05:07	1102
0101	parchment3.html	1010410 17:33		0101	parchment3.ht~	1010411 05:07	1099
0101	pinks01.html	1010331 20:43		0101	pinks01.html	1010401 04:44	1089
0101	skypatch01.html	1010331 20:43		0101	skypatch01.ht~	1010401 04:45	1102
0101	waterlily01.html	1010331 20:44		0101	waterlily01.h~	1010401 04:45	1098
0101	WS_FTP.LOG	1010417 19:28					

ChgDir, MkDir, View, Exec, Rename, Delete, Refresh, DirInfo

Local System: E:\ZONE\html\texture

Remote System: /users/hitmillcom/html/texture

File Transfer Options: ASCII (radio button selected), Binary, Auto

Protocol Status:

- 150 Opening ASCII data connection for directory listing
- Received 912 bytes in 0.1 secs, (80.00 Kbps), transfer succeeded
- 226 transfer complete

Buttons: Close, Cancel, LogWnd, Help, Options, About, Exit

# Modelul FTP

Are două conexiuni

- **control** - pentru comenzi și răspunsuri

- **User PI** – User Protocol Interpreter

- initiază conexiunea de control (lucrează după protocolul telnet)
    - generează **comenzi FTP**

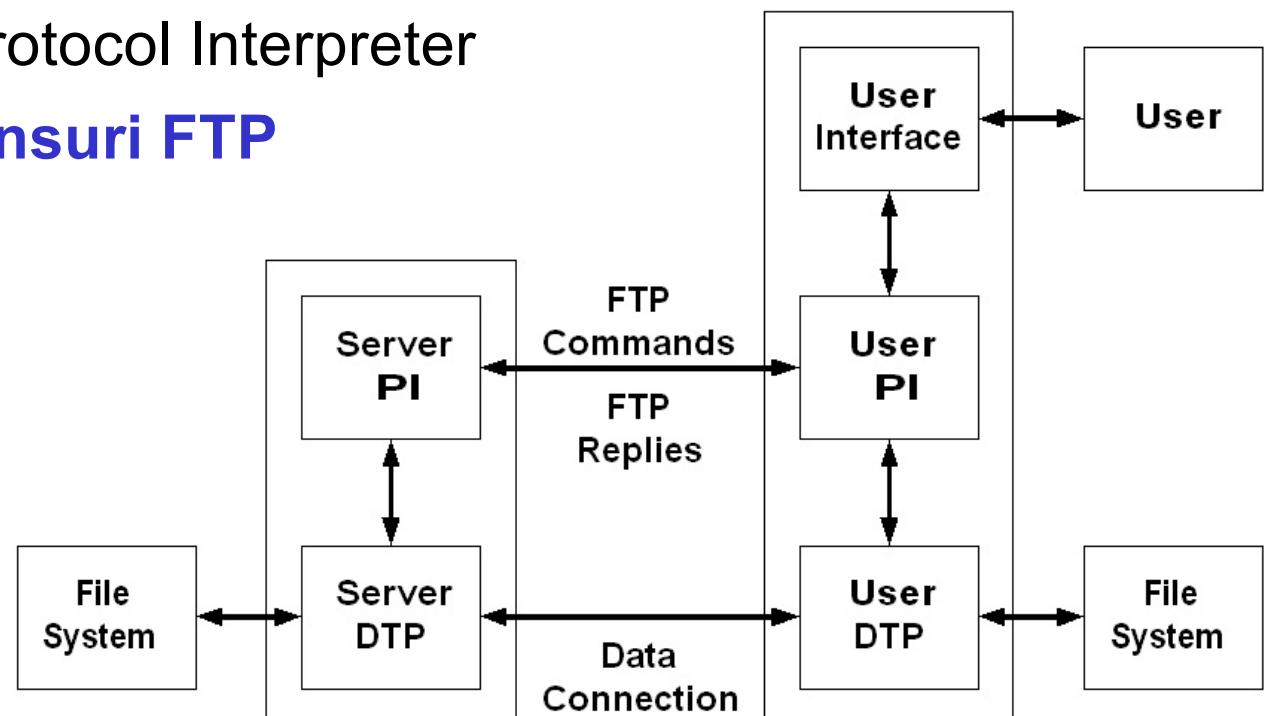
- **Server PI** – Server Protocol Interpreter

- generează **răspunsuri FTP**

- **date** - pentru transmisie / receptie simultană

- User **DTP** – User Data Transfer Protocol

- Server DTP – Server Data Transfer Protocol





# Cateva Comenzi FTP

## Comanda

ABOR

ALLO <bytes>

CWD <dir path>

DELE <filename>

MODE <mode>

MKD <directory>

QUIT

TYPE <data type>

USER <username>

PASS <password>

## Descriere

Abort proces de conexiune de date

Aloca spatiu fisier pe server

Schimba director de lucru pe server

Sterge fisier specificat, pe server

Mod de transfer (S=stream, B=block,  
C=compressed).

Creaza director specificat, pe server

De-logheaza de pe server

Tip date (A-ASCII, E-EBCDIC, B-binar)

username pentru login

parola pentru login

**Format Comenzi:** 3-4 caractere plus parametri



# Cateva Raspunsuri FTP

Raspuns	Descriere
150	Deschide conexiunea
200	OK
220	Serviciu pregatit
221	De-logare retea
226	Inchide conexiunea de date
250	Actiune terminata

**Raspuns:** cod numeric & blanc & text descriptiv



ksh\$ /usr/bin/ftp

ftp> **open ftp.acmemail.example.com**

Connected to ftp.acmemail.example.com (172.16.62.36).

220 Hello, this is the Acme Mail Service.

Name (ftp.acmemail.example.com:root): **MB1234**

331 Password required to access user account MB1234.

Password: **QXJ4Z2AF**

230 Logged in.

ftp> **cd Bills**

250 "/home/MB1234/Bills" is new working directory.

ftp> **ls**

200 PORT command successful.

150 Opening ASCII mode data connection for /bin/ls.

-rw-r--r-- 1 ftpuser ftpusers 14886 Dec 3 15:22 Acmemail.TXT

-rw-r--r-- 1 ftpuser ftpusers 317000 Dec 4 17:40 Yoyodyne.TXT

226 Listing completed.

**O sesiune FTP**  
mesajele introduse de utilizator sunt reprezentate **ingrosat**

Dupa logare,  
utilizatorul:

- schimba directorul distant
- listeaza continutul



## Sesiune FTP (continuare)

ftp> **get Yoyodyne.TXT**

local: Yoyodyne.TXT remote: Yoyodyne.TXT

200 PORT command successful.

150 Opening ASCII mode data connection for Yoyodyne.TXT.

226 Transfer completed.

317000 bytes received in 0.0262 secs (1.2e+04 Kbytes/sec)

ftp> **quit**

221 Goodbye.

- descarca un fisier
- inchide conexiunea



# Conexiunea de date

Serverul la distanță acceptă *conexiune de control* de la client

Se stabilește apoi *conexiunea de date* – metode:

**Activa** – serverul (activ) se conectează la client

- clientul specifică pe conexiunea de control o adresă și un port;
- serverul inițiază conexiunea de date (ex. port **1931**, adresa 192.168.1.2)

**Client:** PORT 192,168,1,2,**7,139**

**Server:** 200 PORT command successful

**Server:** inițiază conexiunea de date la portul specificat de client

**Client :** confirma



# Conexiunea de date

Pasiva – clientul initiaza conexiunea cu serverul (pasiv)

- clientul cere, pe conexiunea de control, serverului sa asculte la o adresa si un port (care nu este portul sau standard);
- serverul comunica adresa si portul (ex. port **3439**, adresa 172.16.62.36);

**Client:** PASV

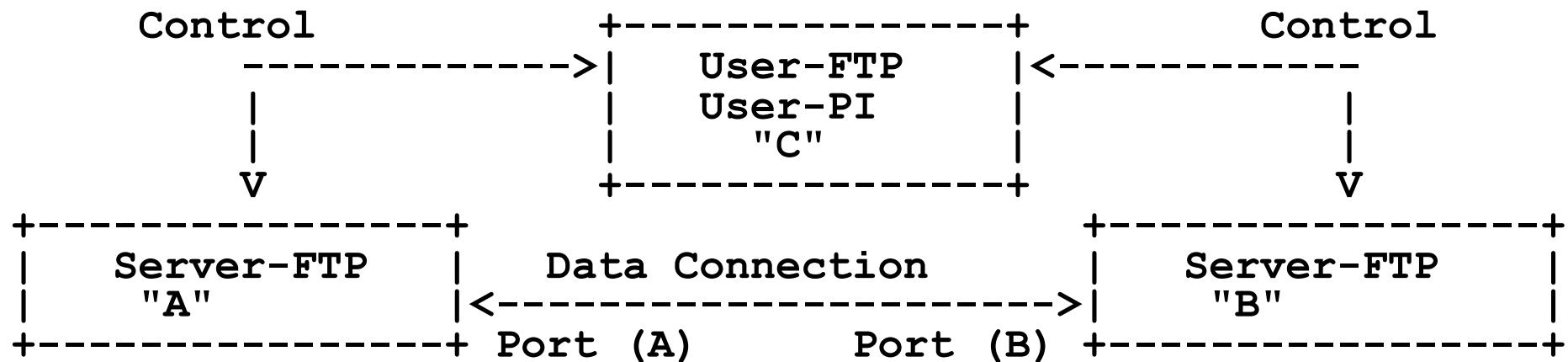
**Server:** Entering Passive Mode (172,16,62,36,**13,111**)

**Client:** initiaza conexiunea de date la portul specificat de server

**Server:** confirma



# Transfer FTP între două servere



User-PI - Server A

C->A : Connect

C->A : PASV

A->C : 227 Entering Passive Mode A1,A2,A3,A4,a1,a2

C->A : STOR file

User-PI - Server B

C->B : Connect

B->C : 200 Okay

C->B : RETR file

B->A : Connect to HOST-A, PORT-a



# Securitate

- FTP transmite *user name* și *password* în clar
- **FTP over SSH**
  - Transmite *password* criptat
- **SFTP** - SSH File Transfer Protocol
  - Securitate pentru **date**
  - Presupune ca serverul a autentificat utilizatorul
- **bbFTP**
  - Securizează *user name* și *password*
  - Utilizat ne-interactiv (*shell script*)
  - Permite **stream-uri paralele** de date
  - Compresie *on-the-fly*
  - Bun pentru volume mari (> 2GB)



# Protocole de Securitate

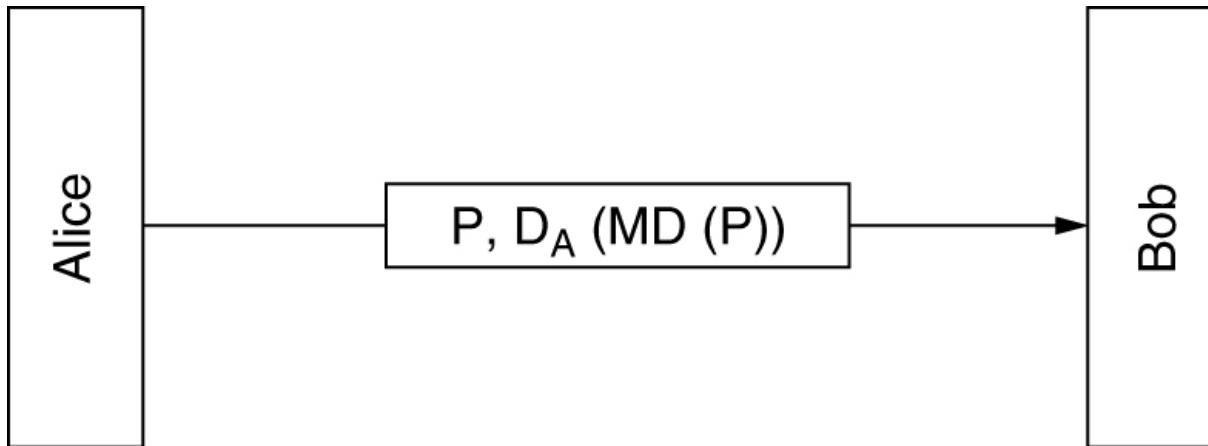
**Rezumate de mesaje, semnaturi digitale și  
protocole de securitate**



# Cuprins

- Rezumatul mesajelor
- Semnaturi digitale
- Certificate de securitate
- PKI
- Securitatea comunicatiei - IPsec
- Protocole de autentificare
- Securitatea e-mail – PGP
- Securitatea Web
- Securitatea DNS

# Rezumatele mesajelor



Este mai usor să  
semnezi digital  
rezumatul decât  
mesajul întreg !

## Folosite în semnaturi digitale datorită proprietăților utile

1. Rezumatul lui P - MD(P) - este ușor de calculat
2. Este imposibil să se afle P din MD(P)
3. Rezumatul nu poate fi trucat: nimeni nu poate găsi P' având un rezumat identic cu P, adică  $MD(P') = MD(P)$
4. O schimbare de 1 bit a intrării schimba mulți biti din ieșire

## Functii hash

- MD5 (Message Digest)
- SHA-1 (Secure Hash Algorithm)

# Functii Hash: MD5 - Message Digest 5

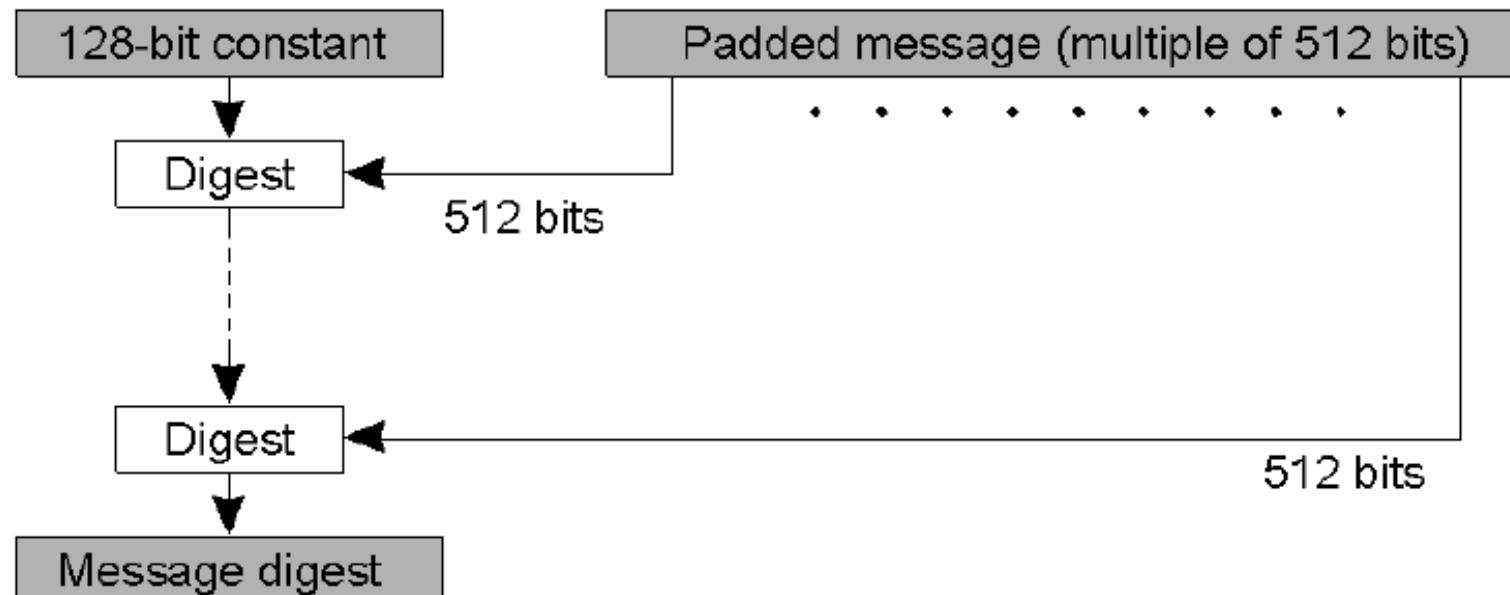
MD5 calculeaza un rezumat de **128 biti** dintr-un mesaj de lungime multipla de 512 biti

Mesajul este completat cu biti pentru a respecta regula

Ultimii **64 biti** precizeaza lungimea originala a mesajului

In fiecare **faza** algoritmul calculeaza un nou rezumat din rezumatul anterior si rezumatul unui bloc de 512 biti.

Primul rezumat este o **constanta** de 128 biti





## Functii Hash: MD5 (2)

O **faza** transforma un **bloc** de mesaj de 512 biti. Are 4 **runde**.

Fiecare **runda** foloseste o functie diferita:

$$F(x,y,z) = (x \text{ AND } y) \text{ OR } ((\text{NOT } x) \text{ AND } z)$$

$$G(x,y,z) = (x \text{ AND } z) \text{ OR } (y \text{ AND } (\text{NOT } z))$$

$$H(x,y,z) = x \text{ XOR } y \text{ XOR } z$$

$$I(x,y,z) = y \text{ XOR } (x \text{ OR } (\text{NOT } z))$$

O runda **are 16 iteratii**.

$b_0, \dots, b_{15}$  – **sub-blocuri** 32-bit (total 512 biti)      p, q, r, s – variabile *digest*

$C_1, \dots, C_{16}$  – constante (in total 64)      <<< denota rotatie stanga

Iterations 1-8	Iterations 9-16
$p \leftarrow (p + F(q,r,s) + b_0 + C_1) \ll 7$	$p \leftarrow (p + F(q,r,s) + b_8 + C_9) \ll 7$
$s \leftarrow (s + F(p,q,r) + b_1 + C_2) \ll 12$	$s \leftarrow (s + F(p,q,r) + b_9 + C_{10}) \ll 12$
$r \leftarrow (r + F(s,p,q) + b_2 + C_3) \ll 17$	$r \leftarrow (r + F(s,p,q) + b_{10} + C_{11}) \ll 17$
$q \leftarrow (q + F(r,s,p) + b_3 + C_4) \ll 22$	$q \leftarrow (q + F(r,s,p) + b_{11} + C_{12}) \ll 22$
$p \leftarrow (p + F(q,r,s) + b_4 + C_5) \ll 7$	$p \leftarrow (p + F(q,r,s) + b_{12} + C_{13}) \ll 7$
$s \leftarrow (s + F(p,q,r) + b_5 + C_6) \ll 12$	$s \leftarrow (s + F(p,q,r) + b_{13} + C_{14}) \ll 12$
$r \leftarrow (r + F(s,p,q) + b_6 + C_7) \ll 17$	$r \leftarrow (r + F(s,p,q) + b_{14} + C_{15}) \ll 17$
$q \leftarrow (q + F(r,s,p) + b_7 + C_8) \ll 22$	$q \leftarrow (q + F(r,s,p) + b_{15} + C_{16}) \ll 22$



# Comentarii

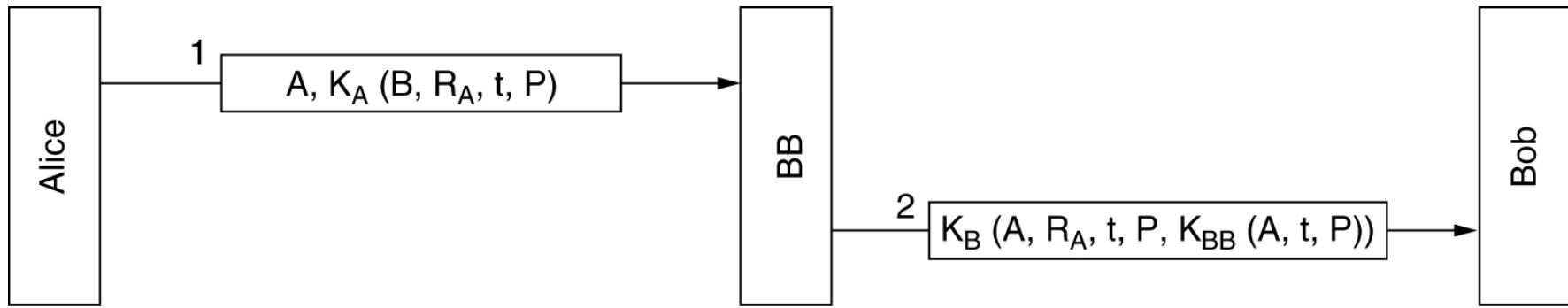
- Rezistenta la coliziuni
  - o functie H este rezistenta la coliziuni daca este foarte greu sa se gaseasca a si b, a≠b astfel ca  $H(a) = H(b)$
- In 2004 s-a aratat ca MD5 nu este rezistent la coliziuni
- S-au dezvoltat si recomandat alte functii de hash
  - SHA1, SHA2
- Obs.
  - criptare # rezumare!



# Semnaturi Digitale

- Bazate pe
  - Chei simetrice
  - Chei publice
- Rezumate de mesaje

# Semnaturi cu chei simetrice



Semnaturi digitale cu Big Brother.

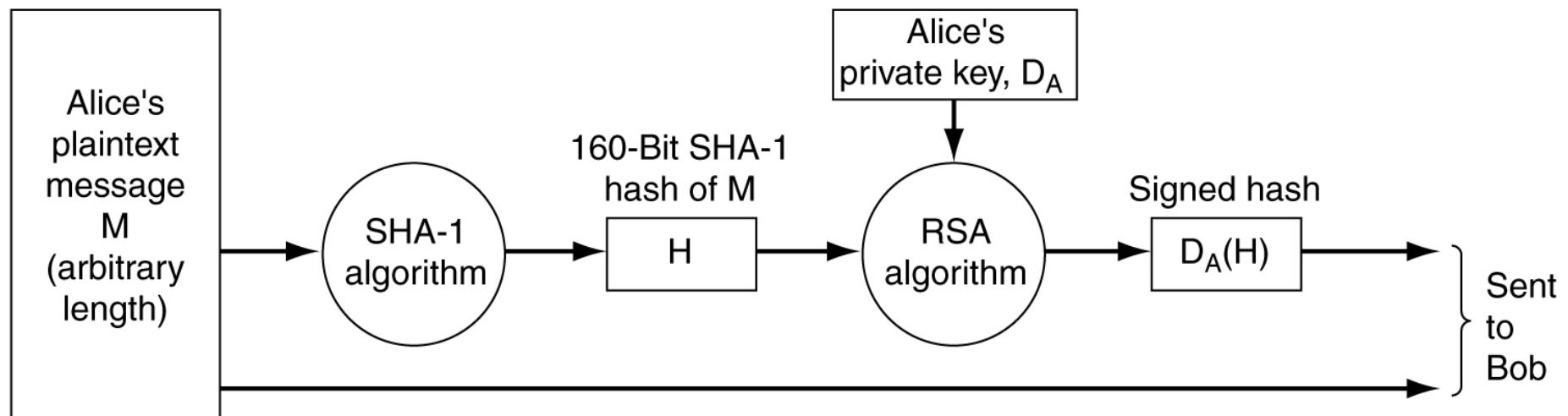
- $R_A$  – numar aleator (control replici)
- $t$  – timestamp (mesaj recent)
- $K_A$  – cheie secreta a lui A (cunoscuta de BB)
- $K_B$  – cheie secreta a lui B (cunoscuta de BB)
- $K_{BB}$  – cheie secreta Big Brother (doar el o cunoaste)

## Comentarii

$t$  și  $R_A$  folosite ptr. detectie atacuri prin replicare mesaje mai vechi  
 $K_{BB}(A, t, P)$  folosit pentru non-repudiere

# Semnaturi cu chei publice

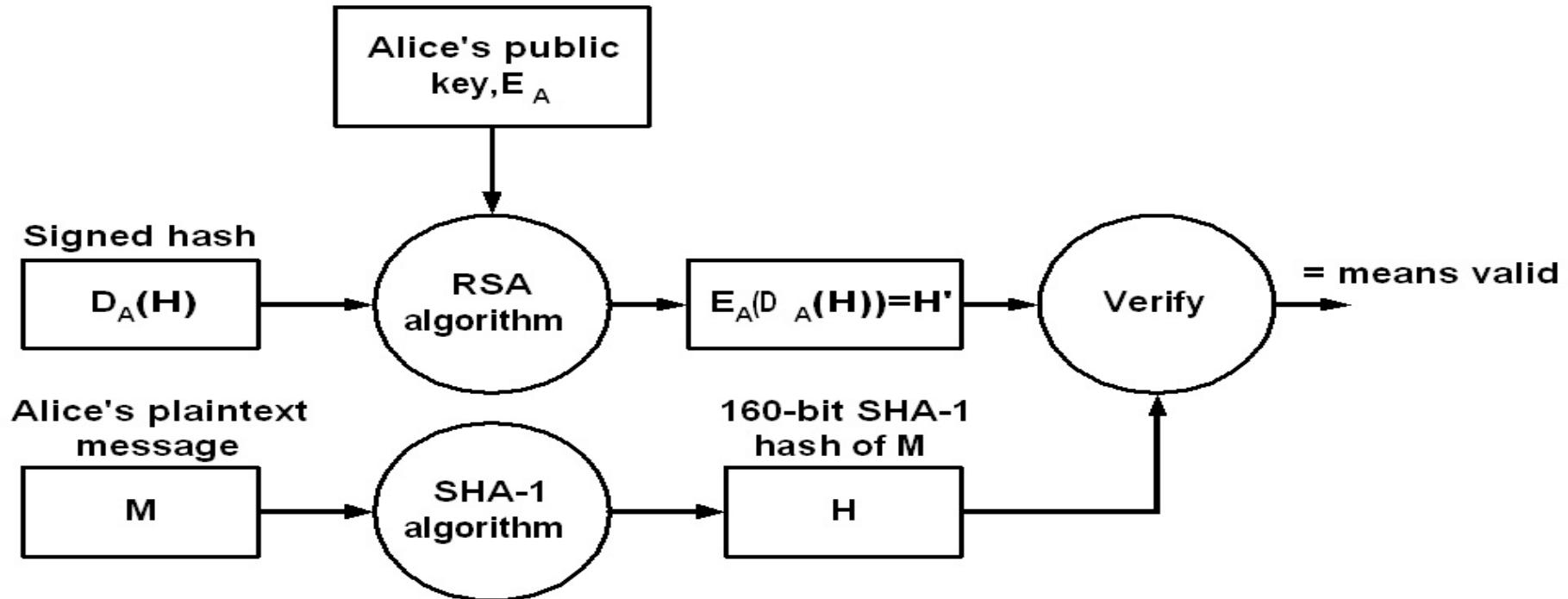
Utilizarea SHA-1 si RSA pentru semnarea mesajelor nesecrete.



