Protocoale de comunicaţie

Introducere

Catalin Leordeanu
catalin.leordeanu@upb.ro

Programa cursului (1)

- **Modele de referinţă**
  - Arhitectura retelelor. Servicii si Protocoale.
  - Modelele ISO OSI şi TCP / IP
- **Legătura de date**
  - Servicii şi funcţii. Detecţia şi corectarea erorilor. Controlul transmisiei
  - Protocoale start-stop şi cu fereastră glisantă
  - Analiza performanţelor protocoalelor legăturii de date
  - Ethernet, PPPoE
- **Protocoale de reţea. Dirijarea datelor**
  - Serviciile nivelului reţea. Organizare internă.
  - Adresare. Algoritmi de dirijare.
  - Protocoale de rutare si control in Internet: IP, ARP, ICMP, OSPF, BGP, IPv6.

Programa cursului (2)

- **Protocoale de transport**
  - Primitivele serviciului de transport.
  - Stabilirea şi eliberarea conexiunii de transport.
  - Controluì fluxului şi retransmiterea adaptivă.
  - Nivelul transport în Internet: TCP, UDP, Wireless TCP
  - Probleme de performanţă şi soluţii
- **Protocoale Internet de aplicatie**
  - DNS. Servere de nume. Rezolvarea numelor
  - HTTP, HTTP/2
  - FTP
  - SMTP. Adresare. Structura mesajelor. MIME, POP3, IMAP

Programa cursului (3)

- **Protocoale de securitate**
  - Noţiuni de criptografie.
  - Algoritmi cu chei simetrice, DES, AES.
  - Algoritmi cu chei publice, RSA
  - Semnături digitale
  - Gestiunea cheilor publice. Certificate, PKI
  - Securitatea comunicaţiei. IPSEC
  - Protocoale de autentificare
  - Securitatea poştei electronice, a Web-ului şi a DNS-ului

Bibliografie (recomandat)

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

2. Ross Kurose, “Computer Networking: A top-down approach”, 6th ed. Pearson, 2017

Regulament (detalii pe site!)

- Nota finala la cursul de **PC** se calculează după următoarea formula

Nota = NotaCurs + NotaLab + NotaTeme + NotaPractic+ NotaExam

unde:

- **NotaCurs** (maxim 1 punct) este punctajul obţinut la testele date la curs
- **NotaLab** este punctajul obţinut in activitatea din laborator (maxim 1 punct)
- **NotaTeme** - 3 teme de casa (maxim 3 puncte) date la laborator
- NotaPractic – nota de la testul practic, dat la laborator (maxim 1.5 puncte)
- **NotaExam** (maxim 4 puncte) este punctajul obţinut la examen

- Disciplina este promovata **numai daca** sunt îndeplinite următoarele condiţii:
  - **NotaCurs** + **NotaTeme** + **NotaLab** >= 3
  - **NotaPractic** >= 0.75
  - **NotaExam** >= 2

- Site: ACS Moodle + OCW (https://ocw.cs.pub.ro/courses/pc)

Organizare Teme

	Saptamana	Tematica	Deadline
Tema 1	sfarsit S4	Router implementation	sfarsitul sapt S6
Tema 2	sfarsit S7	Protocoale de transport (TCP+UDP)	sfarsitul sapt S9
Tema 3	S10	HTTP, Client REST	S12

Modele arhitecturale de referinţă

Cuprins

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

Foarte importante!

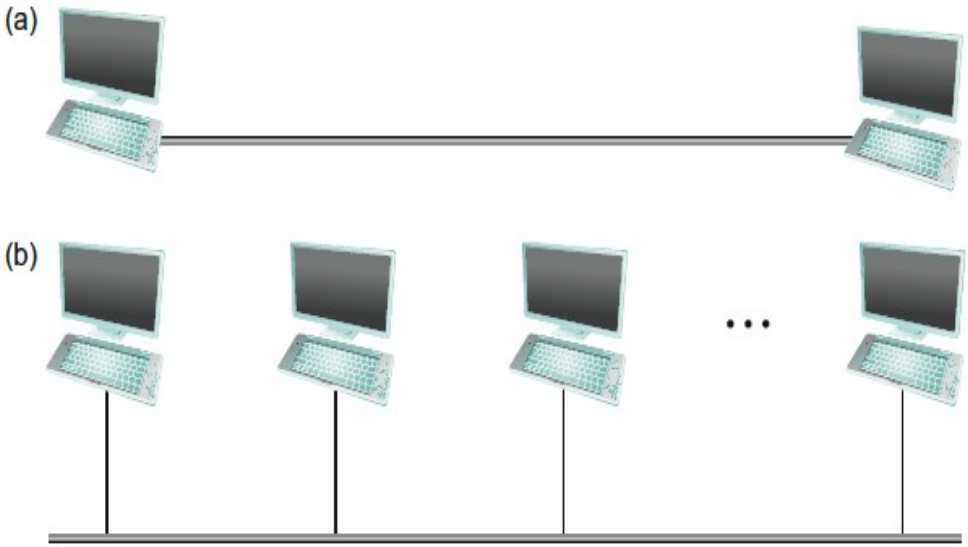
Notiunile fundamentale, utilizate in restul cursului