## Caracteristici

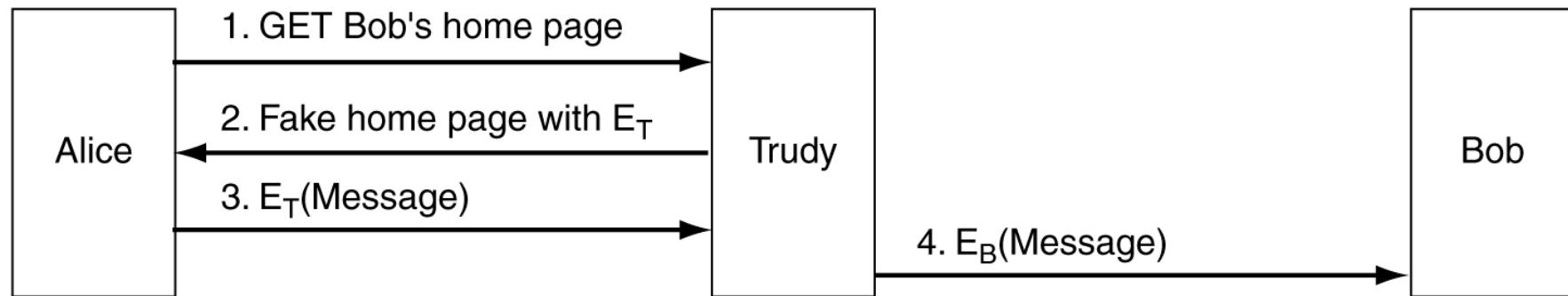
Rezumatul SHA-1 este semnat cu cheia secreta a transmitatorului  $D_A$   
Mesajul  $M$  este transmis in clar

# Verificare semnatura digitală



Orice modificare a textului clar  $M$  este detectată prin  $H \neq H'$   
Un intrus nu poate modifica și  $M$  și rezumatul criptat  $D_A(H)$

# Probleme cu difuzarea cheilor publice



**Problema:** difuzarea cheii publice prin pagina de referinta a proprietarului

Trudy raspunde in locul lui Bob cu cheia sa publica

Trudy poate modifica mesajele trimise de Alice lui Bob

Man-in-the-middle attack



# Certificate de securitate

- Certificate
  - Asociază identitatea cu cheia publică
- X.509
  - Standard de certificate



# Certificate

Rol: leaga cheia publica de un proprietar ([principal](#)) sau de un atribut

I hereby certify that the public key

19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A

belongs to

Robert John Smith

12345 University Avenue

Berkeley, CA 94702

Birthday: July 4, 1958

Email: bob@superduper.net.com

SHA-1 hash of the above certificate signed with the CA's private key

Un certificat nu este secret

este [semnat de o autoritate de certificare - CA \(Certificate Authority\)](#)

CA cripteaaza cu cheia sa privata rezumatul certificatului

Verificarea certificatului de catre Alice

A aplica cheia publica a CA asupra semnaturii

A calculeaza rezumatul SHA-1 al certificatului (fara semnatura)

A compara cele doua rezultate



# Campurile de baza dintr-un certificat X.509

Câmp	Semnificație
Versiune	Ce versiune de X.509 este utilizată
Număr Serial	Acest număr împreună cu numele CA-ului identifică în mod unic certificatul
Algoritm de semnare	Algoritm folosit la semnarea certificatului (ex. MD5 cu RSA)
Emitent	Numele X.500 al CA-ului
Perioada de validitate	Momentele de început și sfârșit ale perioadei de validitate
<b>Numele subiectului</b>	<b>Entitatea care este certificată</b>
<b>Cheia publică</b>	<b>Cheia publică a subiectului și ID-ul algoritmului folosit (ex. RSA)</b>
ID emitent	Un ID optional identificând în mod unic emitentul certificatului (nume X.500 sau DNS)
ID subiect	Un ID optional identificând în mod unic subiectul certificatului
Extensii	ptr identificarea cheii publice a emitentului, a certificatului care conține o anumita cheie publică, scopul utilizării cheii (criptare, semnare,...) și altele
<b>Semnătura</b>	<b>Semnătura certificatului (semnat cu cheia privată a CA-ului)</b>



# PKI - Public Key Infrastructure

- **PKI- Set de componente (hard & soft) care asigura utilizarea corecta a tehnologiei de chei publice**
  - programele,
  - echipamentele,
  - tehnologiile de criptare si
  - serviciile de gestiune a infrastructurii criptografice si a cheilor publice ale utilizatorilor.



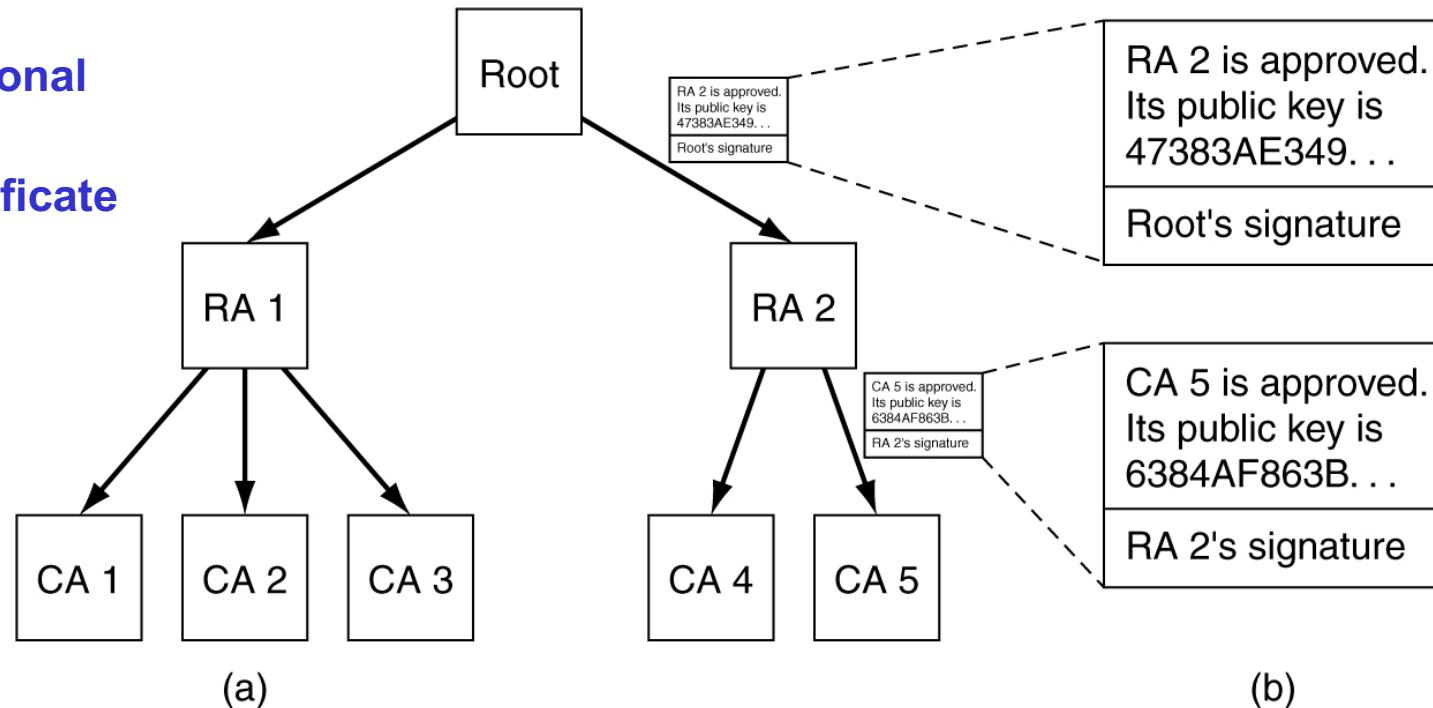
# CA - Certificate Authority

- autoritate de incredere care elibereaza certificate
  - atestă că cheia publică inclusă aparține persoanei cu numele atașat
- CA poate fi:
  - organizatie sau companie - pentru angajati
  - universitate - pentru studenti
  - CA publice (VeriSign) - pentru clienti

# PKI – verificarea cheilor

**RA – Regional Authority**

**CA – Certificate Authority**



(a) PKI ierarhic.

(b) Un lant de incredere (certification path).

A cunoaste si are incredere in Root

- gaseste certificatul lui B semnat de **CA 5**
- certificatul lui CA 5 semnat de **RA 2**
- certificatul lui RA 2 semnat de **Root**

Simplificare

A primește de la B tot lantul de certificate



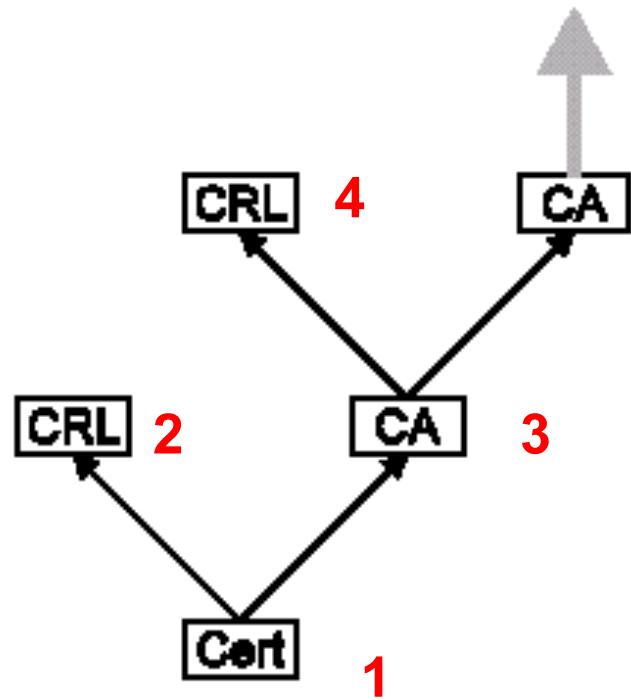
# Revocarea Certificatelor

- Un certificat trebuie **revocat** cand:
  - cheia primara este **compromisa**;
  - cheia primara este **pierduta**;
  - o persoana pleaca din companie
  - altele.
- Revocarea trebuie anuntata tuturor utilizatorilor – dificil !
- Alternativa - se folosesc liste de revocare
  - **CRL – Certificate Revocation List**;

## Metoda

- se verifica liste de revocare inainte de utilizarea certificatelor
- CRL sunt publicate de CA care a emis certificatele
- **Listele pot fi consultate sau duplicate (cache)**
  - difuzarea listelor de revocare – prin HTTP, LDAP sau alte protocoale

# Verificarea revocarii Certificatelor



## Verificare certificate

- 1 - verifica certificat  
2 - verifica CRL  
**repeat**  
    3 - verifica certificatul pentru CA  
    4 - verifica CRL al CA  
**until** radacina



# Securitatea Comunicatiei

- IPsec
- Ziduri de protectie (Firewalls)
- Virtual Private Networks



## IP Security Protocol - IPSec

- Implementat la nivel IP
- Construieste o legatura securizata **unidirectionala** intre transmitator si receptor
  - numita **Security Association - SA**
  - asigura
    - **autentificarea** mesajelor sau
    - **autentificarea si criptarea**
- Securizarea ambelor sensuri → 2 x SA



## Parametri de securitate

- SA nu este legata de un singur algoritm de criptare sau de o singura cheie – **se pot specifica:**
  - **algoritmul si modul** de criptare (ex. DES in mod block-chaining)
  - **cheia** de criptare
  - parametrii de criptare (ex. **Initialization Vector**)
  - protocolul de **autentificare** si **cheia**
  - **durata de viata** a unei asociatii (permite sesiuni lungi cu schimbarea cheii daca este necesar)
  - **adresa** capatului opus al asociatiei
  - **nivelul de senzitivitate** al datelor protejate.



# SA Database

Un sistem pastreaza o **baza de date** cu asociatiile de securitate

- Pentru fiecare SA pastreaza **parametrii de securitate** (slide precedent) și
- **contor** numere de secventa: pentru antete de securitate
- Indicator **overflow** pentru contor numere de secventa: ce-i de facut la depasire limita contor
- fereastra **anti-replay**: determina daca un pachet este o copie
- **Path MTU**: path Maximum Transmission Unit (pentru evitare fragmentare)



## SA Database (2)

Fiecare intrare unic **identificata** de:

- **Security Parameters Index (SPI)**: identificare SA la receptor
- **IP Destination Address**
- **Security Protocol Identifier**

Două protocoale de securitate:

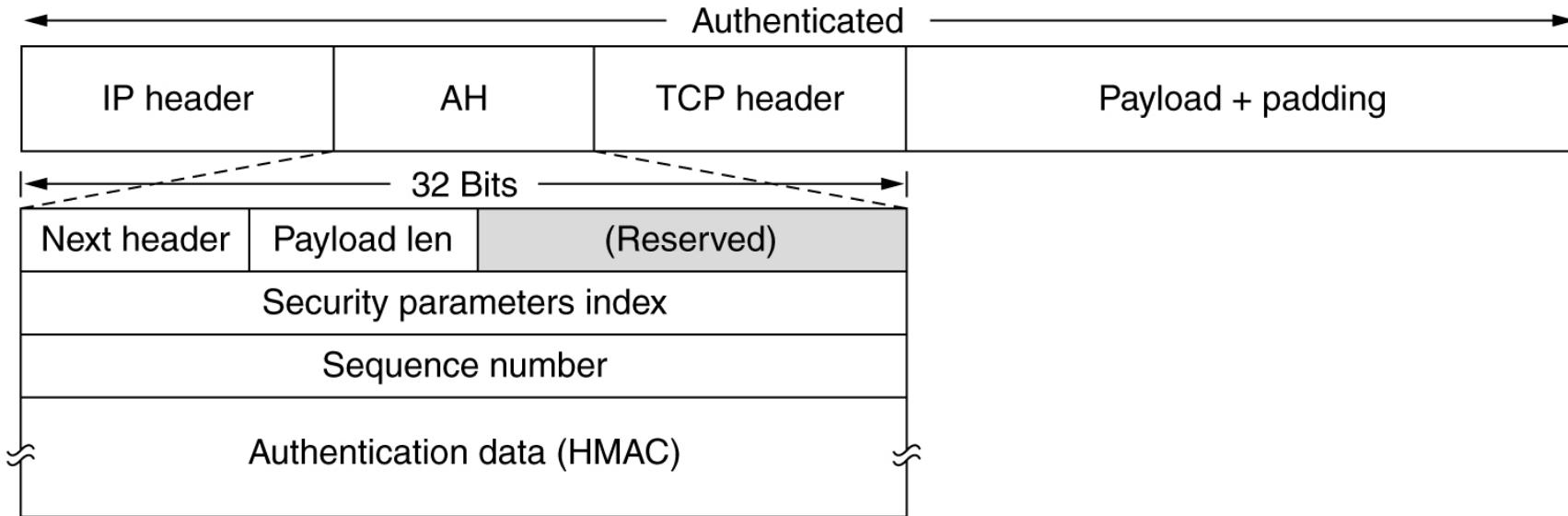
- AH ([Authentication Header](#)) - protocol de autentificare
- ESP ([Encapsulating Security Payload](#)) - protocol combinat criptare/autentificare

Si două **moduri** de lucru

- transport
- tunel



## Protocol AH – în mod transport pentru IPv4

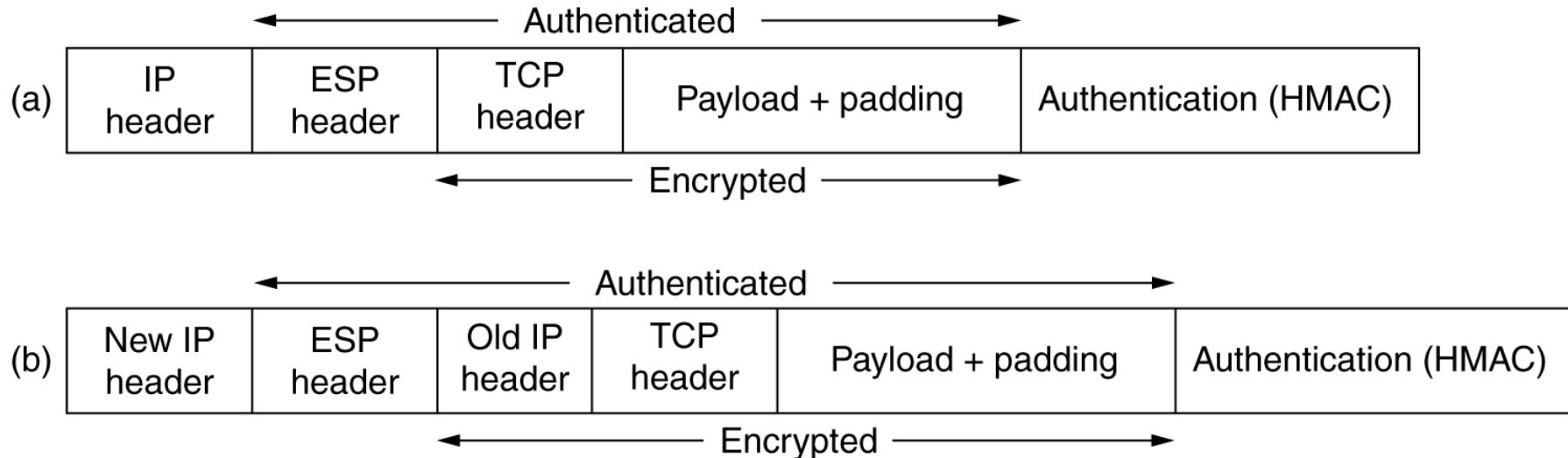


**Authentication Header** – inserat în datagrama IP

- **Next header** – preluat din **IP header** unde este înlocuit cu 51
- **Payload len** – lungime AH (nr cuvinte 32 biti) minus 2
- **Security Parameters Index** – indica înregistrarea din BD a receptorului
- **Sequence number** - evitare atacuri prin replica
- **HMAC** – Hashed Message Authentication Code
  - Utilizează cheia simetrică
  - Calculează rezumat peste întreaga datagramă (campurile variabile neincluse) + cheia simetrică



## ESP in modurile transport si tunel



### ESP – Encapsulating Security Payload

#### (a) ESP in mod transport.

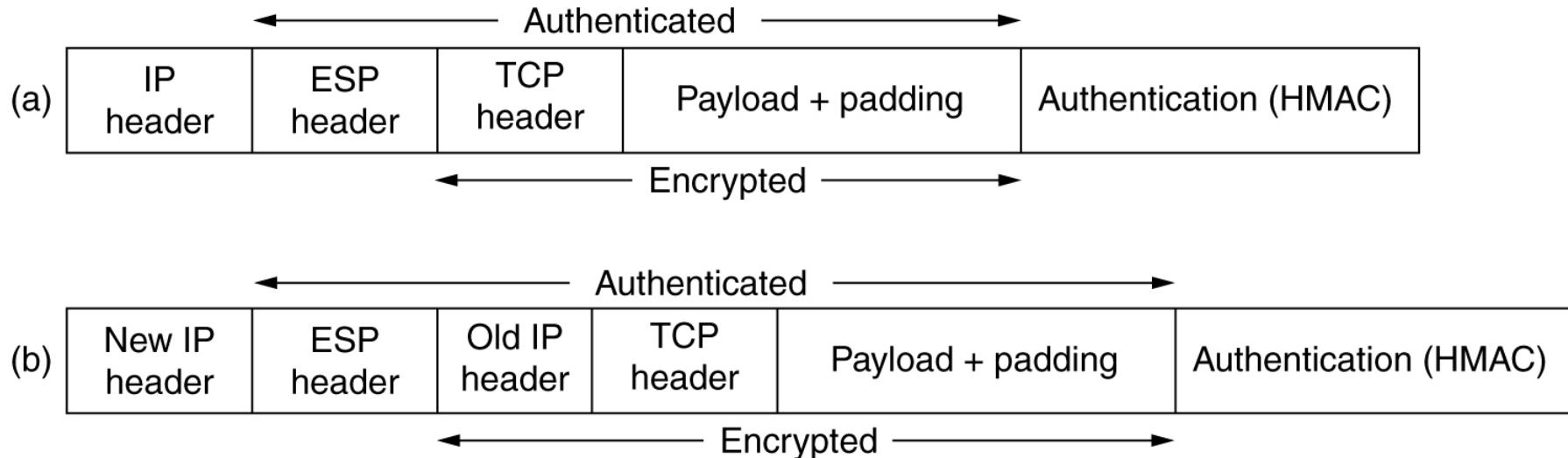
- antetul ESP este plasat intre antetele IP si TCP
- campul “protocol” din antetul IP este modificat si arata ca urmeaza un antet IPsec

#### (b) ESP in mod tunel.

- la pachetul IP se adauga antetul IPsec si un nou antet IP
- tunelul se poate termina inainte de destinatie (de ex. la un firewall)



## ESP in modurile transport si tunel



### ESP – Encapsulating Security Payload

(a) ESP in mod transport. (b) ESP in mod tunel.