Imaginea de ansamblu a protocoalelor ce vor fi studiate

Comunicarea in aplicatii Internet

- Internetul este cunoscut de utilizatori prin aplicatiile sale
  - e-Mail
  - Transferul de fisiere
  - Web etc.
- Bazate pe comunicarea intre calculatoare (noduri)

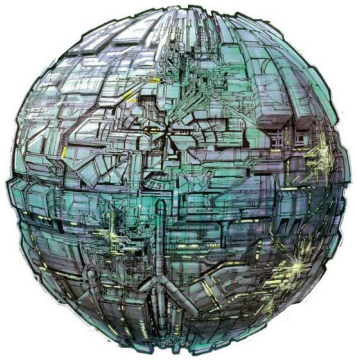
- Legatura poate fi directa
  a. punct la punct – prin fire
  b. acces multiplu – de ex wireless

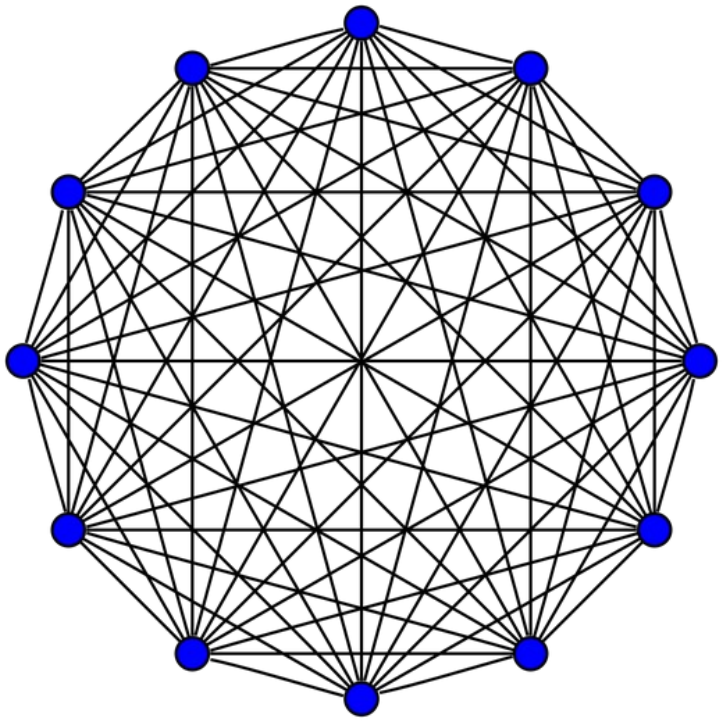


Legatura de date

- functionalitati specifice datelor (≠ comunicari vocale)
  - codificarea bitilor
  - detectia erorilor de transmisie si corectarea lor
  - controlul fluxului de date
  - controlul accesului la mediu (pentru acces multiplu)

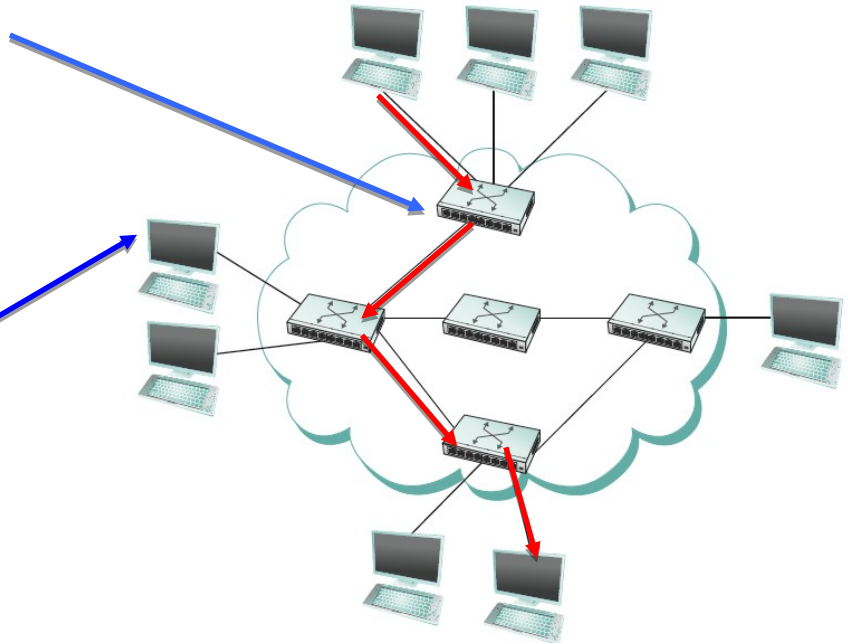
- Legaturile directe intre noduri nu ofera scalabilitate