- criptarea protejeaza incarcatura;
- autentificarea protejeaza antet ESP + criptograma

### ESP header include

Security Parameters Index

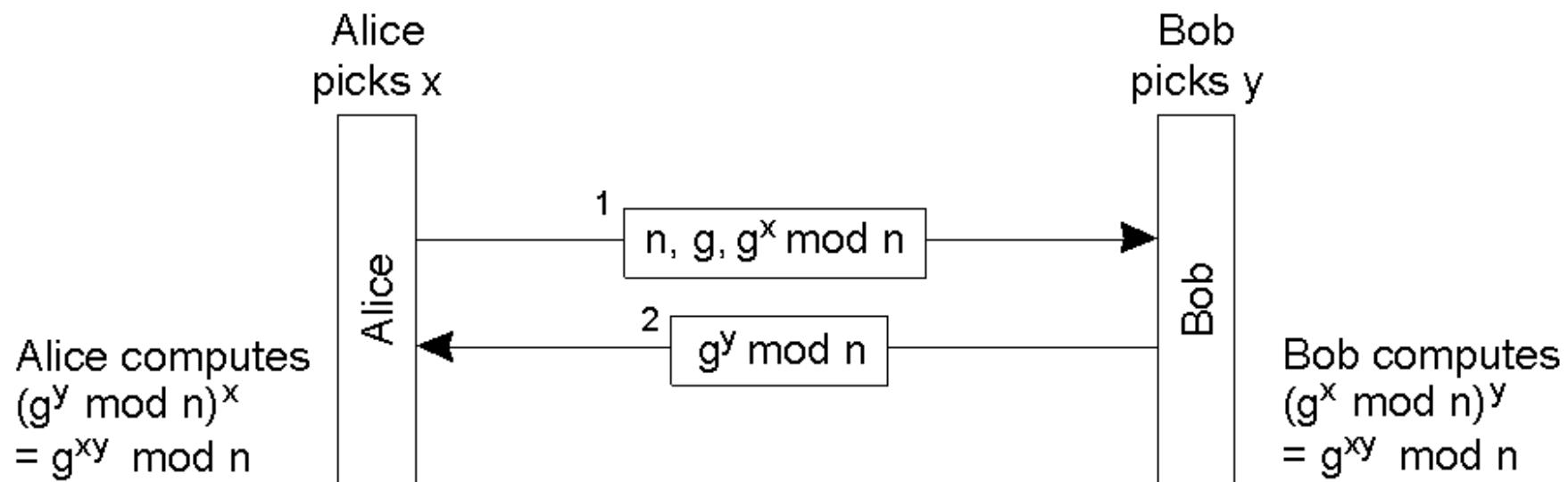
Numar de Sevență

**Vector de initializare** (pentru criptare date)

La sfârșit: HMAC – Hashed Message Authentication Code

# Gestiunea cheilor

- ISAKMP – Internet Security Association Key Management Protocol
- Genereaza o cheie distincta pentru fiecare asociatie
- Implementat cu IKE (ISAKMP Key Exchange)
  - Foloseste Diffie – Hellman
- Pentru Alice:
  - $x$  este cheia privata
  - $g^x \text{ mod } n$  este cheia publica
  - $K_{A,B} = g^{xy} \text{ mod } n$  este cheia secreta partajata cu Bob





# Caracteristici Protocol IPSEC

- IPSec este **orientat pe conexiune** (desi apartine nivelului retea)
- Permite selectia intre **mai multi algoritmi**
  - criptare: DES in mod CBC, 3DES, IDEA, ...
  - autentificare: MD5, SHA (trunchiat la 96 biti)
  - “deschis” la adaugare algoritmi noi
- Permite **stabilirea cheilor** de criptare
- Permite alegerea intre **mai multe servicii**
  - confidentialitate
  - integritate
  - protectie la atacuri prin replica
- Permite **alegerea granularitatii**
  - conexiune TCP
  - toate legaturile intre doua calculatoare (tunel)
  - toate legaturile intre doua rutere, ...



# Protocole de Autentificare

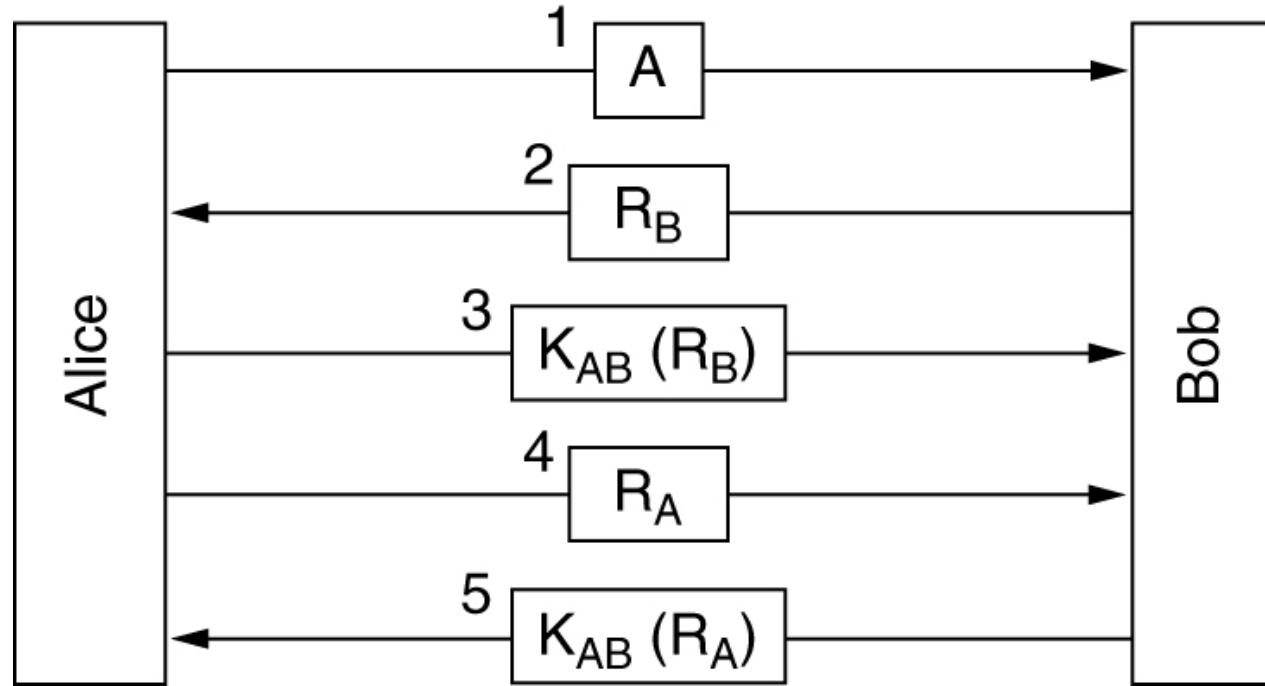
Determina daca o **entitate** (utilizator, proces) este cu adevarat cine / ce pretinde ca este

- diferita de **autorizare**
- se bazeaza pe un schimb de **mesaje** prin Internet (prezentate, de regula, ca schimb intre **Alice si Bob**)
- mesajele pot fi interceptate si folosite de alte entitati (de regula **Trudy**)
- protocolul genereaza si o **cheie de sesiune**

Folosesc **criptografia** cu

- Chei secrete partajate
- Chei publice

# Autentificare cu cheie secreta partajata

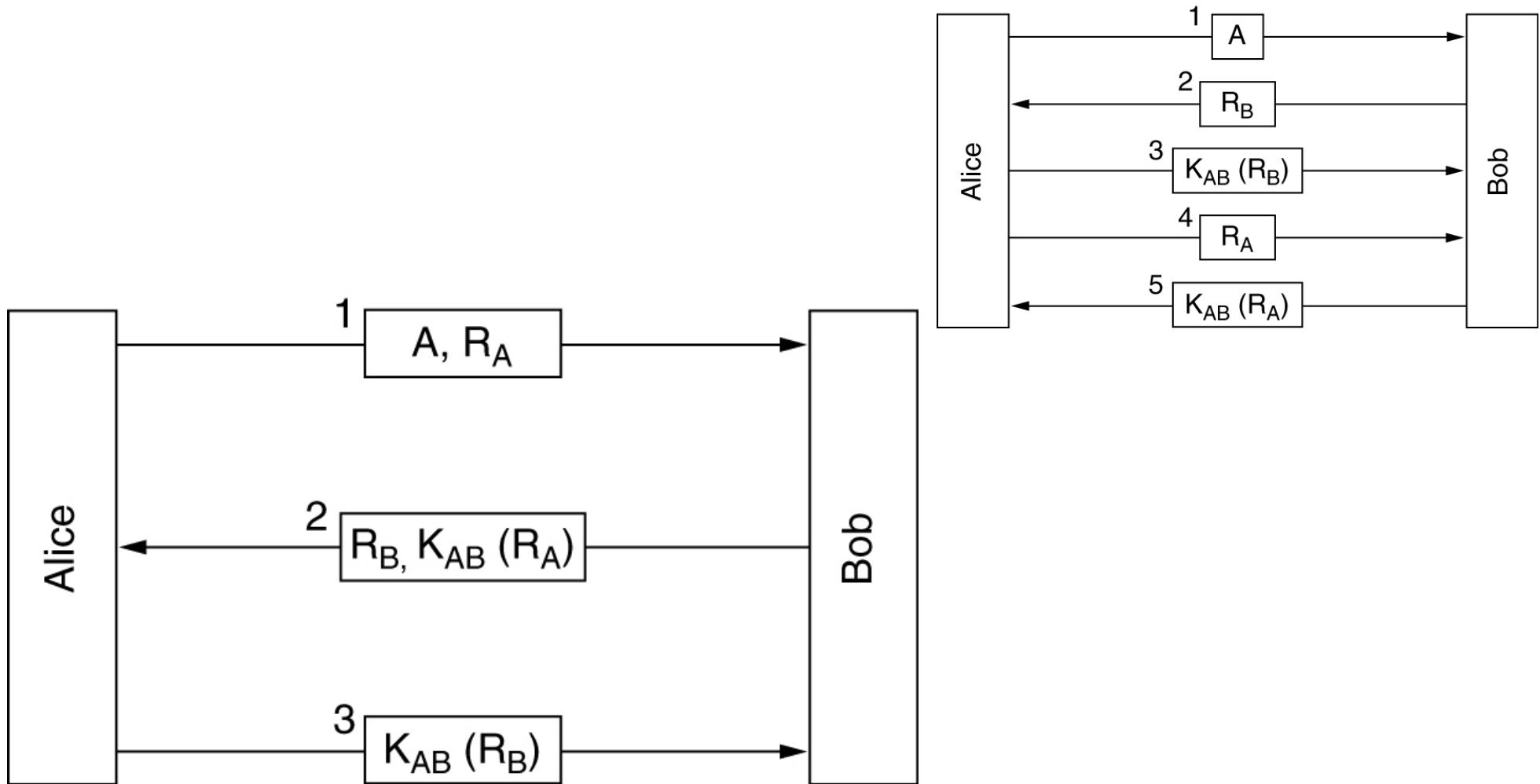


Autentificare reciproca cu un protocol **challenge-response**

Alice si Bob partajeaza cheia  $K_{AB}$

$R_A, R_B$  - numere aleatoare foarte mari, folosite contra **atac prin replica**

## Autentificare cu cheie secreta partajata (2)

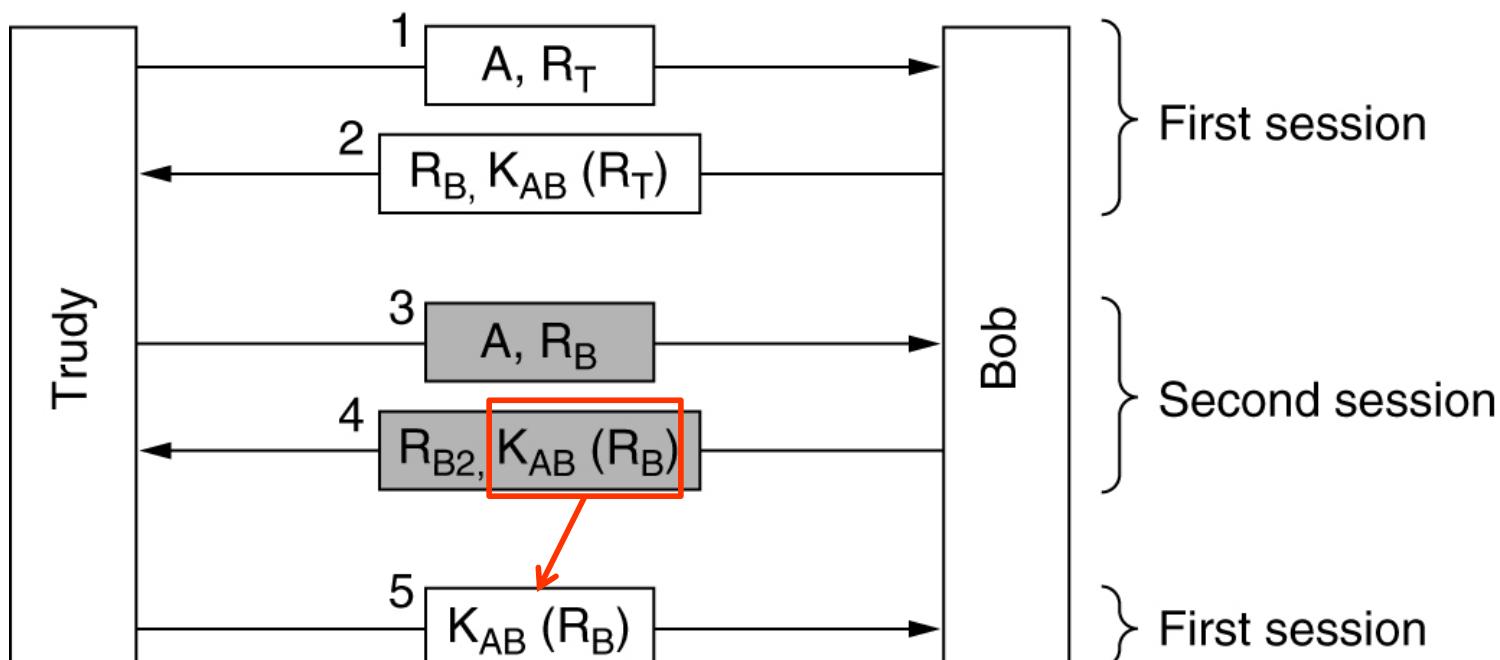
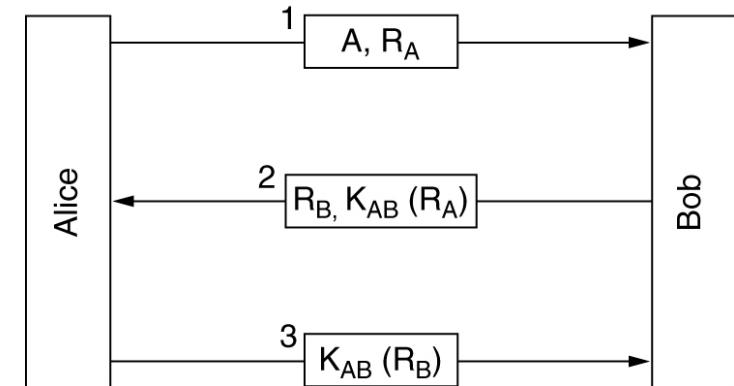


Reducere numar de pasi

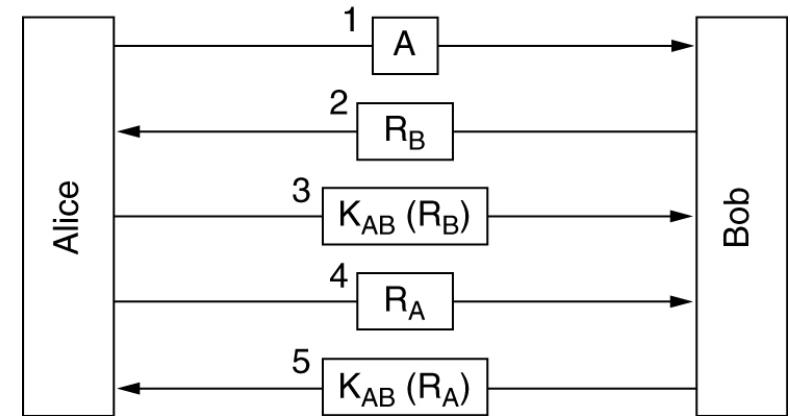
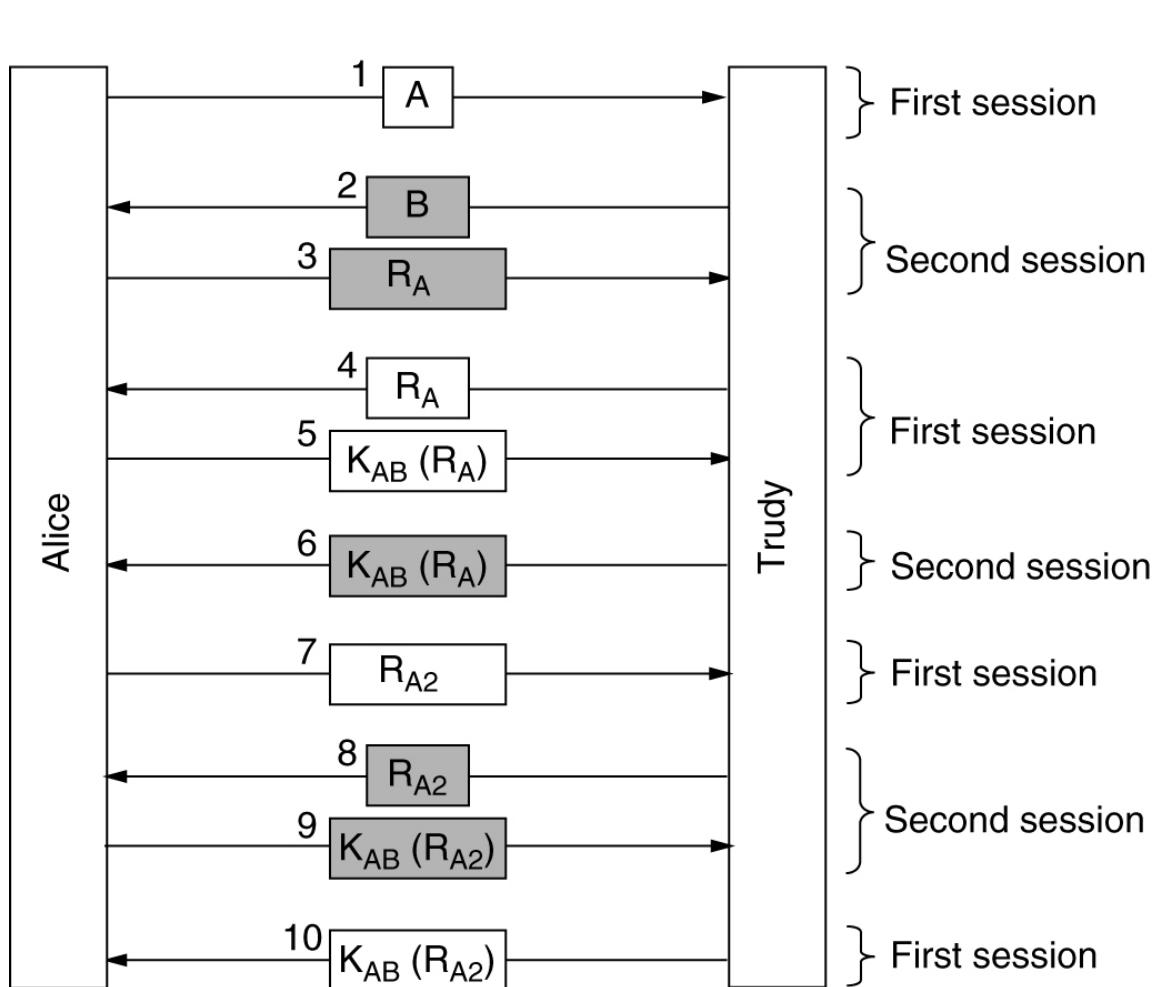
# Atacul prin reflexie

Trudy nu poate cripta  $R_B$  din mesajul 2

Dar **retransmite** un mesaj produs de Bob (4) și reușește să stabilească **o sesiune autenticată** cu Bob

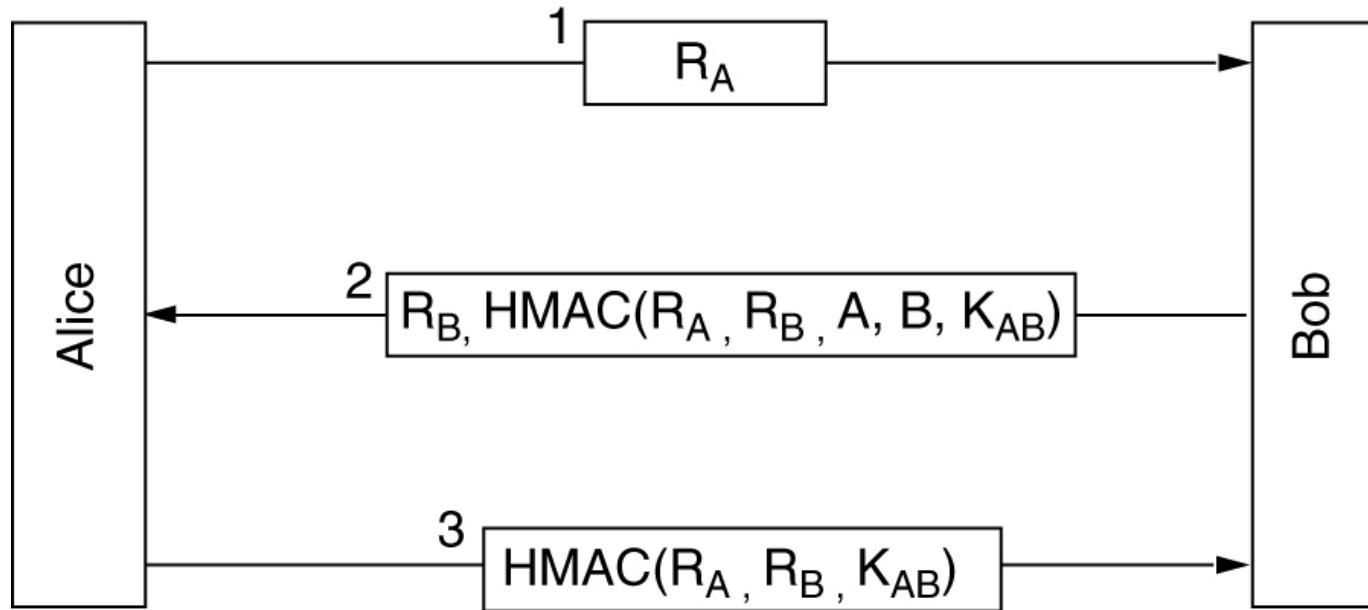


# Atacul prin reflexie pe protocolul initial



Rejucând mesajele 5 și 9,  
 Trudy reușește să  
 stabilească **două sesiuni**  
**autentificate** cu Alice

# Autentificarea cu HMACs

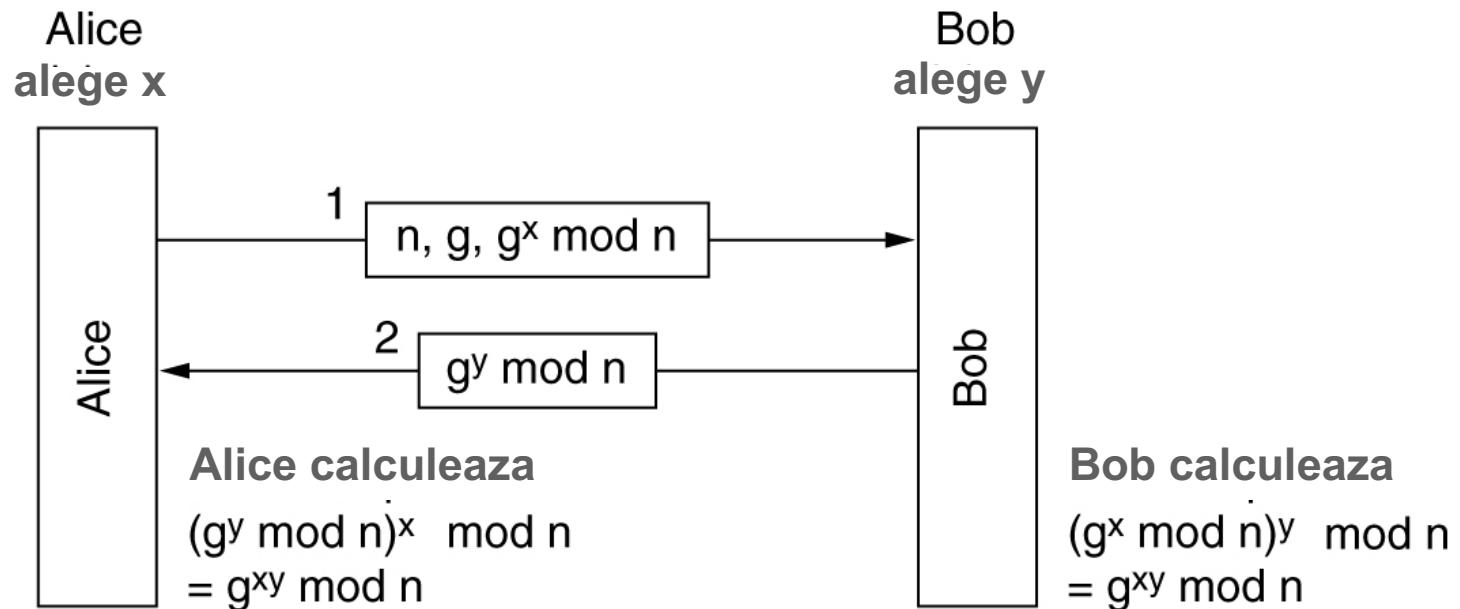


Alice și Bob **partajează** cheia  $K_{AB}$

Fiecare parte poate calcula rezumatul HMAC - Hashed Message Authentication Code – (deoarece conține doar **valori cunoscute**)  
- Hash-based Message Authentication Code (de ex. folosind SHA-1)

Trudy nu poate forța pe Alice sau Bob să cripteze sau să rezume o valoare impusă de ea

# Stabilire cheie partajata: Diffie-Hellman key exchange

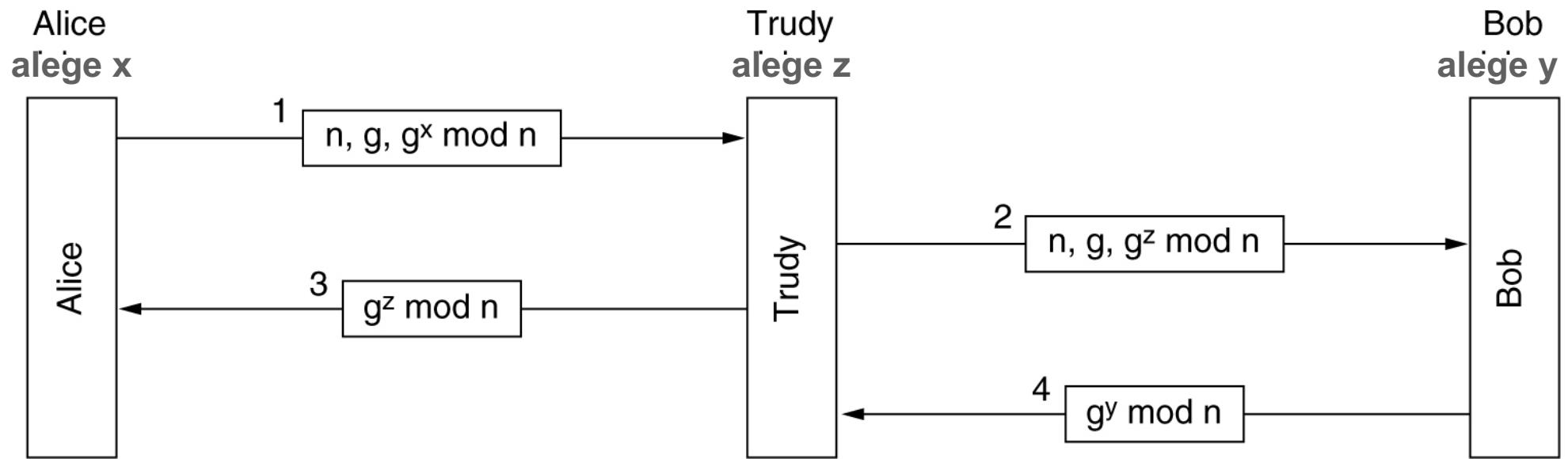


$n, g$  – numere mari  
 $n$  prim  
 $(n-1)/2$  prim

$x$  nu poate fi calculat din  $g^x \text{ mod } n$   
 $g^{xy} \text{ mod } n$  nu poate fi calculat din  $g^x \text{ mod } n$   
 și  $g^y \text{ mod } n$  cand  $n$  este mare

$g < n$  (generator) are proprietatea: orice  $p$  poate fi scris ca  $g^k \text{ mod } n$   
 adica: pentru fiecare  $p$  intre 1 si  $n-1$  inclusiv, exista o putere  $k$  a lui  $g$  astfel ca  
 $p = g^k \text{ mod } n$ .

# Atacul man-in-the-middle

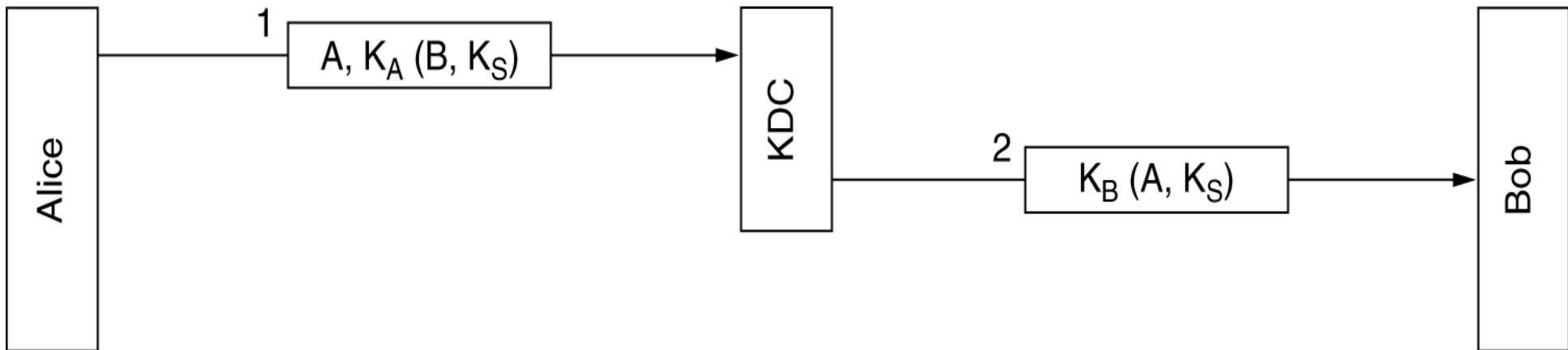


**Vulnerabilitate** –  $g$  și  $n$  sunt publici

- permite stabilirea a două chei: între Alice și Trudy -  $g^{xz} \text{ mod } n$  și între Trudy și Bob -  $g^{zy} \text{ mod } n$

**Rezolvare**: Alice și Bob semnează mesajele schimbate între ei  
Trudy nu poate modifica mesajele

# Autentificarea folosind Key Distribution Center



Alice si Bob folosesc un Centru de distributie a cheilor

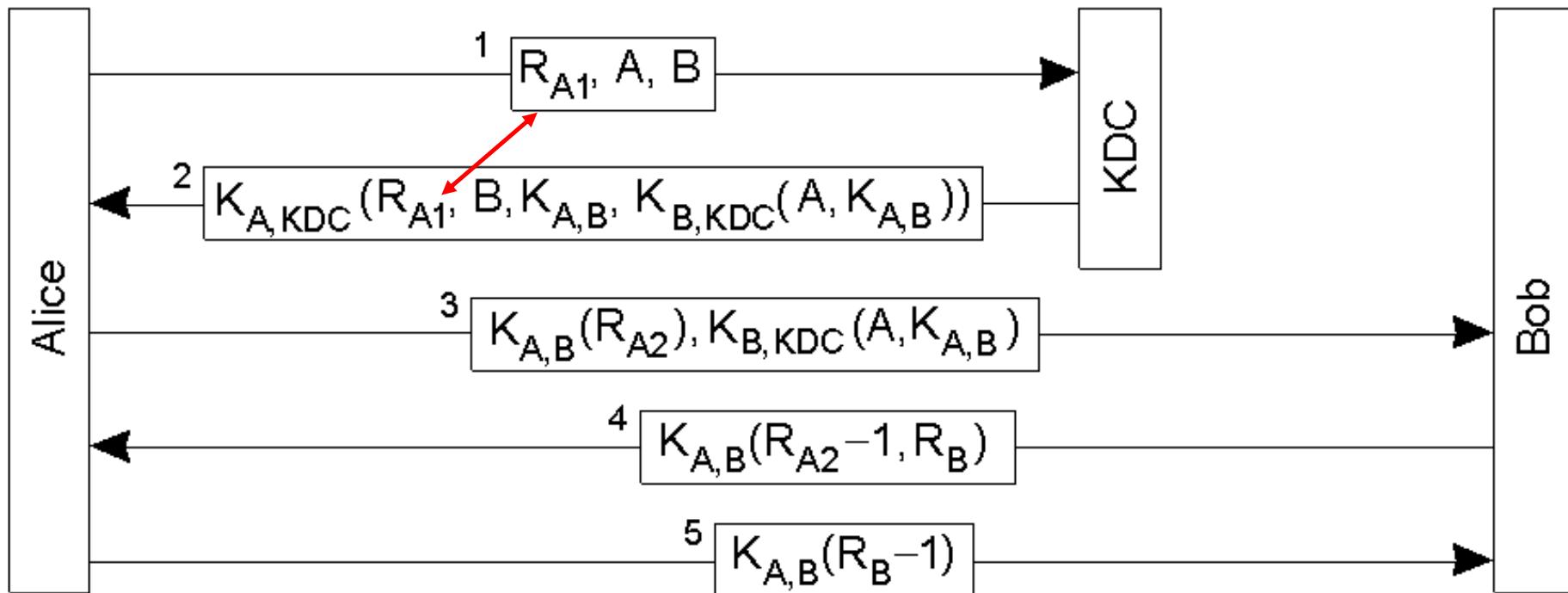
- in care au incredere
- cu care impart cheile secrete  $K_A$  respectiv  $K_B$

Prima incercare, vulnerabila la [replay attack](#)

Trudy **retransmite** mesajul 2 si

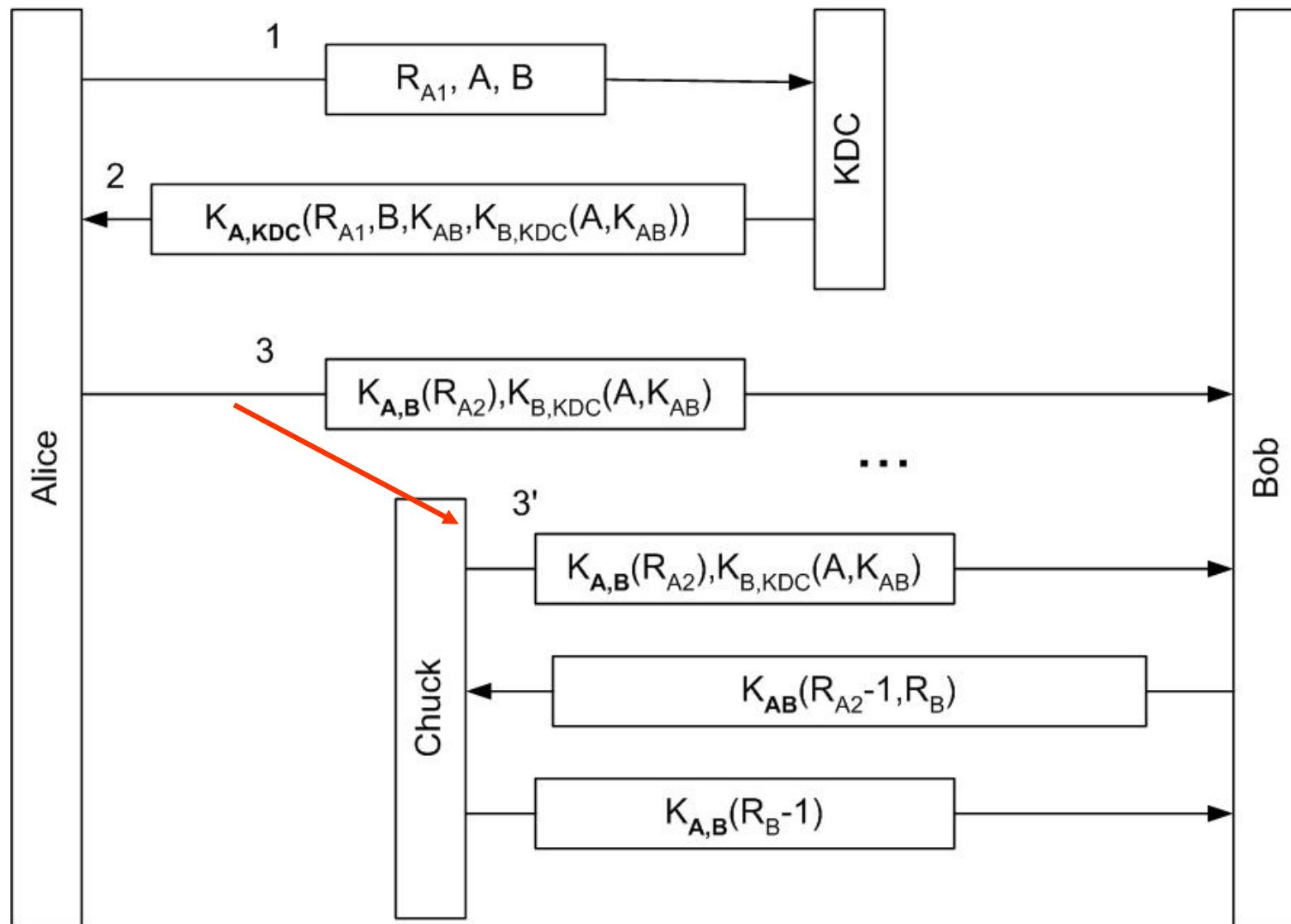
mesajul asociat cu el, criptat deja cu  $K_S$  (de ex. extragerea din contul lui Alice a unei sume de bani)

# Autentificarea cu protocolul Needham-Schroeder



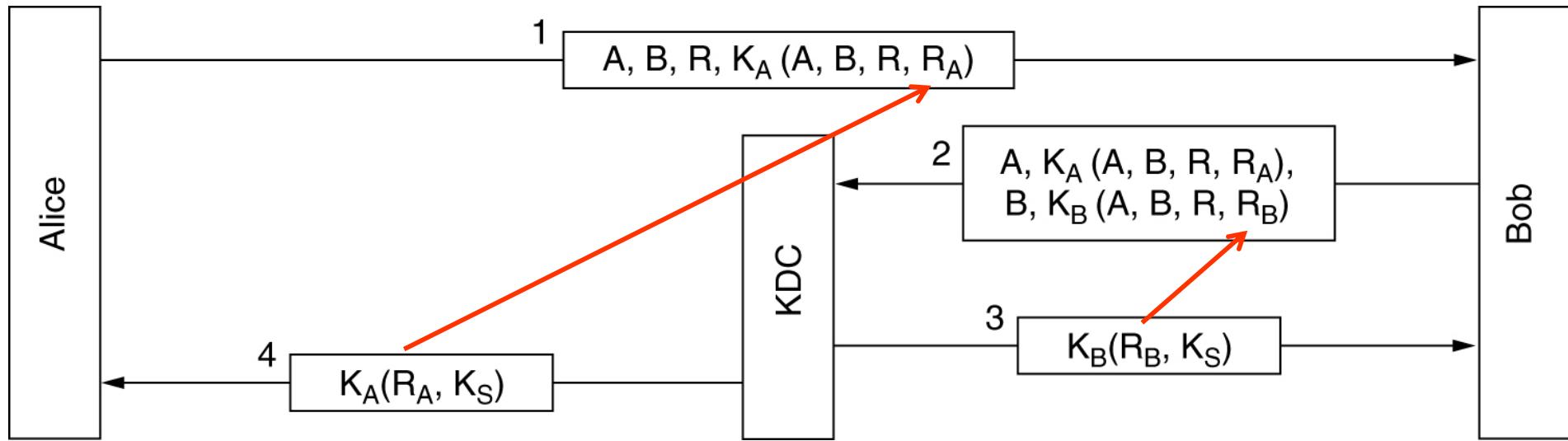
- foloseste **tichete** - ex.  $K_{B,KDC}(A, K_{AB})$ 
  - Alice nu poate intelege sau modifica tichetul, Bob poate
  - Bob capata incredere in cheia  $K_{AB}$  (care vine de la KDC)
- numere aleatoare (nonce) ex.  $R_{A1}$ , folosite contra atac prin **replica**
  - ex. Alice afla ca **mesajul 2 este un raspuns la 1**, nu un mesaj rejucat de Trudy

## Slabiciune Needham-Schroeder



Chuck afla cheia  $K_{AB}$  si rejoaca mesajul 3, pretinzând ca e Alice

# Autentificarea folosind Protocolul Otway-Rees



Protocolul Otway-Rees (simplificat).

KDC trimite cheia de sesiune  $K_S$  după ce verifica dacă identificatorul comun  $R$  apare în ambele parti criptate ale mesajului 2

$R_A, R_B$  – numere aleatoare folosite de KDC în mesajele 3 și 4 pentru a face legătura cu mesajele 1 și 2

**Problema:** Alice ar putea folosi cheia secreta înainte ca Bob să afle de ea



# Securitatea E-Mail - PGP – Pretty Good Privacy

Autor: Phil Zimmermann

Oferă **gama completa** de servicii de securitate

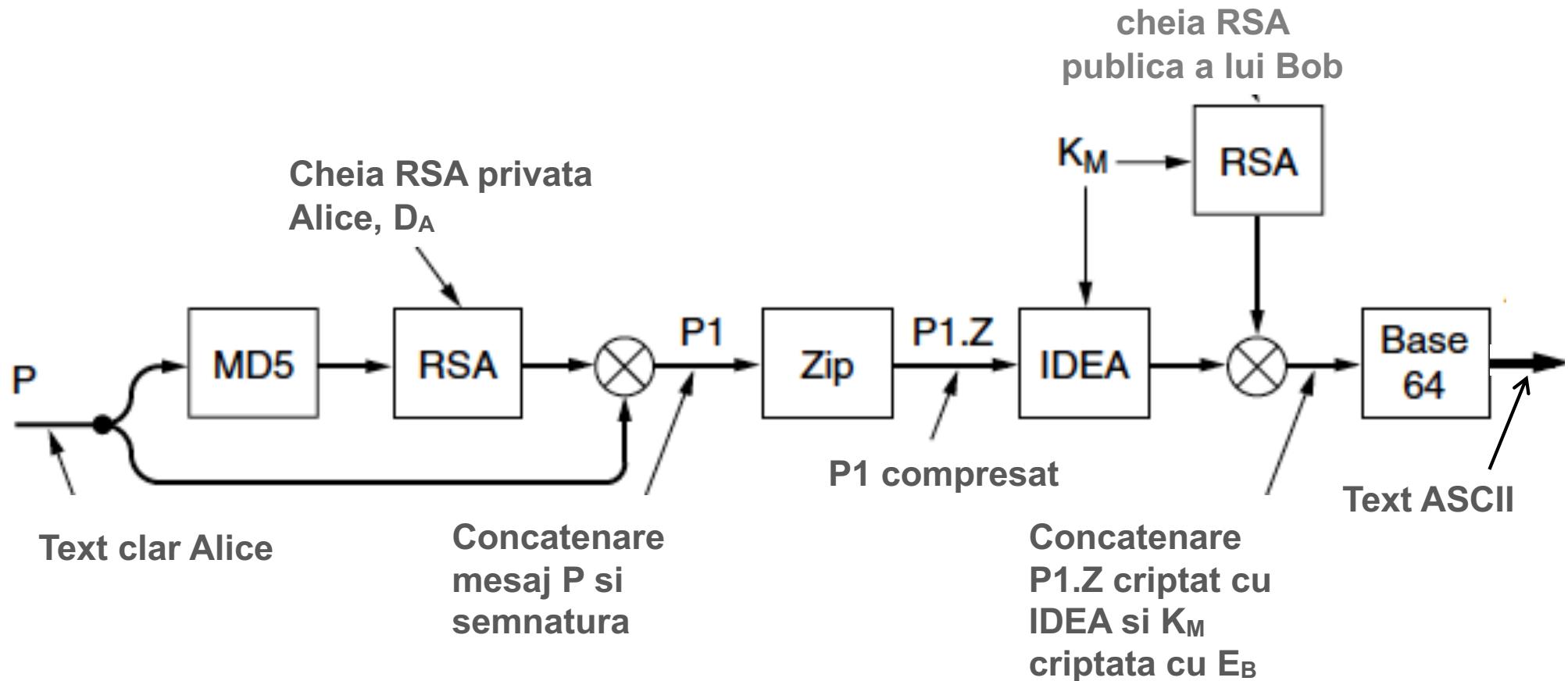
- intimitate (privacy)
- autentificare
- semnatura digitală (integritate)
- compresie

Intregul pachet PGP (inclusiv codul sursă) este **distribuit gratuit** pe Internet

**Criptează** date folosind IDEA (International Data Encryption Algorithm) – similar cu DES și AES

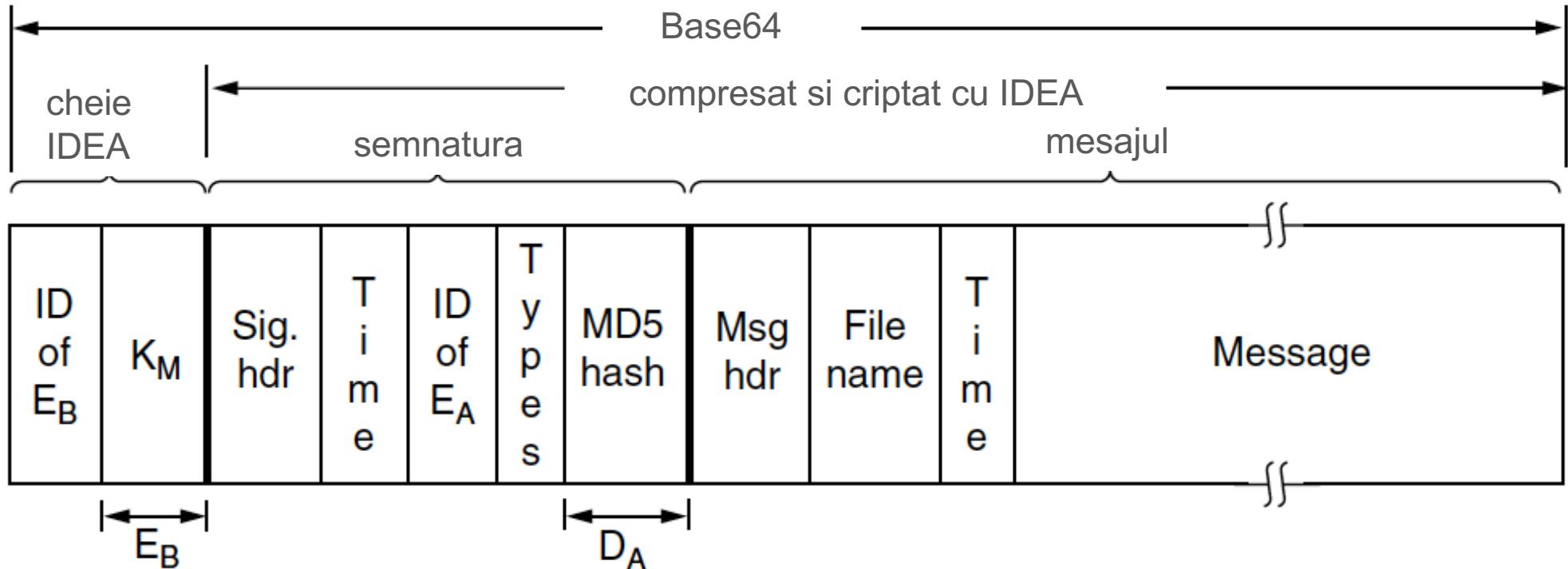
**Semnează** mesajele folosind RSA

# Folosirea PGP pentru a trimite un mesaj



$K_M$  cheie de sesiune 128-bit produsa din textul introdus de Alice

# PGP – Formatul mesajului



File name – nume implicit al fisierului de utilizat la receptie

Types – identifica algoritmul de criptare

ID of  $E_A$  – A poate avea mai multe perechi de chei publica/privata  $E_A/D_A$ ; fiecare pereche are un identificator ID (ultimii 64 biti ai cheii publice)