Retele

- Comunicarea intre calculatoare prin noduri intermediare (switch-uri)
  - atasate la mai multe legaturi
  - formeaza o retea
- Functionare dupa principiul sistemului postal
- Noduri gazdă (ale aplicatiilor)
- *(en. hosts)*
- Fiecare nod are o adresa de retea
- Nodul sursa include in pachet adresa nodului destinatie

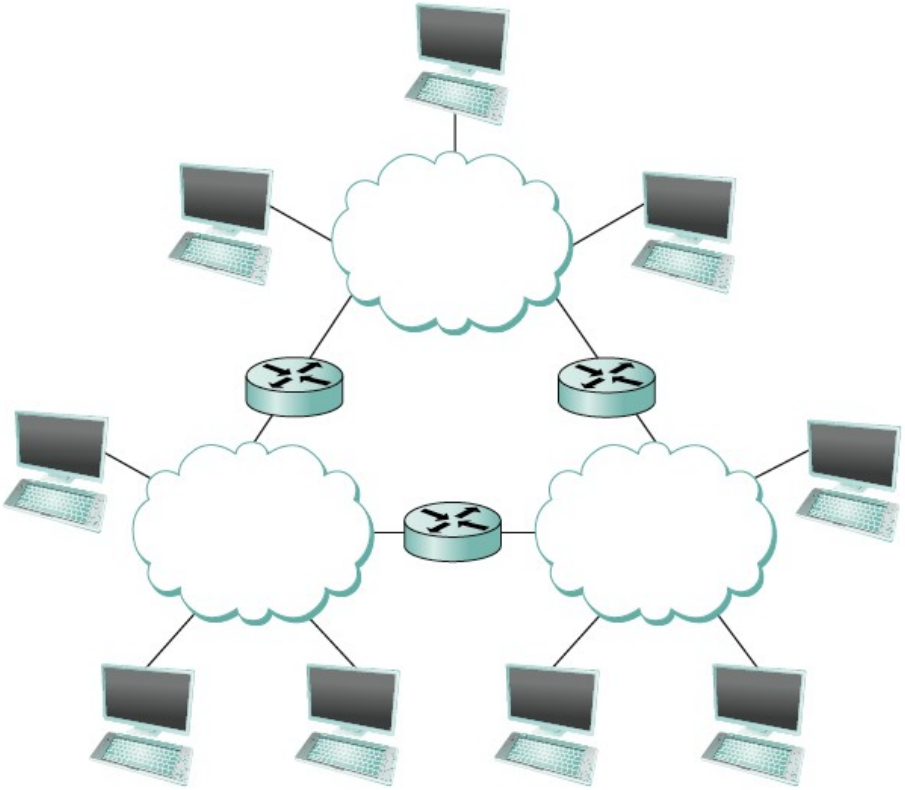


Retele interconectate

- Nodurile intermediare pot fi organizate in mai multe retele interconectate
- Router (sau gateway) - nod conectat la mai multe retele
  - are functii similare cu switch-ul

Nodurile intermediare

- Transmit pachete intre oricare noduri host (gazda)
- Nu asigura corectitudinea transmisiei
  - e.g. pachete pierdute