ID of  $E_B$  – fiecare B poate avea mai multe chei publice; fiecare cheie are un identificator, ID (64 biti) si un indicator de trust (cata incredere are A in aceasta cheie)



# Management chei

Foloseste două fisiere în care se păstrează

- **Private key ring** conține propriile perechi de chei (publică, privată) împreună cu identificatorii lor
- **Public key ring** conține perechi (**key, trust indicator**) pentru cheile publice ale partenerilor

Cheile private se țin criptate cu o parola specială

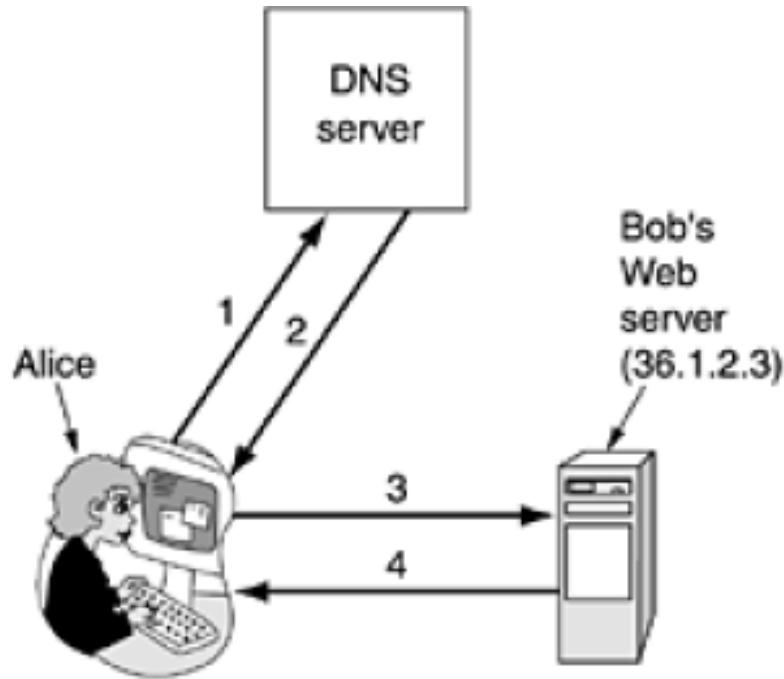
Versiunile actuale PGP folosesc certificate X.509



# Securitatea Web

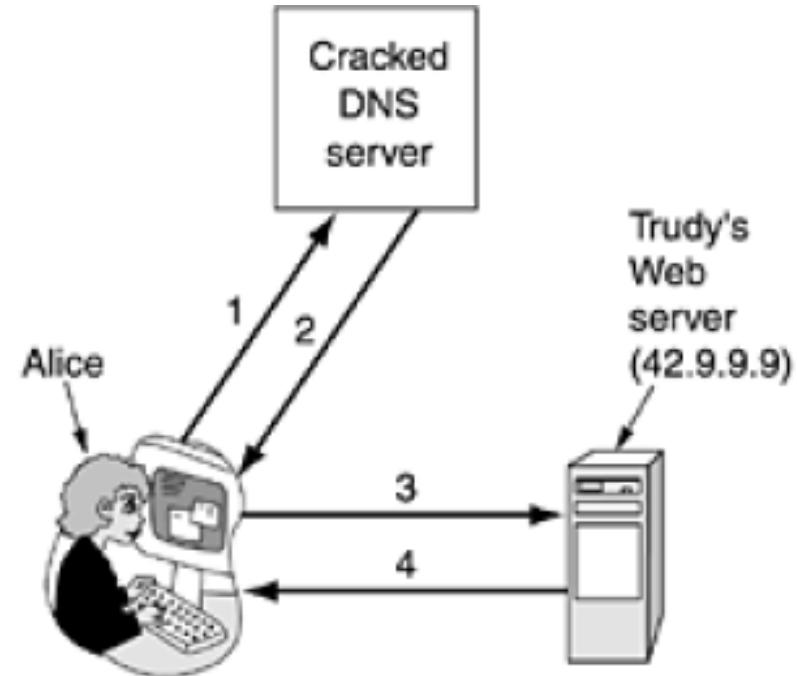
- Atacuri
  - înlocuire Home page
  - Denial-of-service
  - Citire mail-uri
  - Furt numere credit card
- Solutii
  - Secure Naming
  - SSL – The Secure Sockets Layer

# Necesitate Secure Naming



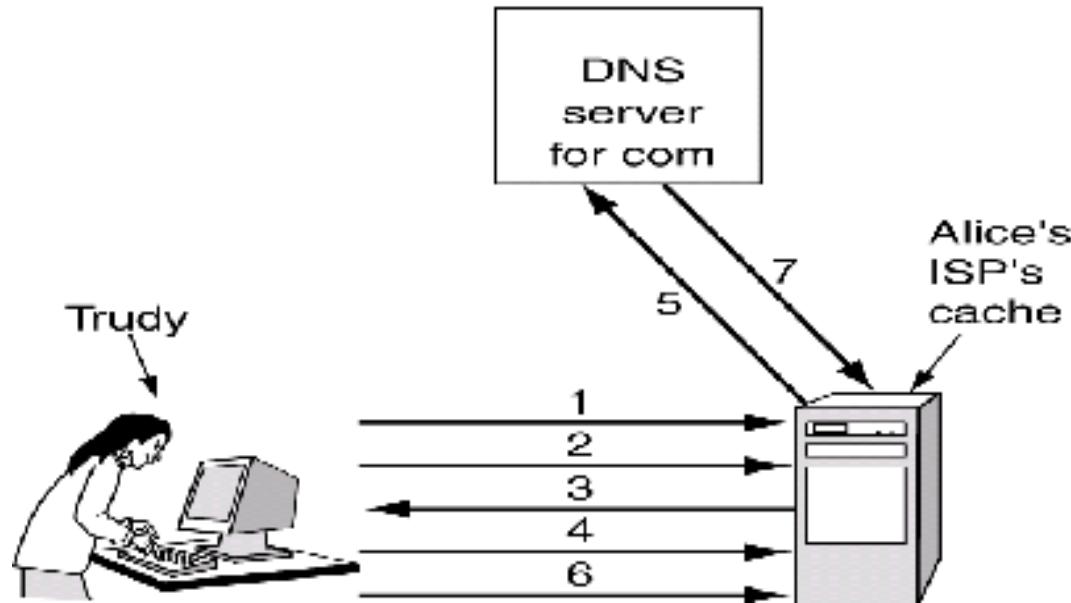
## Situatie Normala.

1. Da-mi adresa IP Bob
2. 36.1.2.3 (adr IP Bob)
3. GET index.html
4. Pagina home Bob



## Un atac bazat pe modificarea înregistrării lui Bob în DNS.

1. Da-mi adresa IP Bob
2. 42.9.9.9 (adr IP Trudy)
3. GET index.html
4. Pagina Bob falsificata de Trudy



## Trudy pacaleste ISP-ul lui Alice

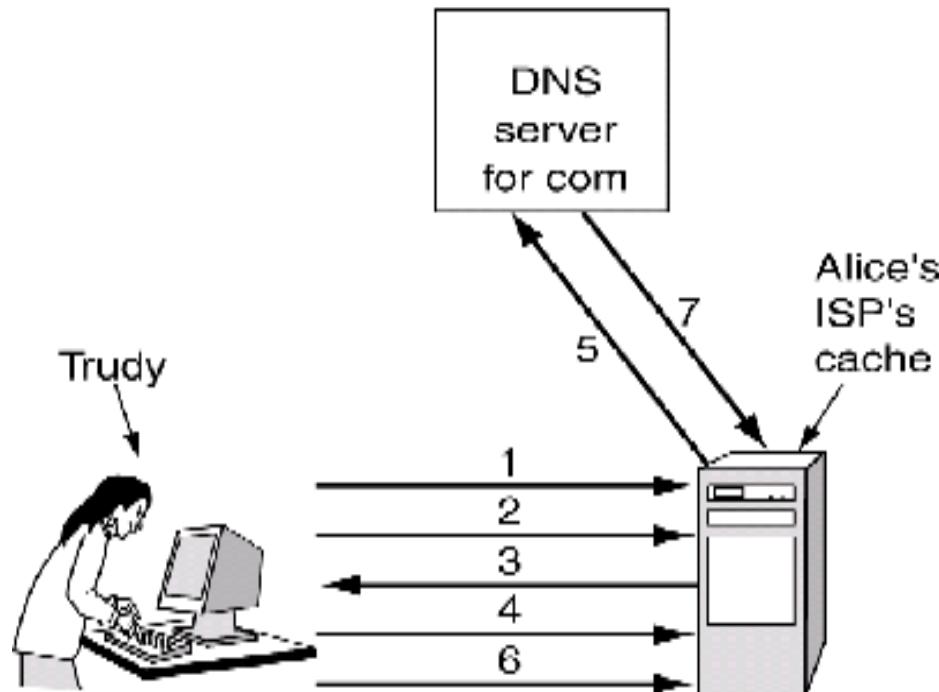
**Probleme:** ISP verifică adresa IP de la care vin răspunsurile DNS

- Trudy poate folosi adresa IP a unui server DNS de nivel înalt (care este publică) pentru a construi un răspuns fals

DNS se bazează pe UDP → DNS folosește

**sequence numbers** pentru a mări cererile și răspunsurile

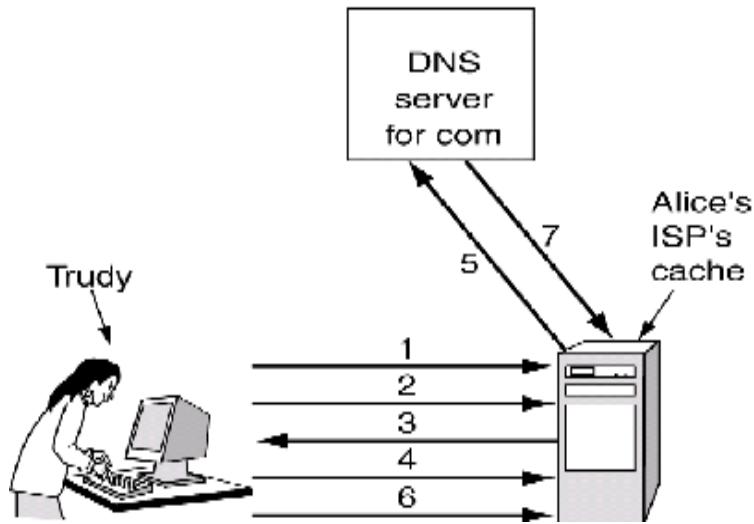
- Trudy înregistrează un domeniu ***trudy-the-intruder.com*** (IP 42.9.9.9) și
- Crează un server DNS ***dns.trudy-the-intruder.com*** (aceeași IP 42.9.9.9)



## Trudy pacaleste ISP-ul lui Alice (2)

1. Cere adresa ***foobar.trudy-the-intruder.com*** - ISP-ul lui Alice afla de la serverul **com** despre noul ***dns.trudy-the-intruder.com*** si il pune in cache
2. Cere ISP-ului adresa pentru ***www.trudy-the-intruder.com***
3. ISP intreaba DNS-ul lui Trudy; intrebarea are un numar de secenta, **n** asteptat de Trudy

## Trudy pacaleste ISP-ul lui Alice (3)



4. Repede, cere adresa **bob.com** (fortand ISP sa intrebe serverul **com** in pasul 5)
5. ISP transmite cererea cu nr secv n+1 catre serverul **com**
6. Trudy transmite repede un **raspuns fals** cu nr secv = **n+1**, adresa IP a serverului **com** drept sursa raspunsului si adresa sa 42.9.9.9 drept adresa lui Bob; raspunsul este considerat bun si este pus in cache-ul ISP
7. Cand soseste raspunsul adevarat, ISP il rejecteaza

Cand Alice cauta **bob.com** primeste **adresa falsa** din cache ISP



# Secure DNS

Inregistrările din DNS au forma

Domain name	Time to live	Class	Type	Value
bob.com.	86400	IN	A	36.1.2.3

## Pentru securitate

fiecarei zone DNS i se aloca o **pereche de chei** publica/privata

Se adauga doua noi tipuri de inregistrari

**KEY record** – cheia publică a unei zone, utilizator, host, etc.

**SIG record** - **hash** semnat al inregistrarilor A și KEY pentru verificarea autenticității.

Domain name	Time to live	Class	Type	Value
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C...



## Secure DNS (2)

Domain name	Time to live	Class	Type	Value
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C...

**Gruparea obtinuta se numeste RRSet (Resource Record Set)**

Clientii primesc de la DNS un RRS cu semnatura SIG  
aplica cheia publica a zonei pentru a decripta SIG  
calculeaza hash-ul pentru A si KEY  
compara cele doua valori (calculata si decriptata)



## Studiu individual

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

8.4 SEMNĂTURI DIGITALE

8.5 GESTIONAREA CHEILOR PUBLICE

8.6 SECURITATEA COMUNICAȚIEI

8.7 PROTOCOALE DE AUTENTIFICARE

8.8 CONFIDENTIALITATEA POȘTEI ELECTRONICE

8.9 SECURITATEA WEB-ULUI

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

8.4 DIGITAL SIGNATURES

8.5 MANAGEMENT OF PUBLIC KEYS

8.6 COMMUNICATION SECURITY

8.7 AUTHENTICATION PROTOCOLS

8.8 EMAIL SECURITY

8.9 WEB SECURITY



# Protocole de Securitate



# Cuprins

- Scopul securitatii si metode de rezolvare
- Modele criptografice cu chei simetrice si publice
- Cifrarea prin substitutie si transpozitie
- DES si AES
- RSA
- Analiza algoritmilor criptografici



# Scopul securității

- confidentialitatea
  - informația este disponibilă doar utilizatorilor autorizați
- integritatea
  - informația poate fi modificată doar de utilizatorii autorizați sau în modalitatea autorizată (mesajul primit nu a fost modificat în tranzit sau măsluit)
- disponibilitatea
  - accesul la informație al utilizatorilor autorizați nu este îngăduit (opusul este denial of service)

## Probleme derivează

- autentificarea
  - determinarea identității persoanei cu care schimbi mesaje înainte de a dezvălui informații importante
- autorizarea (controlul accesului)
  - protecția împotriva accesului ne-autorizat
- non-repudierea
  - transmitatorul nu poate nega transmiterea unui mesaj pe care un receptor l-a primit



# Metode de rezolvare

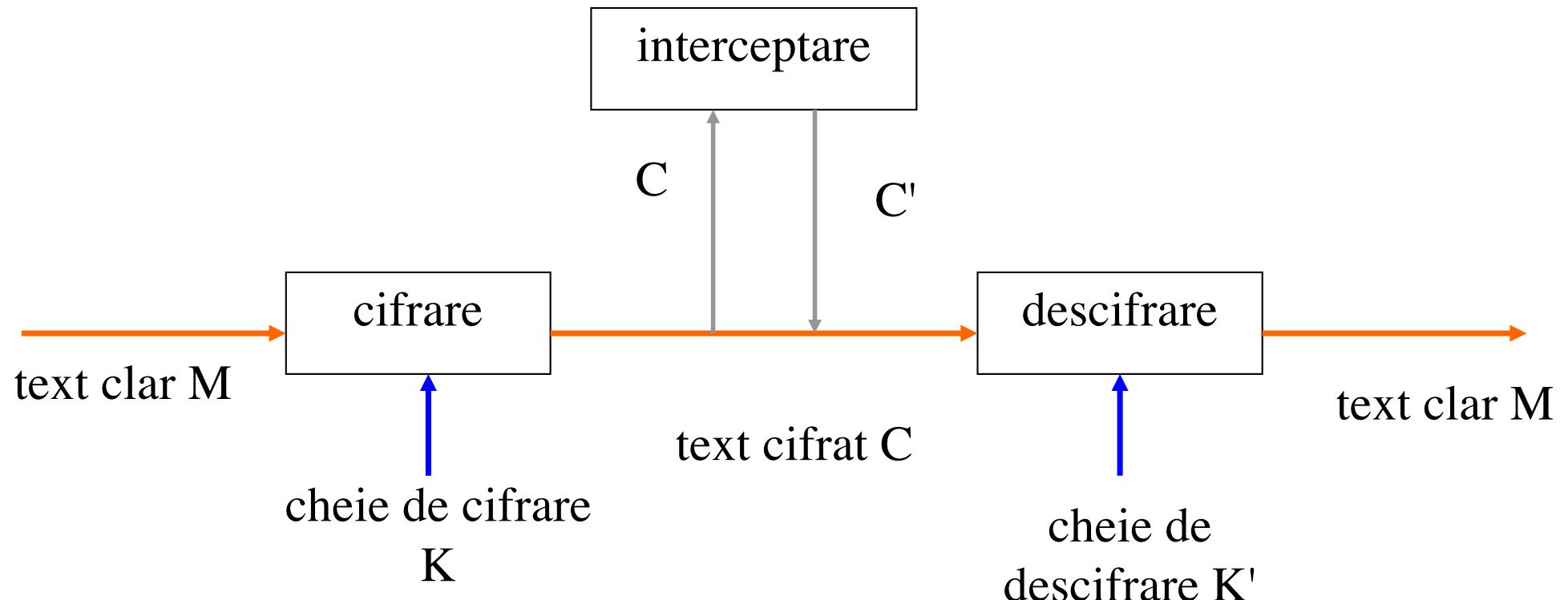
- Organizare
  - Algoritmi de criptare si hash
  - Mecanisme de securitate
    - criptare, rezumare (hash), semnatura digitală
  - Servicii si protocoale de securitate
- Securitatea in ierarhia de protocoale
  - considerata initial in nivelul prezentare al ISO OSI
  - este distribuita, in realitate, diverselor nivele
    - fizic – tuburi de securizare a liniilor de transmisie
    - legatura de date – legaturi criptate
    - retea – ziduri de protectie (firewalls), IPsec
    - transport – end-to-end security
    - aplicatie – autentificarea, non-repudierea



## Alte aspecte

- Politici de securitate.
- Control software (antivirus).
- Control hardware:
  - Cartele inteligente;
  - Biometrie.
- Control fizic (protectie).
- Educație.
- Măsuri legale.

## Modelul de bază al criptării



**confidentialitatea** - intrusul să nu poată reconstitui  $M$  din  $C$  (să nu poată descoperi cheia de descifrare  $K'$ ).

**integritatea** - intrusul să nu poată introduce un text cifrat  $C'$ , fără ca acest lucru să fie detectat (sa nu poată descoperi cheia de cifrare  $K$ ).



# Definiții

- Spargerea cifrurilor = **criptanaliză**.
- Proiectarea cifrurilor = **criptografie**.
- Ambele sunt subdomenii ale **criptologiei**.
- Transformarea  $F$  realizată la cifrarea unui mesaj:  
 $F : \{M\} \times \{K\} \rightarrow \{C\}$ , unde:
  - $\{M\}$  este mulțimea mesajelor;
  - $\{K\}$  este mulțimea cheilor;
  - $\{C\}$  este mulțimea criptogramelor.
- Operații:
  - Cifrarea (**Encryption**):  $C = E_k(M)$ .
  - Descifrarea (**Decryption**):  $M = D_{k'}(C)$ .
- Conotație de ordin practic!



# Problema criptanalistului

- Criptanaliză cu **text cifrat cunoscut**; se cunosc:
  - Un text cifrat;
  - Metoda de criptare;
  - Limbajul textului clar;
  - Subiectul;
  - Anumite cuvinte din text.
- Criptanaliză cu **text clar cunoscut**; se cunosc:
  - Un text clar;
  - Textul cifrat corespunzător;
  - Anumite cuvinte cheie (login).
- Criptanaliză cu **text clar ales**; se cunosc:
  - Mod cifrare anumite porțiuni de text;
  - Exemplu pentru o bază de date - modificare / efect.



# Caracteristicile sistemelor secrete

- **sistem neconditionat sigur**
  - rezistă la orice atac, indiferent de cantitatea de text cifrat interceptat
  - ex. [one time pad](#)
- **computational sigur sau tare**
  - nu poate fi spart printr-o analiză sistematică cu resursele de calcul disponibile.
- **sistem ideal**
  - indiferent de volumul textului cifrat care este interceptat, o criptogramă nu are o rezolvare unică, ci mai multe, cu probabilități appropriate



# Cerințe criptosisteme cu chei secrete

- Cerințe generale:
  - Cifrare și descifrare **eficiente** pentru toate cheile.
  - Sistem **ușor de folosit** (gasire chei de transformare).
  - Securitatea să **depindă de chei**, nu de algoritm.
- Cerințe specifice pentru **confidențialitate**: să fie imposibil computațional ca un criptanalist să determine **sistematic**:
  - Transformarea  $D_k$  din C, chiar dacă ar cunoaște M.
  - M din C (fără a cunoaste  $D_k$ ).
- Cerințe specifice pentru **integritate**: să fie imposibil computațional ca un criptanalist să determine **sistematic**:
  - Transformarea  $E_k$ , din C, chiar dacă ar cunoaște M.
  - Cifrul C' astfel ca  $D_k(C')$  să fie un mesaj valid (fără a cunoaște  $E_k$ ).

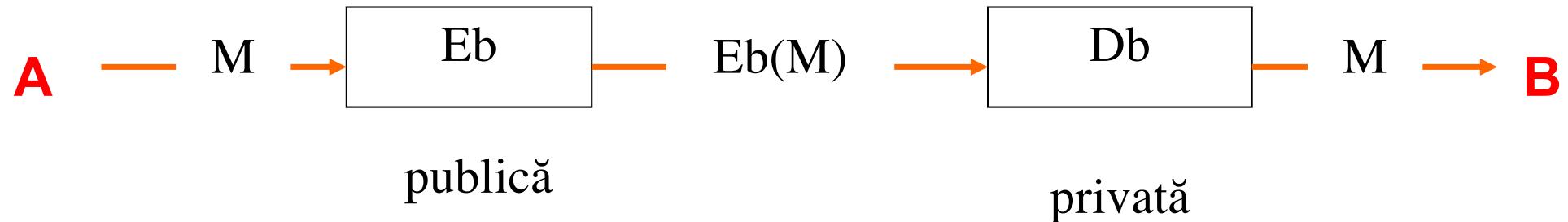


# Modelul criptografic cu chei publice

- Sistemele criptografice:
  - Simetrice.
  - Asimetrice:
    - Propuse de Diffie și Hellman în 1976.
    - Chei diferite de cifrare E și descifrare D cu proprietatea
      - $D(E(M)) = M$ .
    - Nu se pot deduce (ușor) una din alta, mai precis:
      - Este extrem de greu să se deducă D din E;
      - D nu poate fi "spart" prin criptanaliză cu text clar ales.



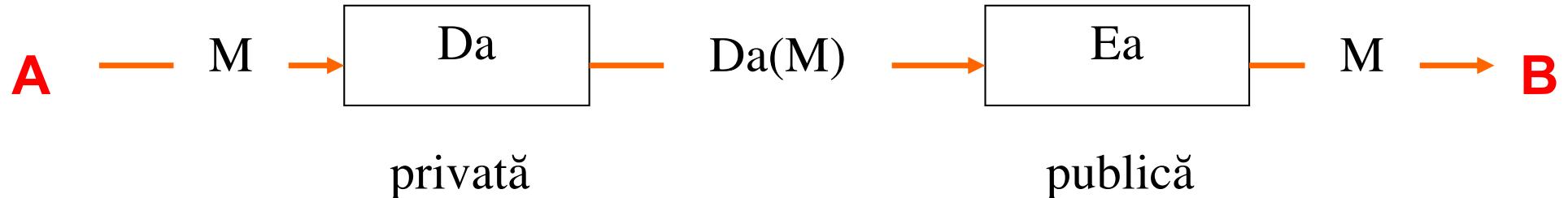
# Schema de confidențialitate



- Într-un sistem asimetric, un utilizator B:
  - Face publică cheia (transformarea)  $E_b$  de cifrare.
  - Păstrează secretă cheia (transformarea)  $D_b$  de descifrare.
- Se asigură **confidentialitatea**
  - doar B, care are cheia privată  $D_b$  poate intelege mesajul M



# Schema de integritate



- Pentru integritate / autentificare:
  - condiția necesară este ca transformările  $E_a$  și  $D_a$  să comute, adică

$$E_a(D_a(M)) = D_a(E_a(M)) = M.$$

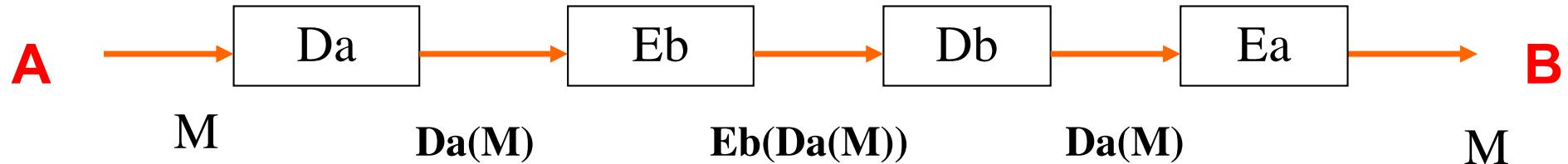
Cheia **Da** se foloseste pentru **criptarea** mesajului **M**

Se asigură **integritatea**

- Oricine poate folosi cheia publică **Ea** pentru a decripta mesajul
- Nimeni nu poate modifica mesajul **M** deoarece nu cunoaste cheia privată **Da**



## Schema de autentificare



autentificare  $M$  este criptat mai intai cu **cheia privata** a lui A

- $Da(M)$  este un fel de “semnatura” a mesajului

rezultatul este apoi criptat cu **cheia publica** a lui B

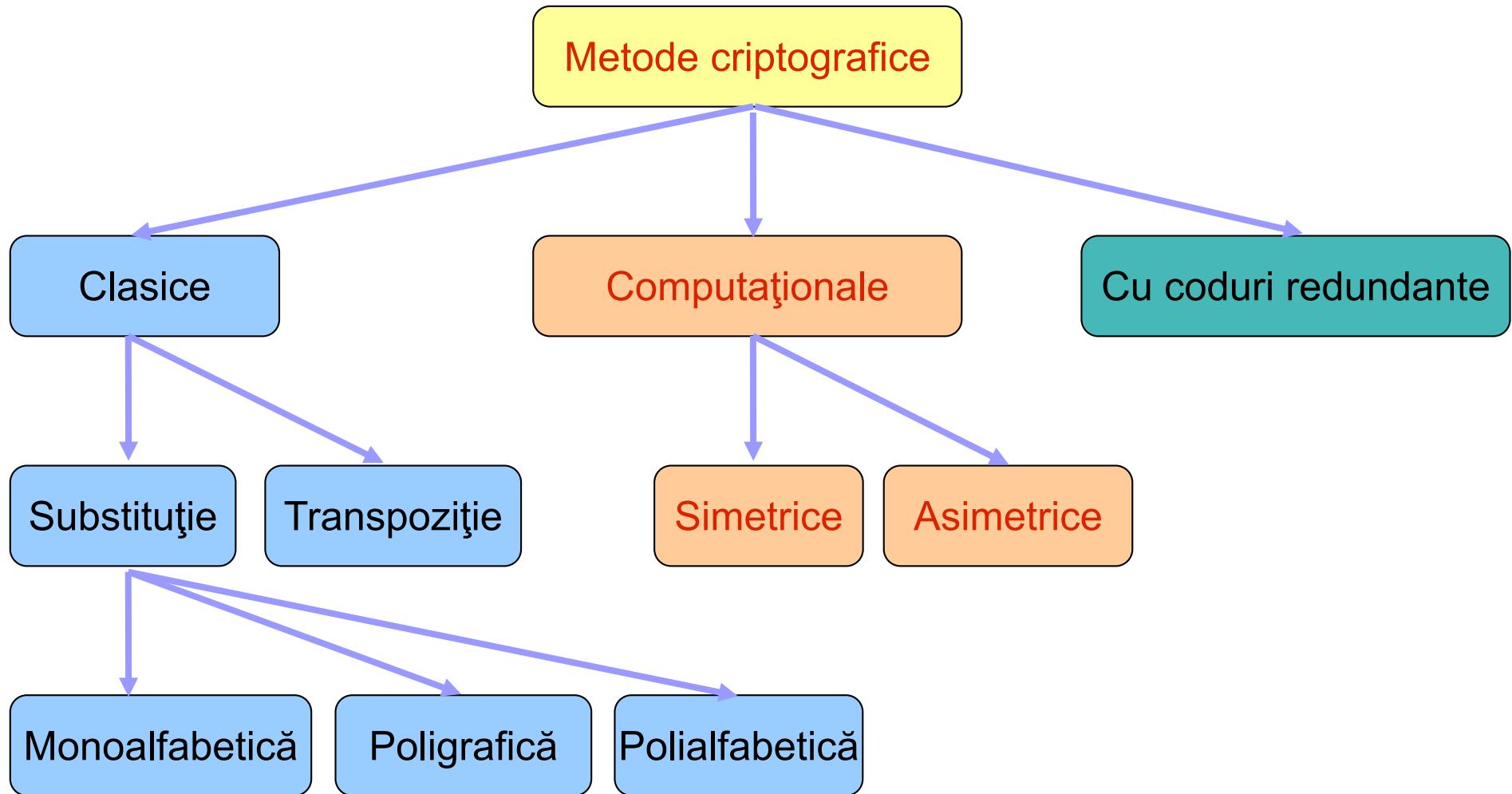
Se asigura că A este sursa mesajului

și că mesajul este confidențial

ne-repudiere folosind perechea  $Da(M)$  și  $M$ , B poate demonstra ca a primit mesajul de la A



# Clasificare generală





# Cifrarea prin substituție

## Cifrul lui Cezar (substituție monoalfabetică)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

textul clar: **CRYPTOGRAFIE**  
 text cifrat: **FULSWRJUDILH**

fiecare literă este  
 înlocuită de litera aflată  
 la distanța 3 de ea

Relatia de calcul

$$c[i] = (m[i] + 3) \bmod 26$$

În general

$$c[i] = (a \cdot m[i] + b) \bmod n.$$



## Substitutia polialfabetică (Vigenere)

foloseste 36 de cifruri Cezar si o **cheie** de cifrare de lungime l  
fiecare litera din cheie = **substitutul literei A** din textul clar

Exemplu: cheia POLIGRAF

<b>POLIGRAF</b>	<b>POLIGRAG</b>	<b>POLIGRAF</b>	<b>POLIGRAF</b>	<b>POLI</b>	<b>cheie</b>
<b>AFOSTODATA</b>	<b>CANPOVE</b>	<b>STIAFOS</b>	<b>TACANIC</b>	<b>IODATA</b>	<b>clar</b>
<b>PTZAZFDF</b>	<b>IONITGOATGE</b>	<b>QGWOXIQ</b>	<b>LVOTITSOE</b>	<b>I</b>	<b>cifrat</b>

Litera O din cheie substituie A din textul clar;  
Litera F (situata la 5 pozitii de A) este inlocuita de T (aflata la 5 pozitii de O)



## Cifrarea prin transpozitie

Modifică ordinea caracterelor. Uzual:

- textul clar este dispus în liniile successive ale unei matrice și
- parcurgerea acesteia după o anumită regulă pentru stabilirea noii succesiuni de caractere.

Exemplu

- caracterele dispuse pe linii sunt citite pe coloane,
- ordinea coloanelor este dată de ordinea alfabetică a literelor unei chei.

cheie: **POLIGRAF**

ordine: **76543812**

text clar: **AFOSTODATACANPOVESTIAFOSTCANICIO**

**POLIGRAF**

**AFOSTODA**

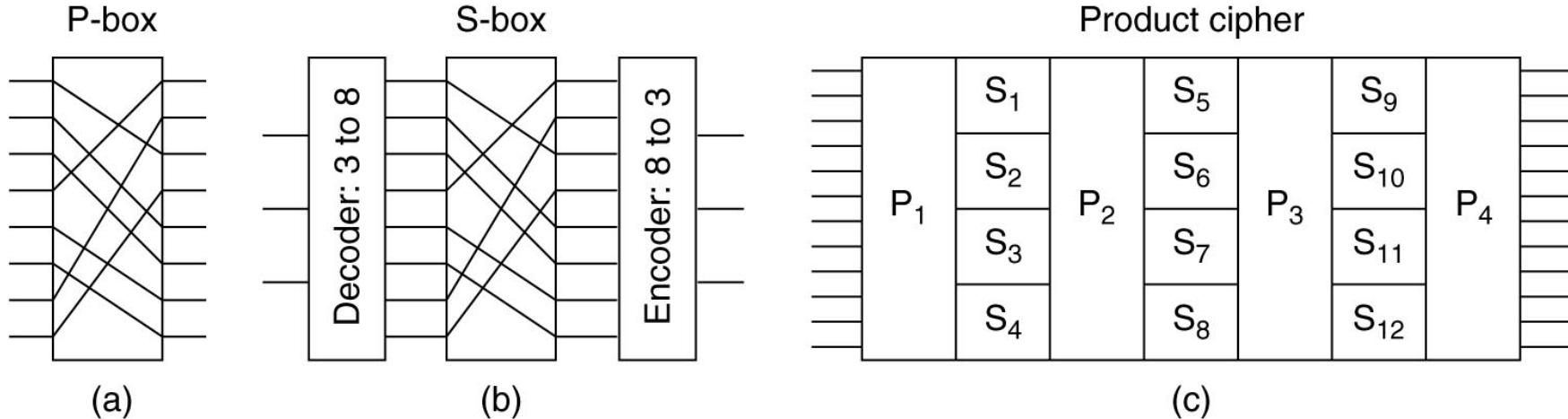
**TACANPOV**

**ESTIAFOS**

**TCANICIO**

text cifrat: **DOOIAVSOTNAISAINOCTAFASCATETOPFC**

# Cifruri produs



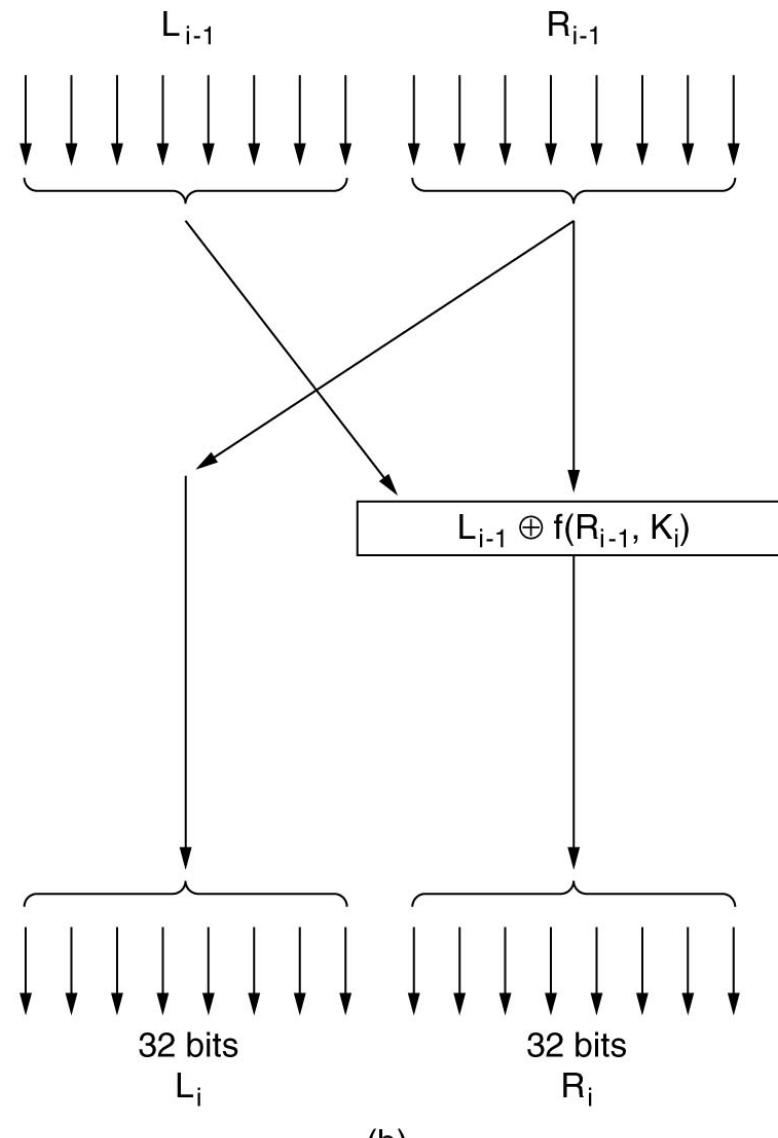
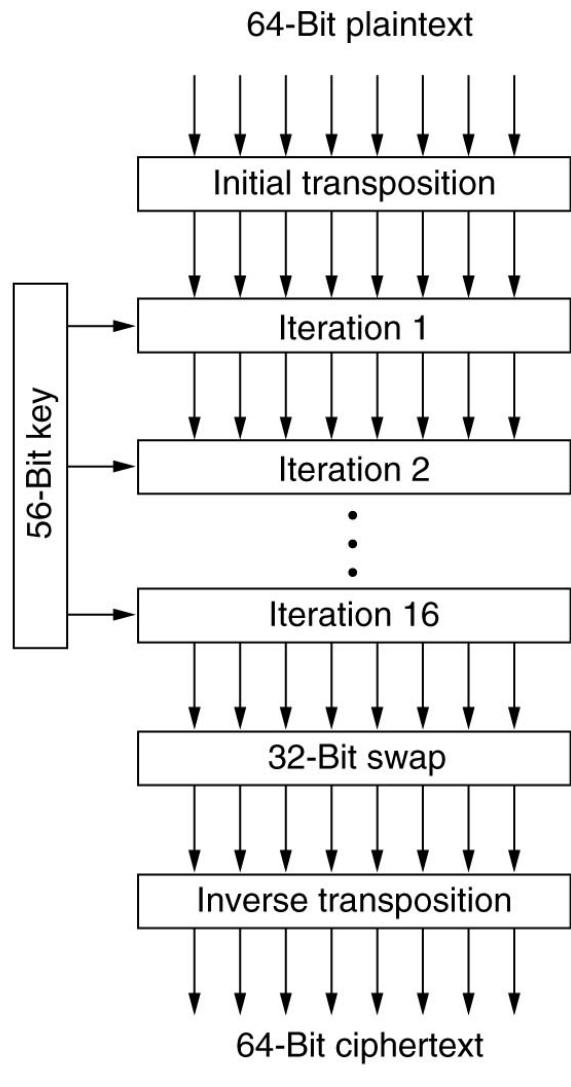
**Principii** pentru a obține o securitate mai mare:

- compune două cifruri "slabe", complementare
  - P-box – permutare (transpozitie) - asigură **difuzia**
  - S-box – substitutie - asigură **confuzia**
- repetă aplicarea permutării și substituției

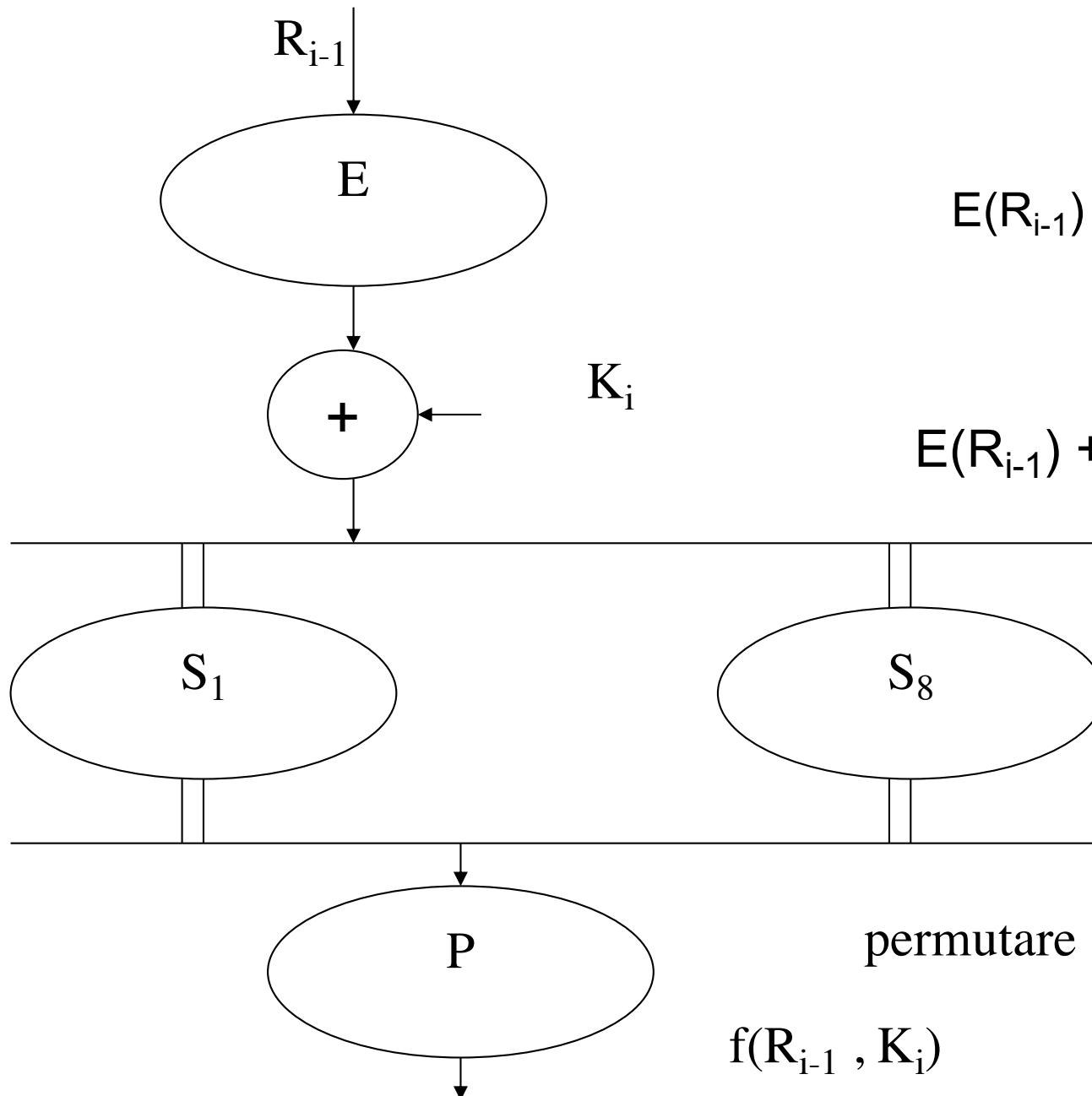
# DES (Data Encryption Standard)

## Schimă generală

O iterare



# Calculul lui $f(R_{i-1}, k_i)$



32 biti

expandare la 48 de biti  
prin repetarea unora

 $E(R_{i-1})$ 

$$E(R_{i-1}) + K_i \Rightarrow B_1 B_2 \dots B_8$$

8 blocuri a 6 biti sunt  
intrari in S-boxes

 $S_j(B_j)$ 

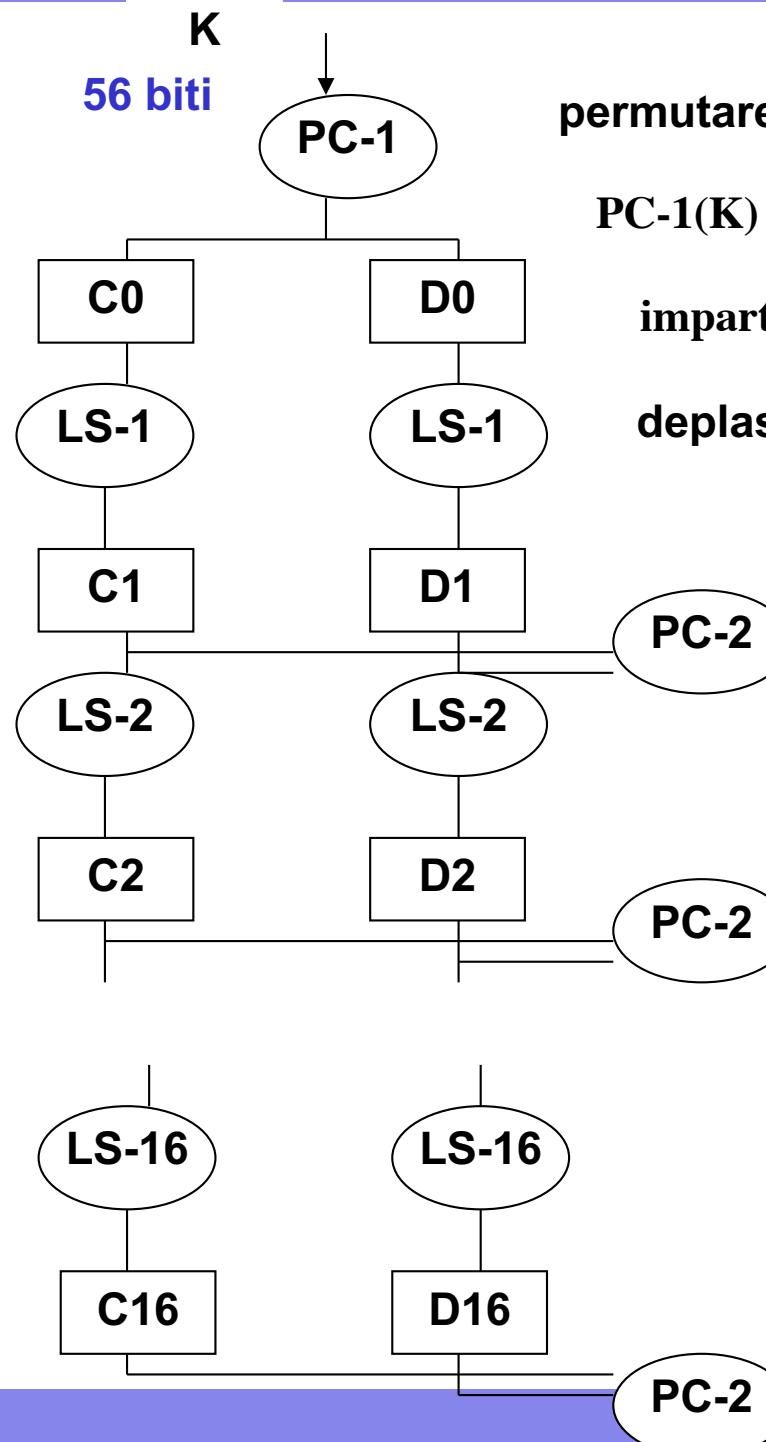
Fiecare  $S_j$  produce un  
cod de 4 biti

permutare  $P(S_1(B_1), \dots, S_8(B_8))$

 $f(R_{i-1}, K_i)$ 

32 biti

## Calculul cheilor





# Comentarii

- Transpozițiile, expandările, substituțiile sunt definite în DES
- Același algoritm folosit la criptare și decriptare