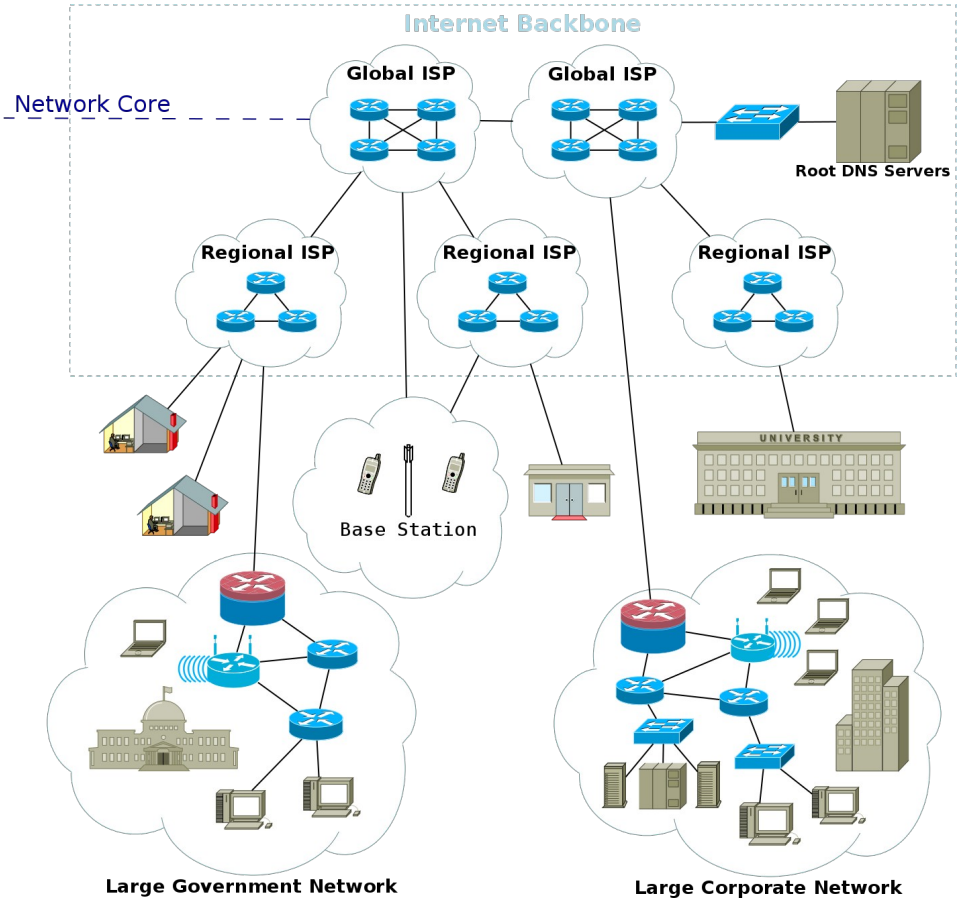
Internet

- "Network of networks"
- Organizare vag ierarhica

- Standarde internet:
- RFC: request for comment
- IETF: Internet Engineering Task Force https://www.ietf.org/

- Infrastructura de comunicatie
- permite functionarea aplicatiilor
- distribuite (Web, VoIP, email,
- jocuri, file sharing, etc.)

- Servicii de comunicatie:
  - Transfer sigur de date de la
  - sursa la destinatie
  - Transfer de date nesigur
  - ("best effort")



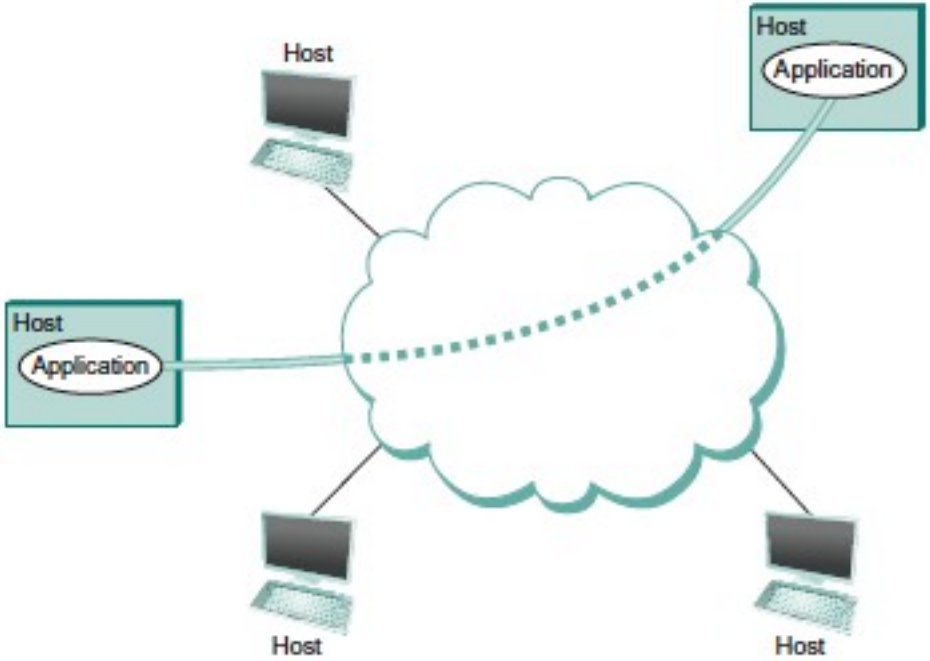
Comunicarea intre aplicatii

Reteaua oferă canale “logice” prin care procesele de aplicatie sa comunice intre ele

Facilitati suplimentare:

- identificarea unica a fiecarui capat de canal logic prin (adresa de retea, port)

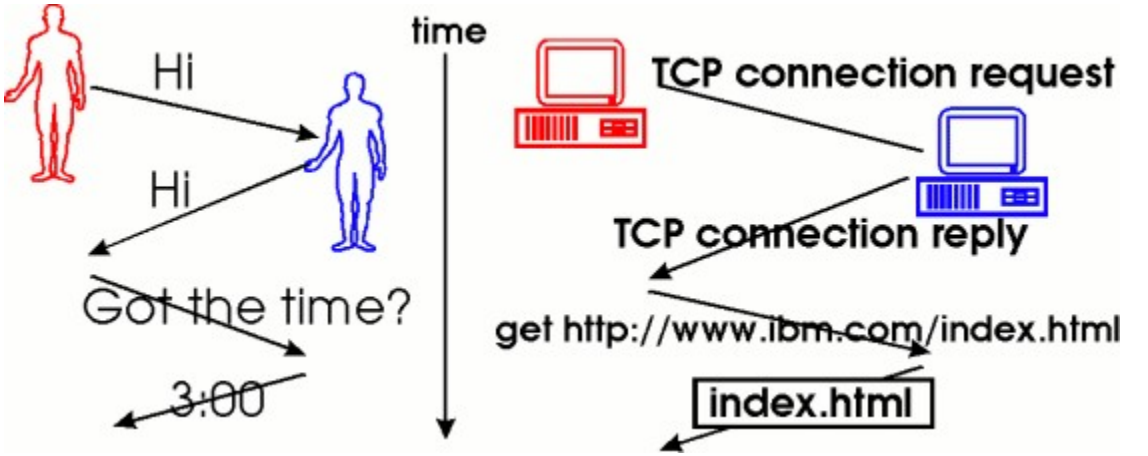
- receptia corectă a mesajelor
- livrarea mesajelor in ordine
- confidentialitatea
- etc.