La criptare:  $L_j = R_{j-1}$

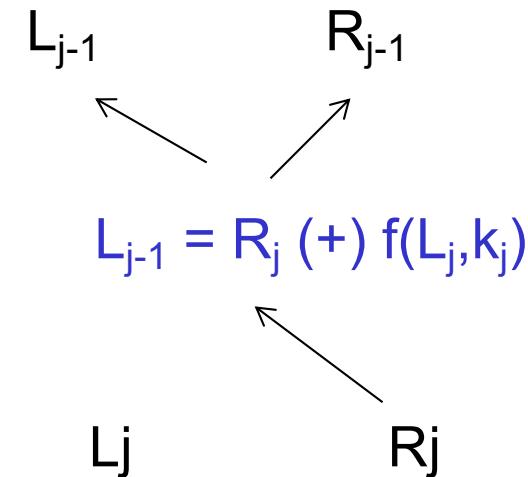
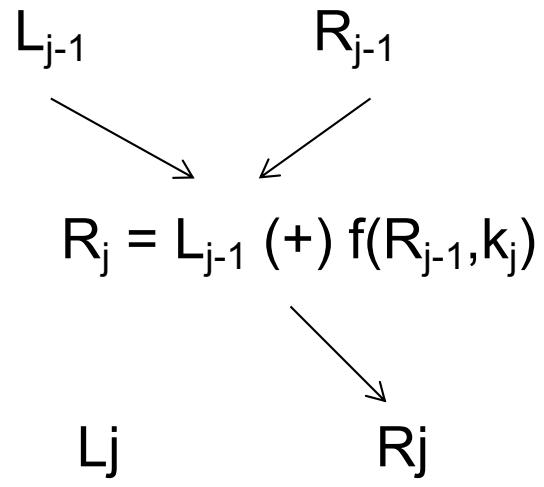
$$R_j = L_{j-1} (+) f(R_{j-1}, k_j)$$

De unde:  $R_{j-1} = L_j$

$$L_{j-1} = R_j (+) f(R_{j-1}, k_j)$$

și  $L_{j-1} = R_j (+) f(L_j, k_j)$

Decriptare = ordine inversă criptării (cu cheile în ordinea  $k_{16} - k_1$ )





## Comentarii (2)

- Elementele cheie ale algoritmului nu au fost făcute publice
  - Controverse
    - Există trape (capcane) care să ușureze decriptarea de către NSA?
      - NSA declară că NU.
      - Descoperirea și folosirea unei astfel de trape de un criptanalist răuvoitor
        - unele detalii despre S-box au fost dezvăluite de NSA



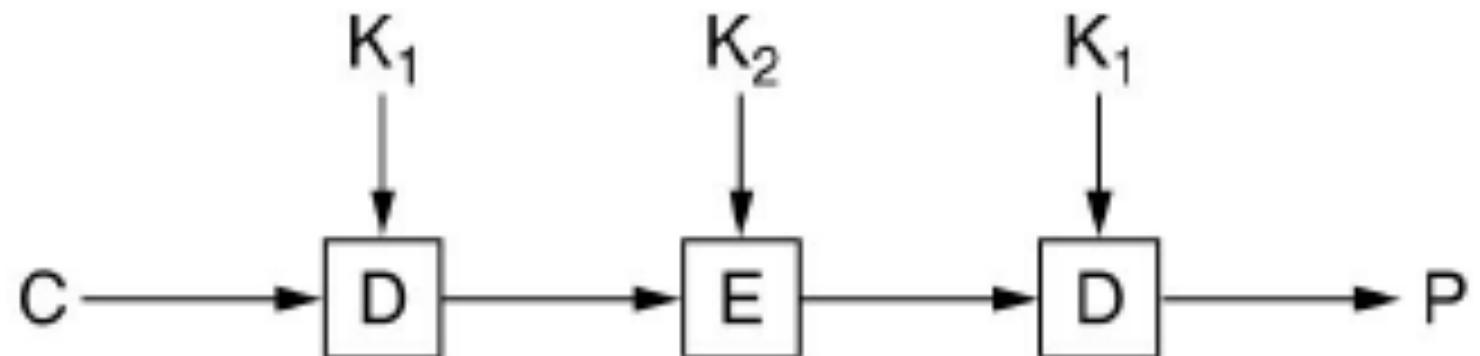
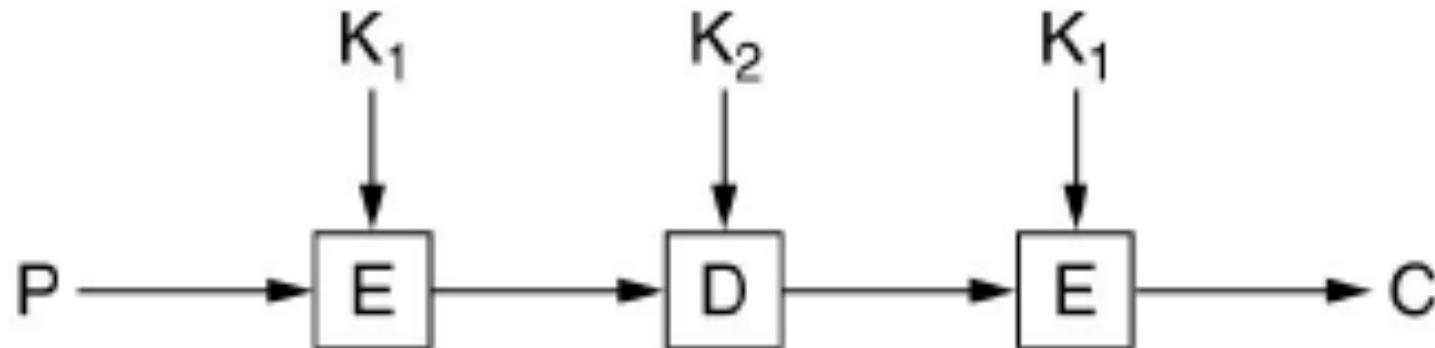
## Comentarii (3)

- Număr de iterații (16) sunt suficiente pentru **difuzie**?
  - Experimental, după 8 iterații nu se mai văd dependențe ale bițiilor de ieșire de grupuri de biți din intrare
- Lungimea cheii
  - Cheie DES de 56 biți spartă prin forță brută (4 luni \* 3500 mașini) în 1997
  - Dar, nu au fost raportate deficiențe în algoritm
  - Triple DES “mărește” lungimea cheii

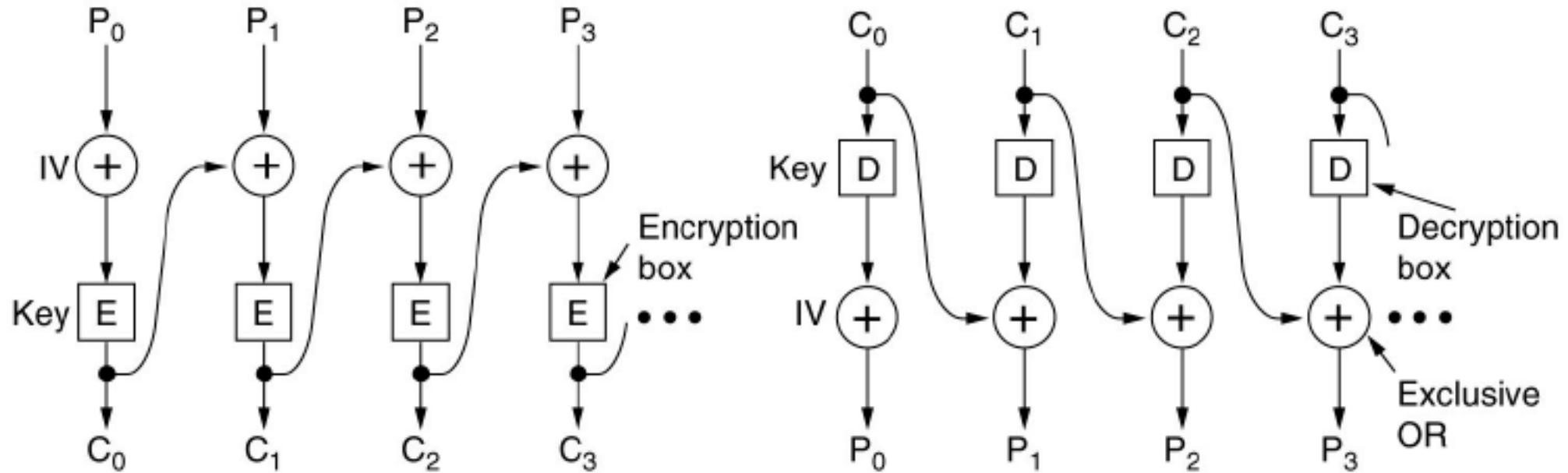
# Triplu DES

“Creste” lungimea cheii folosind

- 2 chei
- 3 runde de criptare / decriptare



# Inlantuirea blocurilor cifrate: CBC - Cipher Block Chaining



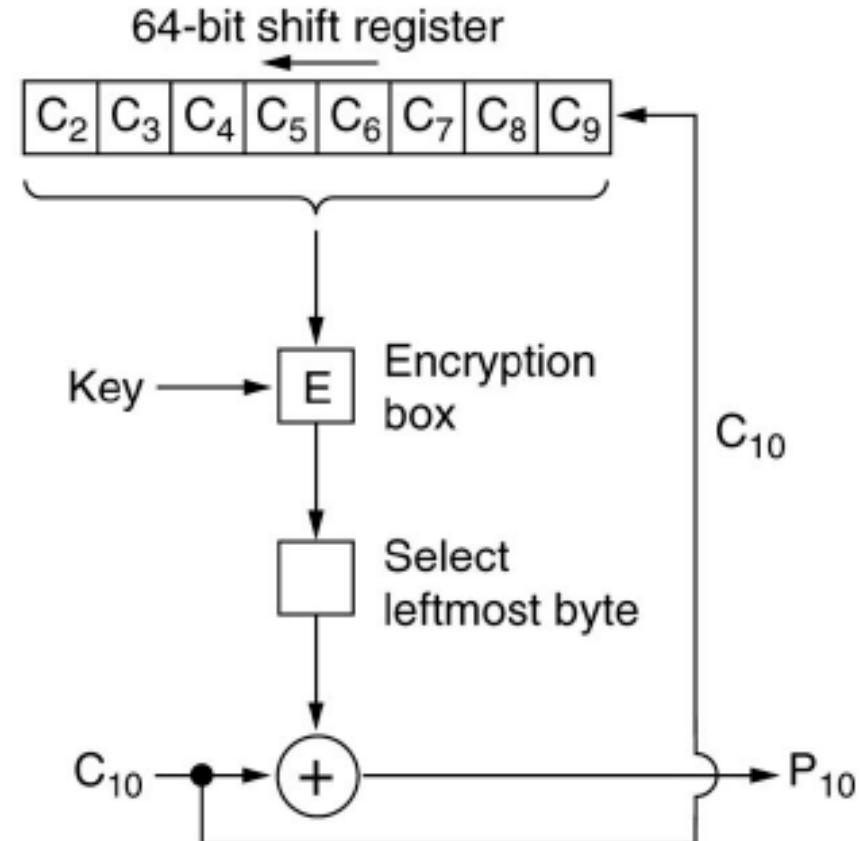
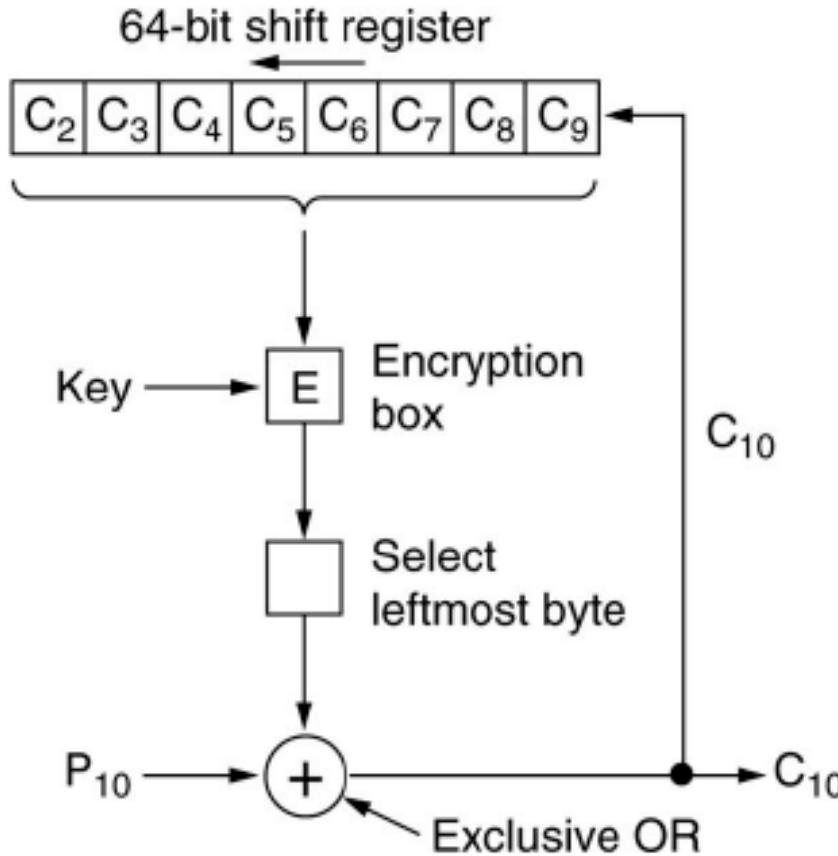
**Key** – cheie secreta

**IV** – Initialization Vector

- ales aleator, acelasi pentru criptare si decriptare
- folosit pentru combinarea cu primul bloc de text clar

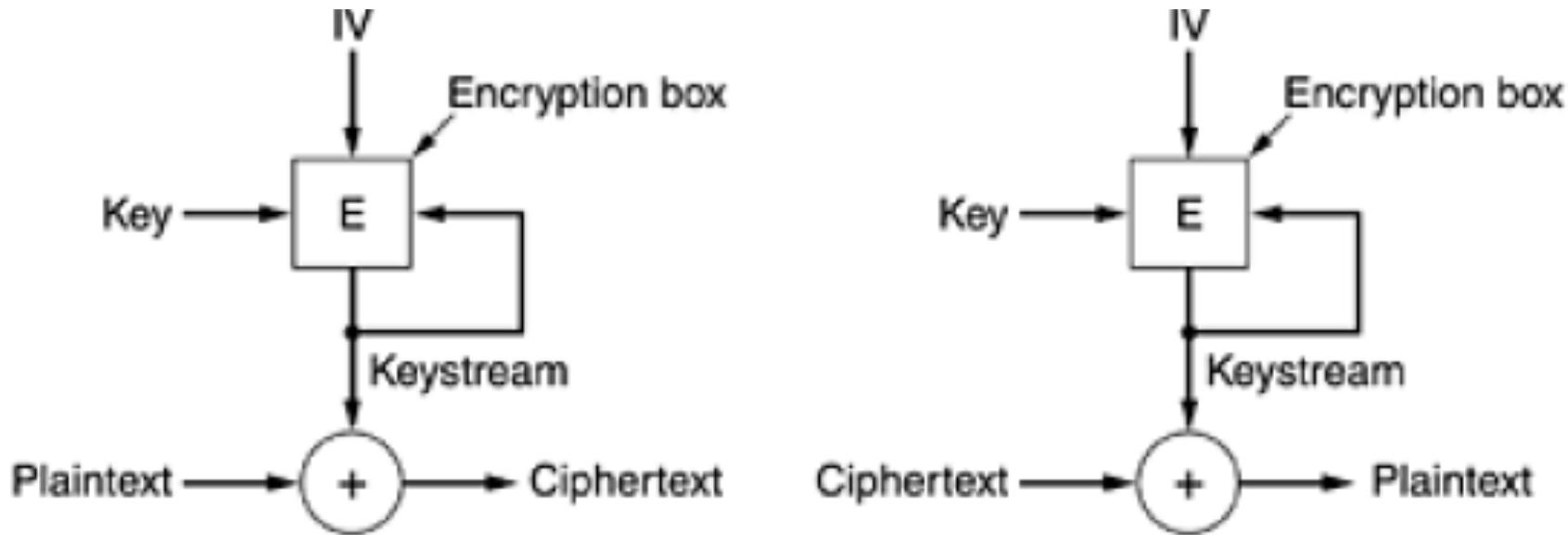
Urmarea: un acelasi text clar repetat in mesaj va fi criptat diferit

# Reactie cifrata: CFB - Cipher-Feed Back



Foloseste un **Initialization Vector** ca prima valoare în **Registrul de deplasare**  
 O eroare de un bit în criptogramă conduce la decriptarea eronată a 8 octetii  
 – generati în pasii în care bitul se află în registrul de deplasare

# Cifrarea secvențială (Stream Cipher)



Foloseste un **Initialization Vector** ca intrare DES pentru a genera **prima cheie** de criptare/decritare, care este folosita ca intrare DES pentru a genera **a doua cheie** etc.

**Sirul de chei** generat este **combinat XOR** cu textul clar pentru a produce criptograma

Sirul de chei depinde doar de cheia principala **Key** si de vectorul de initializare **IV**

→ Nu trebuie refolosita repetat aceeasi pereche (Key, IV)



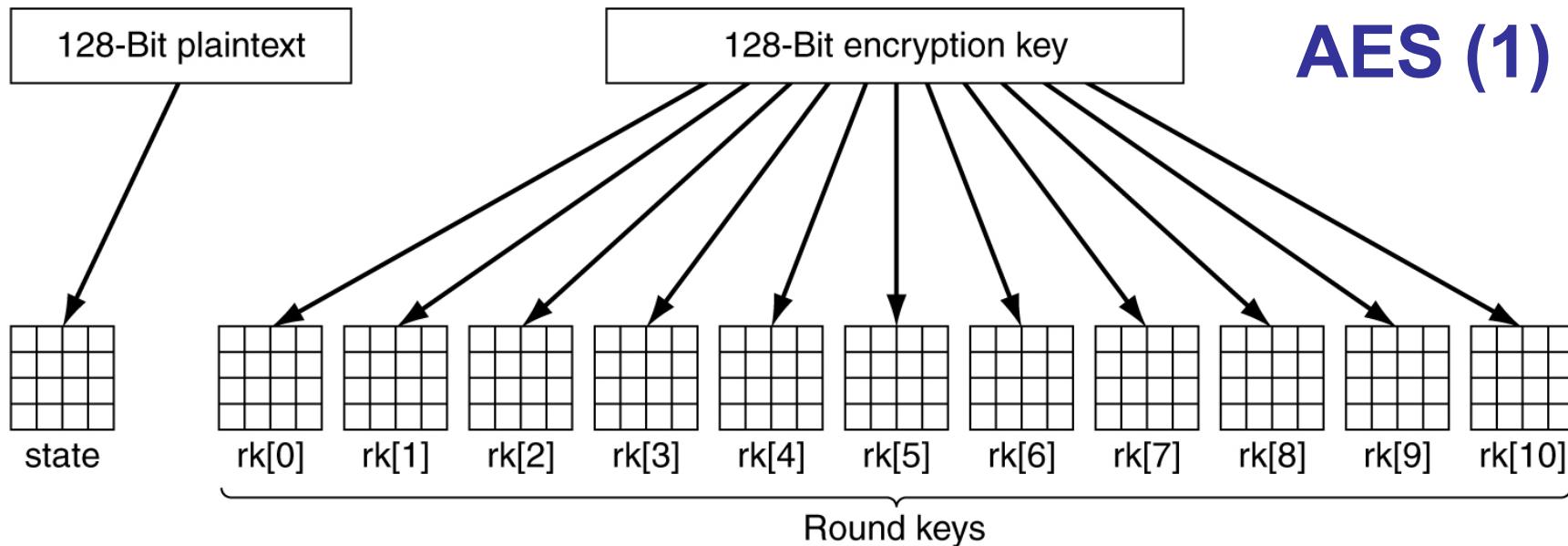
# AES – Advanced Encryption Standard

Regulile concursului organizat de NIST (ianuarie 1997) erau:

1. Algoritmul trebuie să fie un **cifru bloc simetric**.
2. Tot proiectul trebuie să fie public
3. Trebuie să fie suportate **chei** de 128, 192, și de 256 biți
4. Trebuie să fie posibile atât **implementări** hardware cât și software
5. Algoritmul trebuie să fie public sau oferit cu licență nediscriminatorie.

Finaliștii și scorurile lor au fost următoarele:

1. **Rijndael** (din partea lui Joan Daemen și Vincent Rijmen, 86 voturi)
2. Serpent (din partea lui Ross Anderson, Eli Biham și Lars Knudsen, 59 voturi)
3. Twofish (din partea unei echipe condusă de Bruce Schneier, 31 voturi)
4. RC6 (din partea RSA Laboratories, 23 voturi)
5. MARS (din partea IBM, 13 voturi)



Foloseste o matrice de **stare** de  $4 \times 4$  octeți (bloc de 128 biți)  
si mai multe **chei de runda** fabricate din cheia de baza

Numarul de runde **n** depinde de lungimea cheii

$n=10$  pentru cheie de lungime 128; 12/ 192, 14/ 256

Initial

- se copiază un bloc de 128 biti text clar în **state**
- se face XOR între **state** și cheia **rk[0]**



# AES (2)

In fiecare runda se fac patru operații

**substituție** – la nivel octet, folosește tabel substituție

**rotate\_rows** – prin deplasare circulară la stânga la nivel octet

1    5    9    13

2    6    10    14

3    7    11    15

4    8    12    16

1    5    9    13

6    10    14    2

11    15    3    7

16    4    8    12



**mix\_columns** – elementele unei coloane sunt înmulțite cu o matrice

$$\begin{array}{c|c} s'0i & \left| \begin{array}{cccc} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{array} \right| \\ s'1i & \left| \begin{array}{c} s0i \\ s1i \\ s2i \\ s3i \end{array} \right| \\ s'2i & \\ s'3i & \end{array}$$

**xor\_roundkey\_into\_state** – combinare cu o cheie de rundă  $rk[i]$

Rijndael definit în **câmp Galois  $G(2^8)$**  prin polinomul  $P = x^8+x^4+x^3+x+1$

număr = coeficienții unui polinom

$$\text{Ex. } 23_{(10)} = 10111_{(2)} \text{ este polinomul} \quad \begin{aligned} &1*x^4 + 0*x^3 + 1*x^2 + 1*x + 1 \\ &x^4 + x^2 + x + 1 \end{aligned}$$

adunarea coeficienților făcută modulo 2

înmulțirea făcută ca la polinoame, dar modulo P

$$\text{Ex. } (x^3+1)*(x^4+x) = x^7+x^4+x^4+x = x^7+x$$



# Algoritmul AES (3)

```

#define LENGTH 16                                /* # bytes in data block or key */
#define NROWS 4                                   /* number of rows in state */
#define NCOLS 4                                   /* number of columns in state */
#define ROUNDS 10                                 /* number of iterations */
typedef unsigned char byte;                     /* unsigned 8-bit integer */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
    int r;                                       /* loop index */
    byte state[NROWS][NCOLS];                   /* current state */
    struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1]; /* round keys */

    expand_key(key, rk);                        /* construct the round keys */
    copy_plaintext_to_state(state, plaintext);  /* init current state */
    xor_roundkey_into_state(state, rk[0]);       /* XOR key into state */

    for (r = 1; r <= ROUNDS; r++) {
        substitute(state);                      /* apply S-box to each byte */
        rotate_rows(state);                    /* rotate row i by i bytes */
        if (r < ROUNDS) mix_columns(state);   /* mix function */
        xor_roundkey_into_state(state, rk[r]);  /* XOR key into state */
    }
    copy_state_to_ciphertext(ciphertext, state); /* return result */
}

```



## Comentarii

- Nu au fost probleme la utilizare
- Experimental – difuzie bună
- Metodă **bazată pe algebră** (câmpuri Galois)
  - **substituții** și **mixare** coloane folosesc operații cu sens în teoria algebrică (nu simple tabele greu de explicat)
- autorii nu au oferit argumente matematice
- nu sunt suspectate **trape** (sau “scurtături” ascunse)



# Cifrarea prin functii greu inversabile

- **functii greu inversabile**
  - cunoscând  $x$  este usor de calculat  $f(x)$
  - calculul lui  $x$  din  $f(x)$  este foarte dificil.
- **adaptare:**
  - calculul lui  $x$  din  $f(x)$  trebuie să fie o **problemă intratabilă** doar pentru criptanalist
    - problemă intratabilă - nu există un algoritm de rezolvare în timp polinomial.
  - destinatarul autorizat
    - **are cheia sau**
    - **dispune de o trapă ce face problema usor de rezolvat.**
- **Metode**
  - algoritmi exponentiali
  - problema rucsacului.



# Algoritmi exponentiali – RSA

În RSA (Rivest, Shamir si Adleman):

Criptarea și decriptarea se fac prin funcții exponentiale

**Criptarea** se face prin calculul

$$C = (M^e) \text{ mod } n$$

unde **(e, n)** reprezintă **cheia de criptare**.

M este un bloc de mesaj (valoare întreagă între 0 și n-1)

C este criptograma.

**Decriptarea** se face prin calculul

$$M = (C^d) \text{ mod } n$$

unde **(d, n)** este **cheia de decriptare**



## Algoritmi exponentiali – RSA

**Conditia:** functiile de criptare si decriptare trebuie sa fie inverse una alteia:

$$(M^e \bmod n)^d \bmod n = M$$

Conditia poate fi indeplinita daca

- $e$  este un intreg relativ prim cu  $\Phi(n)$   
 $\Phi(n)$  este Functia lui Euler  
adica nr de întregi pozitivi  $< n$  relativ primi cu  $n$
- $d$  este inversul multiplicativ al lui  $e$  modulo  $\Phi(n)$   
 $e * d \bmod \Phi(n) = 1$
- $n$  este produsul a doua numere prime,  $n = p * q$   
caz in care  $\Phi(n) = (p-1)(q-1)$



## Motivatie

Functia lui Euler  $\Phi(n)$  = nr de întregi pozitivi  $< n$  relativ primi cu  $n$

daca  $p$  prim  $\Rightarrow \Phi(p) = p-1$ .

daca  $n = p^k q$  cu  $p, q$  prime atunci

$$\Phi(n) = \Phi(p^k) * \Phi(q) = (p-1)(q-1)$$

**Teorema (Euler).** Pentru orice  $a$  si  $n$  cu  $(a,n) = 1$  avem

$$a^{\Phi(n)} \bmod n = 1$$

Aceasta proprietate este folosita in demonstrarea urmatoarei

**Teorema (cifrare).** Date fiind  $e$  si  $d$  care satisfac

$$ed \bmod \Phi(n) = 1$$

si un mesaj  $M \in [0, n-1]$ , avem

$$(M^e \bmod n)^d \bmod n = M$$



## Metoda RSA

1. Se aleg două numere prime  $p$  și  $q$ ,  
(de ex. de 1024 biți).
2. Se calculează  $n = p \times q$   
și  $z = (p - 1) \times (q - 1)$ .
3. Se alege  $d$  un număr relativ prim cu  $z$   
 $d$  poate fi un numar prim care satisface  
$$d > (p-1) \text{ si } d > (q-1)$$
4. Se găsește  $e$  astfel încât  $e \times d = 1 \text{ mod } z$ .
5.  $(e, n)$  este **cheia de criptare**.  
 $(d, n)$  este **cheia de decriptare**.



## Comentarii

Cheia de criptare  $(e, n)$  se face publică

Problema:

- cunoașterea lui  $(e, n)$  să nu permită deducerea lui  $d$

Securitate pastrată deoarece:

- $p$  și  $q$  sunt numere prime foarte mari
- $p$  și  $q$  sunt pastrate secrete

Cifrarea și desifrarea sunt comutative și mutual inverse

$$(M^d \bmod n)^e \bmod n = M$$

→ RSA utilizată ptr confidentialitate și autentificare.

Nu au fost identificate atacuri reusite cu RSA



# Metoda MH (Merkle si Hellman) - optional

Problema rucsacului

Se dau C și ponderile A = (a[1], a[2], ..., a[m])

Se cere determinarea lui X = (x[1], x[2], ..., x[m]) cu elemente binare, a.i.

$$C = \sum_{i=1,m} x[i] * a[i]$$

Găsirea unei solutii = backtracking

=> număr operatii care creste exponential cu m.

O solutie x poate fi verificată prin cel mult m operații de adunare



Varianta rucsac simplu a problemei (trapa):

dacă A satisface proprietatea de dominantă (este o secvență super-crescătoare), adică

$$a[i] > \sum_{j=1, i-1} a[j]$$

atunci problema poate fi rezolvată în timp liniar.

Ex.

<b>text clar</b>	1	0	1	0	0	1
<b>rucsac</b>	1	2	5	9	20	<b>43</b>
<b>text cifrat</b>	1		5			<b>43</b>

suma = 49 reprezinta criptograma  
decriptarea ?



Cheie publică: secventa (oarecare) de intregi

Cheie secreta: secventa super-crescătoare

Contribuția Merkle și Hellman

conversie secventa oarecare  $\Leftrightarrow$  secventa super-crescătoare

Solutia: aritmetică modulară

rucsac simplu  $A = [a_1, a_2, \dots, a_m]$

rucsac greu  $G = [g_1, g_2, \dots, g_m]$

se obține prin calcule  $g_i = w * a_i \text{ mod } n$

Ex.

rucsac simplu  $A = [1, 2, 4, 9]$ ,  $w = 15$ ,  $n = 17$

$$1 * 15 \text{ mod } 17 = 15 \text{ mod } 17 = 15$$

$$2 * 15 \text{ mod } 17 = 30 \text{ mod } 17 = 13$$

$$4 * 15 \text{ mod } 17 = 60 \text{ mod } 17 = 9$$

$$9 * 15 \text{ mod } 17 = 135 \text{ mod } 17 = 16$$

rucsac greu  $G = [15, 13, 9, 16]$

Obs. înmulțirea mod  $n$  strică proprietatea de dominanță



Conditii:

Toate numerele din G trebuie sa fie distincte intre ele

Conversia inversa de la G la A trebuie sa produca o solutie unica

Impun restrictii asupra lui  $n$  si  $w$ ; ex.:

$$w = 3; n = 6$$

$x$	$3*x$	$3*x \text{ mod } 6$
1	3	3
2	6	0
3	9	3
4	12	0
5	15	3
6	18	0

$$w = 3; n = 5$$

$x$	$3*x$	$3*x \text{ mod } 5$
1	3	3
2	6	1
3	9	4
4	12	2
5	15	0
6	18	3

Cerinte:

1.  $n$  trebuie sa fie mai mare decat suma tuturor ai
2.  $w$  si  $n$  trebuie sa fie prime intre ele (se alege  $n$  prim)

=>  $w$  are un invers multiplicativ  $w^{-1}$  ( $w * w^{-1} = 1 \text{ mod } n$ )



## Criptare

Obtine criptograma C din textul clar P prin  $C = G * P$

unde G este rucsacul greu,  $G = w * A \text{ mod } n$  (adica  $g_i = w * a_i \text{ mod } n$ )

Ex:  $P = [1, 0, 1, 0]$ ,  $G = [15, 13, 9, 16] \rightarrow C = 15 + 9 = 24$

## Decriptare

Receptorul cunoaste A,w, n si, bineintelese, G

Deoarece  $C = G * P = w * A * P \text{ mod } n$ , rezulta

$$w^{-1} * C = w^{-1} * G * P = w^{-1} * w * A * P \text{ mod } n = A * P \text{ mod } n$$

din care P se afla prin rucsac simplu

Ex.  $A = [1, 2, 4, 9]$ ,  $w = 15$ ,  $n = 17$ ,  $C = 24$

$$w^{-1} = 15^{-1} = 8 \text{ mod } 17 \rightarrow w^{-1} * C = 8 * 24 = 192 \text{ mod } 17 = 5$$

$$A = [1, 2, 4, 9] \Rightarrow P = [1, 0, 1, 0]$$



## Comentarii ← optional

Metoda pare sigura

S-au gasit metode de atac prin ocolire rucsac greu în  
anumite cazuri

Oricum, algoritmul MH este greu de utilizat



# Analiza algoritmilor criptografici

## Problema:

Pot criptogramele interceptate de un intrus să faciliteze spargerea confidentialității criptografice ?

## O soluție este Teoria Informatiei

- care măsoară cantitatea medie de **informatie** transmisa de o sursă
- și, echivalent, cantitatea de **incertitudine** înlăturată (în medie) de un mesaj

Bazata pe noțiunile de **entropie** și **entropie condiționată**



# Entropia

**Entropia** este cantitatea medie de informatie transmisa de o sursă

Fie S o sursă de informații

- care transmite mesajele  $X_1, \dots, X_n$
- cu probabilitățile  $p(X_1), \dots, p(X_n)$ , ptr. care  $\sum_{i=1,n} p(X_i) = 1$ .

Entropia  $H(X)$  este:

$$H(X) = \sum_{i=1,n} p(X_i) * \log (1/p(X_i)) = -\sum_{i=1,n} p(X_i) * \log (p(X_i))$$

- $\log(1/p(X_i))$  = cantitatea de informatie primită la receptia lui  $X_i$ .
- baza logaritm = 2 → cantitatea măsurată în **număr de biti**

## Exemplu

- aruncarea monedei – cap sau pajura
- probabilități egale,  $1/2$
- informatie 1 bit:  $H(X) = \sum_{i=1,2} (1/2)^* \log (1/(1/2)) = 1$



# Incertitudinea

Entropia = cantitatea de **incertitudine** înălțurată (în medie) de un mesaj

Exemplu:

o sursă poate trimite  $n = 4$  mesaje, cu probabilități egale  $p(X) = 1/4$

$$H(X) = \sum_{i=1,4} (1/4) * \log (4) = 2$$

fiecare mesaj înălțură o **incertitudine** de 2 biti

(înainte nu se stia care din cele 4 mesaje va fi primit)

Entropia depinde de **distributia probabilitatilor** mesajelor

–  $H(X)$  maxim      când  $p(X_1) = p(X_2) = \dots = p(X_n) = 1/n$ .

$$H(X) = \sum_{i=1,n} (1/n) * \log (n) = \log (n)$$

–  $H(X)$  descrește      când distribuția mesajelor se restrâng.

–  $H(X) = 0$       când  $p(X_i) = 1$  pentru un mesaj i.



# Entropie conditionata - Echivocitatea

Exemplu:

- mesajele X sunt conditionate de mesaje Y

Fie  $m=4$  și  $p(Y) = 1/4$  pentru fiecare Y.

Presupunem ca fiecare mesaj Y restrânge X astfel:

dupa Y1: urmeaza X1 sau X2, fiecare cu prob. 1/2

dupa Y2: urmeaza X3 sau X4,

dupa Y3: urmeaza X2 sau X3,

dupa Y4: urmeaza X4 sau X1.

Problema:

- cum se calculeaza entropia lui X ?



## Entropie conditionata – Echivocitatea (2)

Dat fiind  $Y$  din mulțimea mesajelor  $Y_1, \dots, Y_n$  cu  $\sum_{i=1,n} p(Y_i) = 1$ ,

- fie:  $p_Y(X)$  probabilitatea mesajului  $X$  condiționat de  $Y$ .

$p(X,Y)$  probabilitatea mesajelor  $X$  și  $Y$  luate împreună:

$$p(X,Y) = p_Y(X)*p(Y).$$

- **Echivocitatea** este entropia lui  $X$  condiționat de  $Y$ :

$$H_Y(X) = \sum_{X,Y} p(X,Y)*\log (1/p_Y(X))$$

$$H_Y(X) = \sum_{X,Y} p_Y(X)*p(Y) \log (1/p_Y(X))$$

$$= \sum_Y p(Y) \sum_X p_Y(X)*\log (1/p_Y(X)).$$



Pentru exemplu:

$$p(Y) = 1/4 \quad p_Y(X) = 1/2$$

Echivocitatea este:

$$H_Y(X) = \sum_Y p(Y) \sum_X p_Y(X) * \log (1/p_Y(X)).$$

$$H_Y(X) = \sum_{i=1,4} (1/4) * \sum_{j=1,2} (1/2) * \log (1/(1/2))$$

$$H_Y(X) = 4 * (1/4) * 2 (1/2) * (\log 2) = \log 2 = 1.$$

**H(X) = 2** pentru mesaje X neconditionate

**H<sub>Y</sub>(X) = 1** pentru mesaje X conditionate

→ cunoașterea lui Y reduce incertitudinea lui X la un bit.



# Confidențialitatea perfectă

Cunoasterea unor criptograme nu reduce confidentialitatea

Fie:

- $M$  - multime texte clare cu probabilitatea  $p(M)$ ,  $\sum_M p(M) = 1$ .
- $C$  - criptograme, cu probabilitatea  $p(C)$ ,  $\sum_C p(C) = 1$ .
- $K$  - chei cu probabilitatea  $p(K)$ ,  $\sum_K p(K) = 1$ .
- $p_C(M)$  probabilitatea să se fi **transmis  $M$**  când se recepționează  $C$ .

Confidențialitatea perfectă  $\Leftrightarrow p_C(M) = p(M)$ .

Fie  $p_M(C)$  probabilitatea să se **recepționeze  $C$**  când s-a transmis  $M$ :

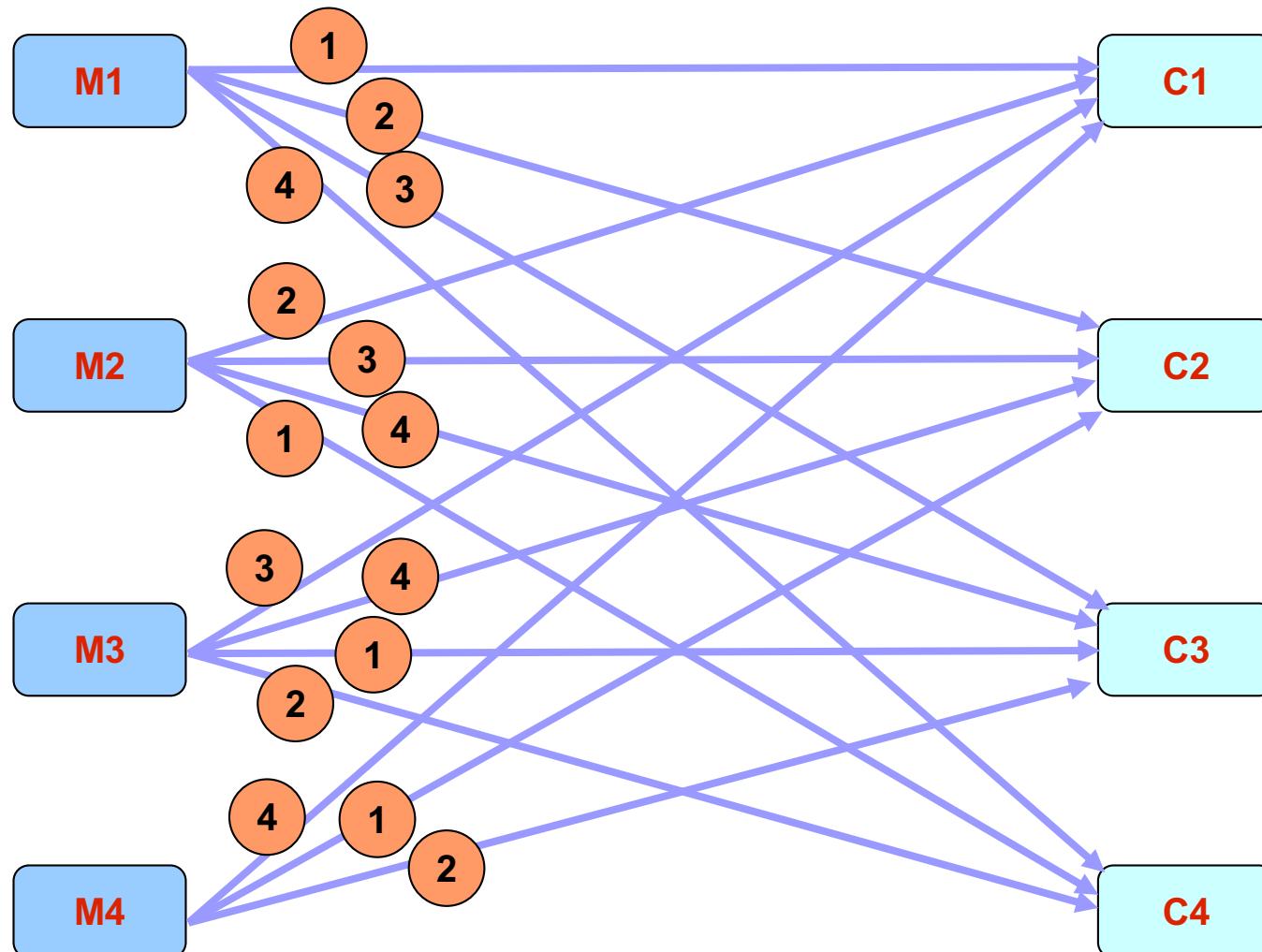
$$p_M(C) = \sum_{k, E_k(M)=C} p(k).$$

• Confidențialitatea perfectă:

$$p_M(C) = p(C), \text{ pentru toate } M \text{ și orice } C.$$

# Confidențialitatea perfectă

- Confidențialitatea perfectă este posibilă dacă se folosesc chei la fel de lungi ca mesajele codificate.





## Distanța de unicitate

- “Spargerea” confidențialității depinde de cantitatea de criptograme de care intrusul dispune
  - cantitatea de incertitudine în  $K$  cunoscând  $C$ , este exprimată ca entropia (echivocitatea) cheii conditionata de criptograme:
$$H_C(K) = \sum_C p(C) \sum_K p_C(K) \log (1/p_C(K))$$
- Dacă echivocitatea  $H_C(K)=0$  nu există incertitudine și cifrul se poate sparge.
- Când crește lungimea  $N$  a textelor cifrate echivocitatea scade.
- Distanța de unicitate:
  - **Cel mai mic  $N$  pentru care  $H_C(K)$  este foarte apropiat de 0.**
- Cifru neconditionat sigur:
  - $H_C(K)$  nu se apropie niciodată de 0.



# Redundanta limbajului (1)

Pentru un **limbaj**, consideram mulțimea mesajelor **X** de lungime **N**

- cu entropia **H(X)**
- fiecare mesaj este o secvență de **N** simboluri dintr-un alfabet A care are **L** simboluri

Nu toate combinatiile de **N** simboluri au sens

- ex. anumite succesiuni de consoane, digrame, trigrame, etc.
- **Redundanța limbajului**, **D** este

$$D = R - r$$

- **R** este **rata absolută** a limbajului
- **r** este **rata** limbajului
- **D = 3.2 ... 3.7** pentru limba engleză.



## Rata limbajului

Rata limbajului este entropia pe simbol daca nu toate combinatiile de N simboluri au sens

este raportul entropia mesaj / numar simboluri din mesaj:

$$r = H(X) / N$$

- $r$  = numar de biti pentru un simbol  
cu care se pot reprezenta (un numar de)  $2^r$  simboluri
- pentru limba engleză  $r = 1 \dots 1.5$  biti pe litera
- Numarul total al mesajelor de lungime  $N$  cu sens este  $2^{rN}$



## Rata absolută a limbajului

Rata absolută a limbajului este entropia pe simbol daca toate combinatiile de N simboluri ar avea sens

Se considera ca cele L simboluri au aceeasi probabilitate =  $1/L$

Rezulta:

$$R = \sum_{i=1,L} (1/L) \log (L) = \log L$$

- pentru limba engleză  $R = \log 26 = 4.7$  biți pe literă
- Numarul de simboluri L se poate scrie  $L = 2^R$
- Numarul de mesaje de lungime N (cu sau fara sens) este  $2^{RN}$



# Calcul aproximativ distanță unicitate (1)

Distanța de unicitate N este:

$$N = H(K) / D$$

## Justificare

- Ipoteze:

- Sunt  $2^{RN}$  mesaje posibile, din care  $2^{rN}$  au sens.
- Toate mesajele cu sens au aceeași probabilitate,  $1/2^{rN}$ .
- Toate mesajele fără sens au probabilitate 0.
- Sunt  $2^{H(K)}$  chei cu probabilități egale.
- Cifrul este aleator:
  - Pentru fiecare k și C, descifrarea  $D_K(C)$  este variabilă aleatoare independentă uniform distribuită pe toate mesajele, cu sau fără sens.



## Calcul aproximativ distanță unicitate (2)

Fie criptograma  $C = E_K(M)$ .

- Criptanalistul are de ales între  $2^{H(K)}$  chei, doar una este corectă.
  - Rămân  $2^{H(K)} - 1$  chei care pot da o soluție falsă (adică  $C$  se obține criptând un alt mesaj  $M'$  cu înțeles)
- cu aceeași probabilitate (= mesaje cu sens / total mesaje)

$$q = 2^{rN} / 2^{RN} = 2^{(r-R)N} = 2^{-DN}$$

- Numărul de soluții false  $F$  (= nr chei incorecte \* probabilitatea unei chei de a da soluție falsă):

$$F = (2^{H(K)} - 1)q = (2^{H(K)} - 1) 2^{-DN} \approx 2^{H(K)-DN}$$

conditia de unicitate  $\rightarrow \log F = H(K)-DN = 0$

$$\rightarrow N = H(K) / D$$



# Analiza cifrării prin substituție

- Substituție **monoalfabetică**:
  - $N = H(K) / D = \log n! / D$  sunt  $n!$  chei posibile
  - Pentru limba engleză:
    - $N = \log 26! / 3.2 = 27.6$
- Substituție **periodică** cu perioada **d** și **s** simboluri în alfabet:
  - Sunt  $s^d$  chei posibile pentru fiecare substituție simplă:
    - $N = H(K) / D = \log s^d / D = (d * \log s) / D$
  - Pentru cifrul **Vigenere**  $s = 26$ :
    - $N = d * 4.7 / 3.2 = 1.5 d$



# Analiza cifrării prin transpoziție

- Caracteristici:
  - Cifrul permutează caracterele cu o perioadă fixă  $d$ .
  - Sunt  $d!$  permutări posibile.
  - Toate sunt echiprobabile.
- $H(K) = \log d!$ 
  - $N = H(K) / D = \log d! / D$
  - $N = d \log (d/e) / D$
- Pentru  $d = 27$  și  $D = 3.2$  rezultă:
  - $N = 27.9$



## Studiu individual

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

8.1 CRIPTOGRAFIA

8.2 ALGORITMI CU CHEIE SECRETĂ

8.3 ALGORITMI CU CHEIE PUBLICĂ

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

8.1 CRYPTOGRAPHY

8.2 SYMMETRIC-KEY ALGORITHMS

8.3 PUBLIC-KEY ALGORITHMS