Functionarea canalului logic

- Difera de la o aplicatie la alta
- o cerere – un raspuns in descarcarea unei pagini Web
  - clientul (browser) trimite o cerere GET catre un server Web
  - serverul trimite ca raspuns pagina
  - clientul o afisaza pe ecran
- o cerere – o serie de raspunsuri in aplicatii audio/video
  - clientul transmite cererea
  - serverul trimite ca raspuns o serie de mesaje
    - continutul livrat utilizatorului pe masura ce este primit
    - constrangeri de timp

Ce este un protocol de comunicatie?



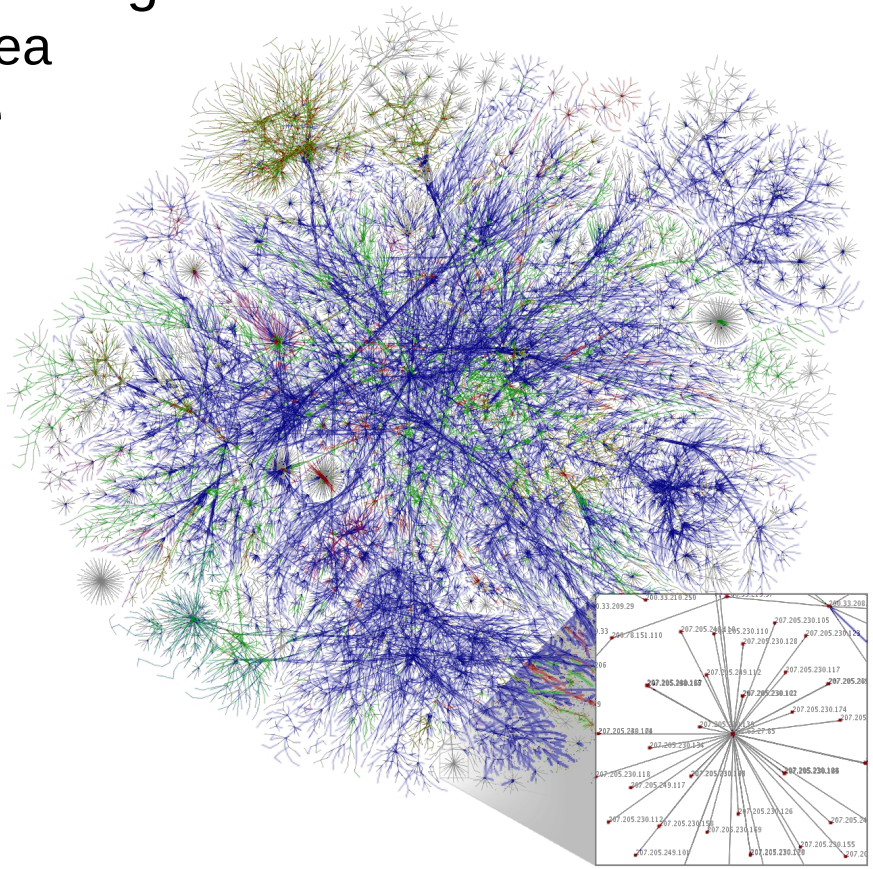
Protocol de comunicatie uman vs. intre calculatoare

Protocol – set de reguli respectate de parti pentru a comunica intre ele;
Se refera la

- Ordinea si formatul mesajelor transmise intre 2 sau mai multe entitati
- operatiile executate la trimiterea sau la primirea unui mesaj
  - trimite cerere conectare, primeste raspuns, confirma raspuns etc.

Arhitectura Retelelor de Calculatoare

- Arhitectura
  - care sunt componentele si cum interactioneaza
- Retea = ansamblu de componente logice
  - simplifica intelegerea si realizarea
- Componente pentru comunicare
  - pe o legatura de date
  - intre doua noduri gazda
  - pe canale logice



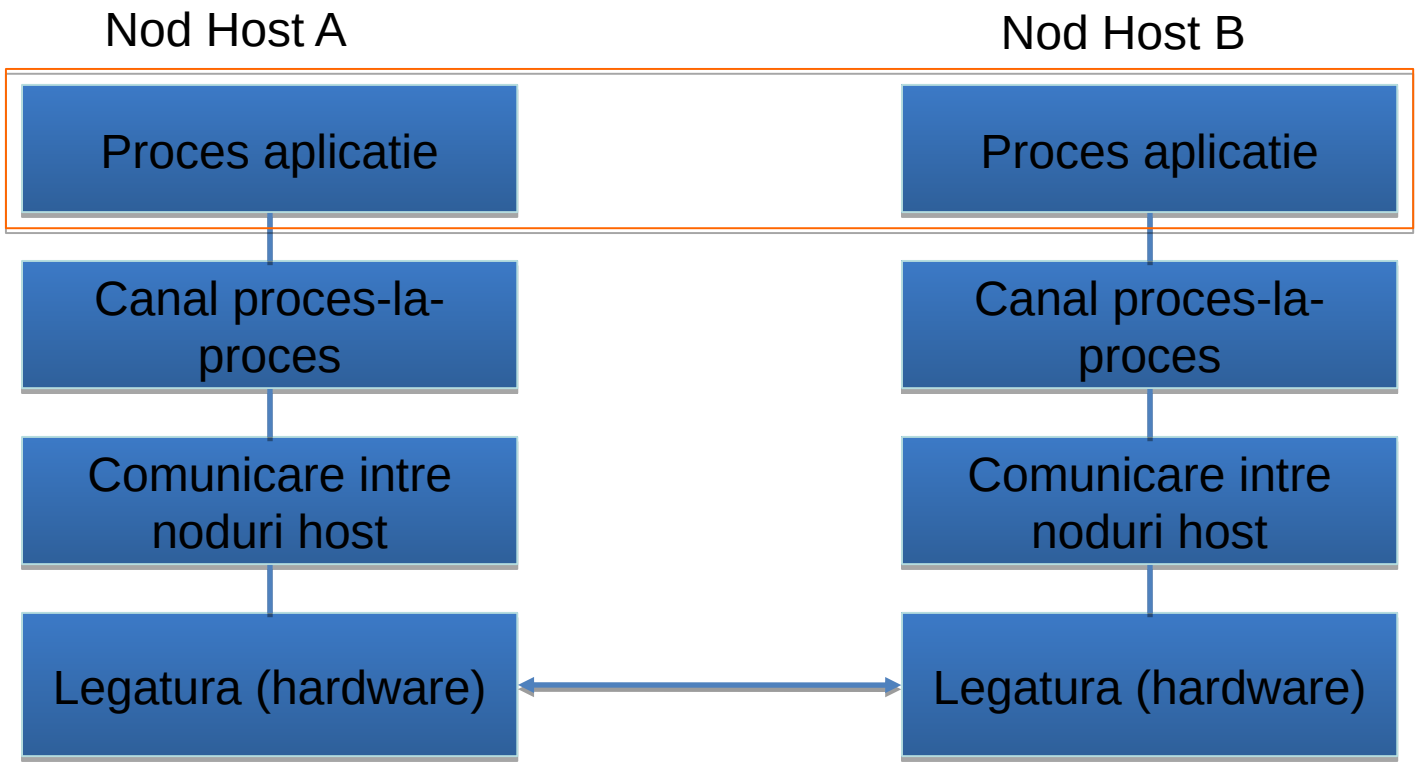
Sursa: Opte Project

Arhitectura ierarhica

Arhitectura este o ierarhie de nivele functionale

Fiecare componenta apartine unui nivel diferit

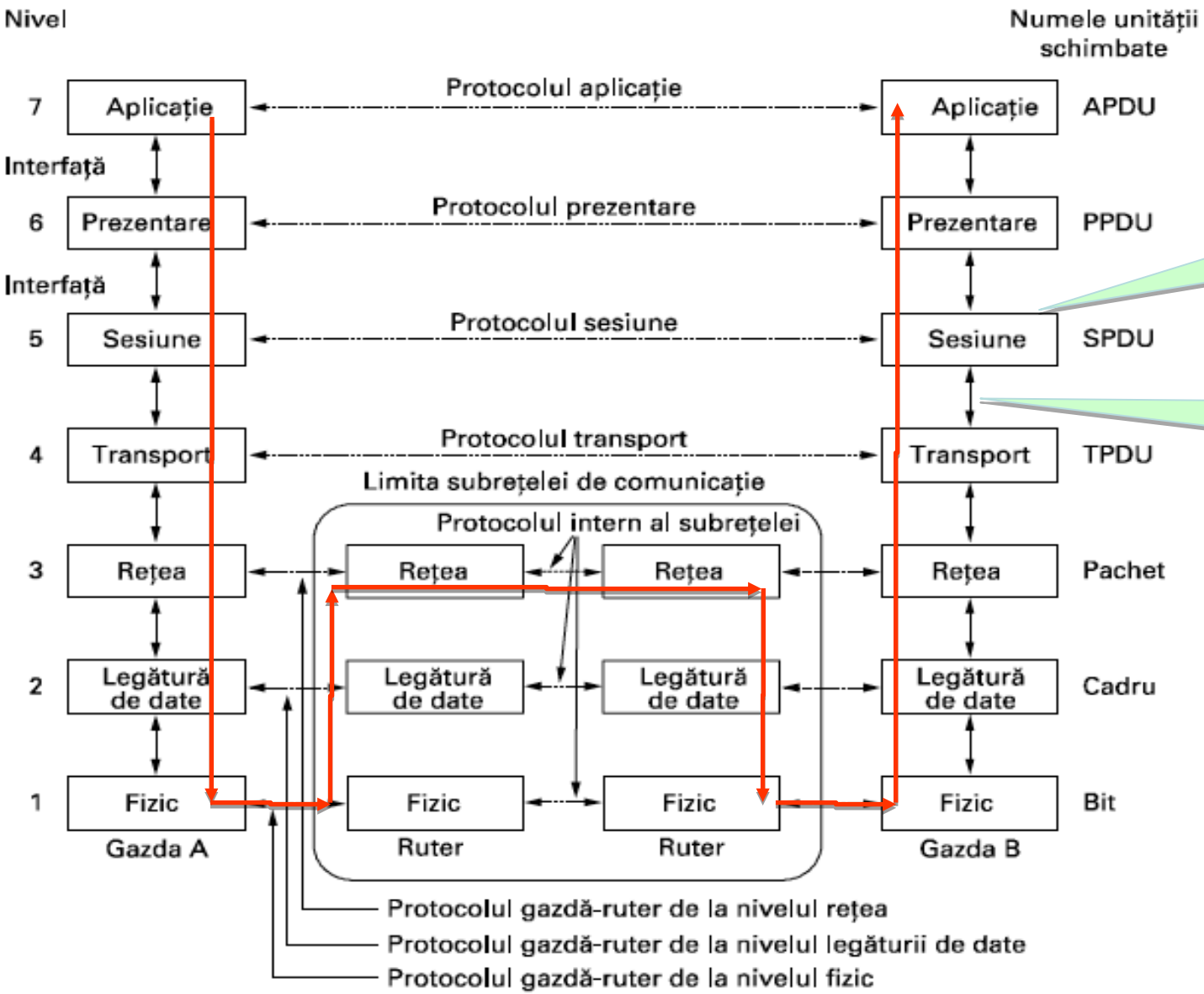
si foloseste serviciul componentei de nivel inferior



Modelul de Referinţă ISO OSI (Open Systems Interconnection)

UPPER LAYERS	7	**Application Layer** ✓ Message format, Human-Machine Interfaces
	6	**Presentation Layer** ✓ Coding into 1s and 0s; encryption, compression
	5	**Session Layer** ✓ Authentication, permissions, session restoration
TRANSPORT SERVICE	4	**Transport Layer** ✓ End-to-end error control
	3	**Network Layer** ✓ Network addressing; routing or switching
	2	**Data Link Layer** ✓ Error detection, flow control on physical link
	1	**Physical Layer** ✓ Bit stream: physical medium, method of representing bits

Modelul de Referinţă ISO OSI



entitate de protocol

interfaţă

Nivel fizic

- **Funcţie** - transmitere a şirurilor de biţi pe un canal de comunicaţie
- Principalele **probleme**
  - codificarea zerourilor şi a unităţilor
  - stabilirea şi desfiinţarea conexiunilor fizice
  - modul de transmisie (semiduplex sau duplex) etc.
- **Exemplu**
  - 802.11 Wi-Fi

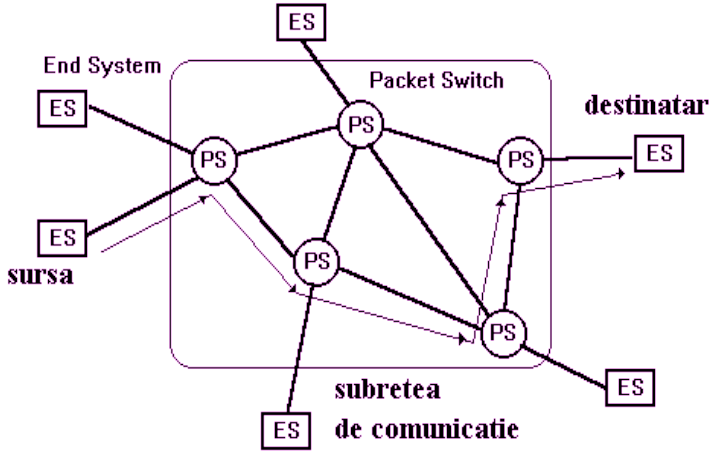
Legătura de date

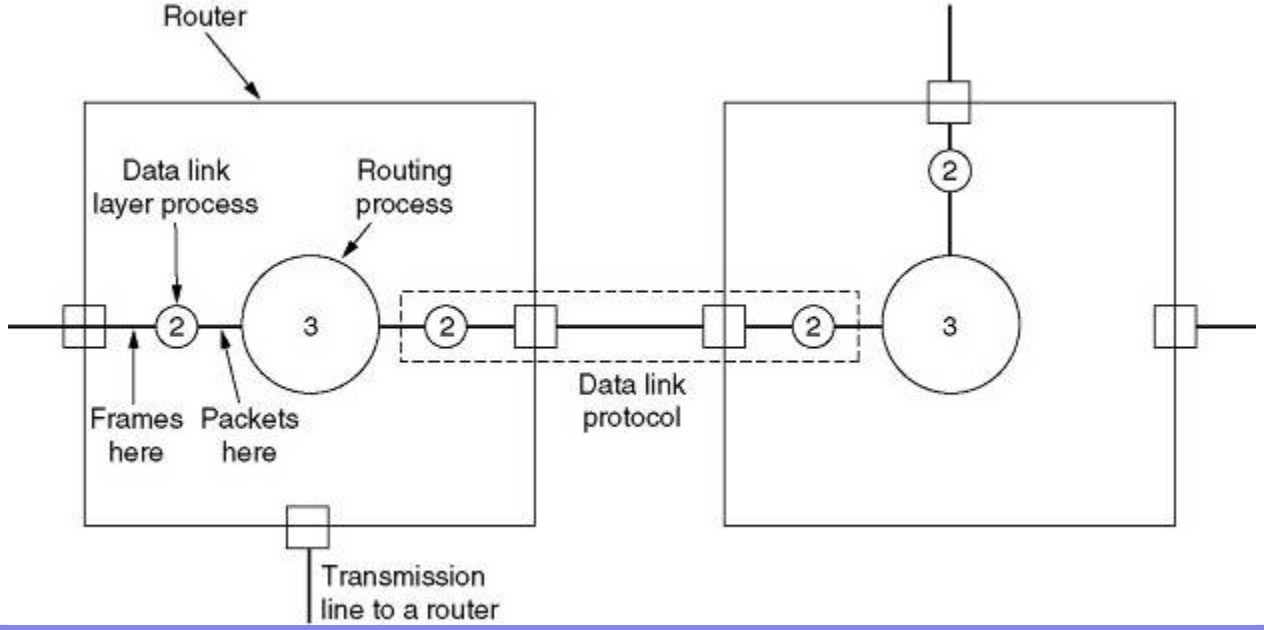
- **Funcţie** – realizează comunicarea sigură şi eficientă a datelor între două noduri adiacente (conectate printr-un canal fizic de comunicaţie)
- **Probleme rezolvate**
  - Încadrare
  - Control erori
  - Control flux
  - Transmisie transparentă
  - Management legătură
- **Exemplu** de cadru: HDLC (High Level Data Link Control)

  flag address command data FCS flag
- **Implementare** prin
  - adaptoare de retea
  - drivere din sistemul de operare al calculatorului

Nivel reţea

- **Funcţie** - transmiterea pachetelor între oricare două noduri din retea
- **Probleme rezolvate**
  - alegerea legăturii următoare (dirijarea)
  - adresarea
  - calculul tabelelor de dirijare





Nivel Transport

Funcţie - asigura un transfer de date corect, eficient între procese din sistemul sursă şi din sistemul destinatar

Oferă

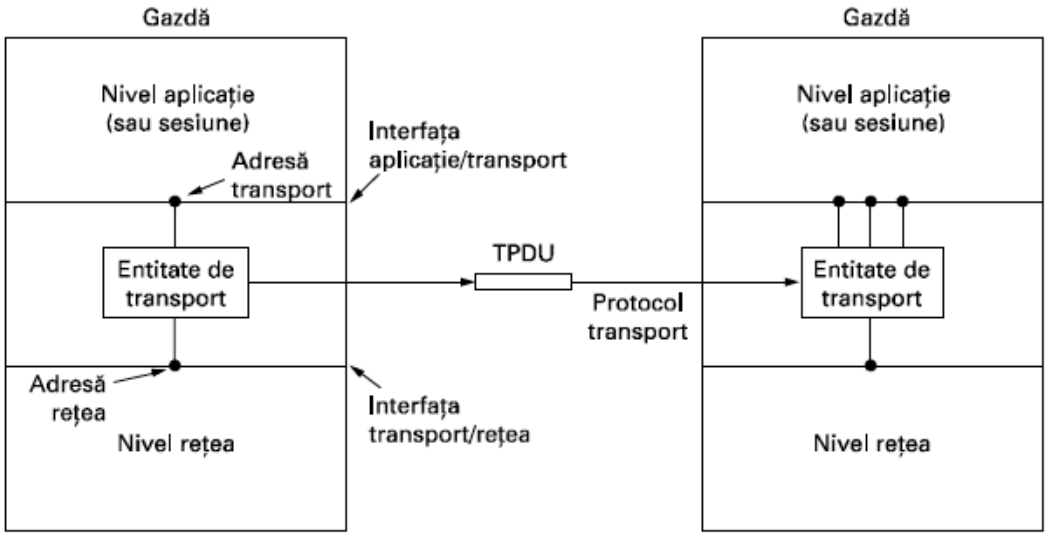
- un transfer sigur al datelor, chiar cu o reţea nesigură;
- o interfaţă uniformă pentru nivelul superior, independent de tipul reţelei utilizate.

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

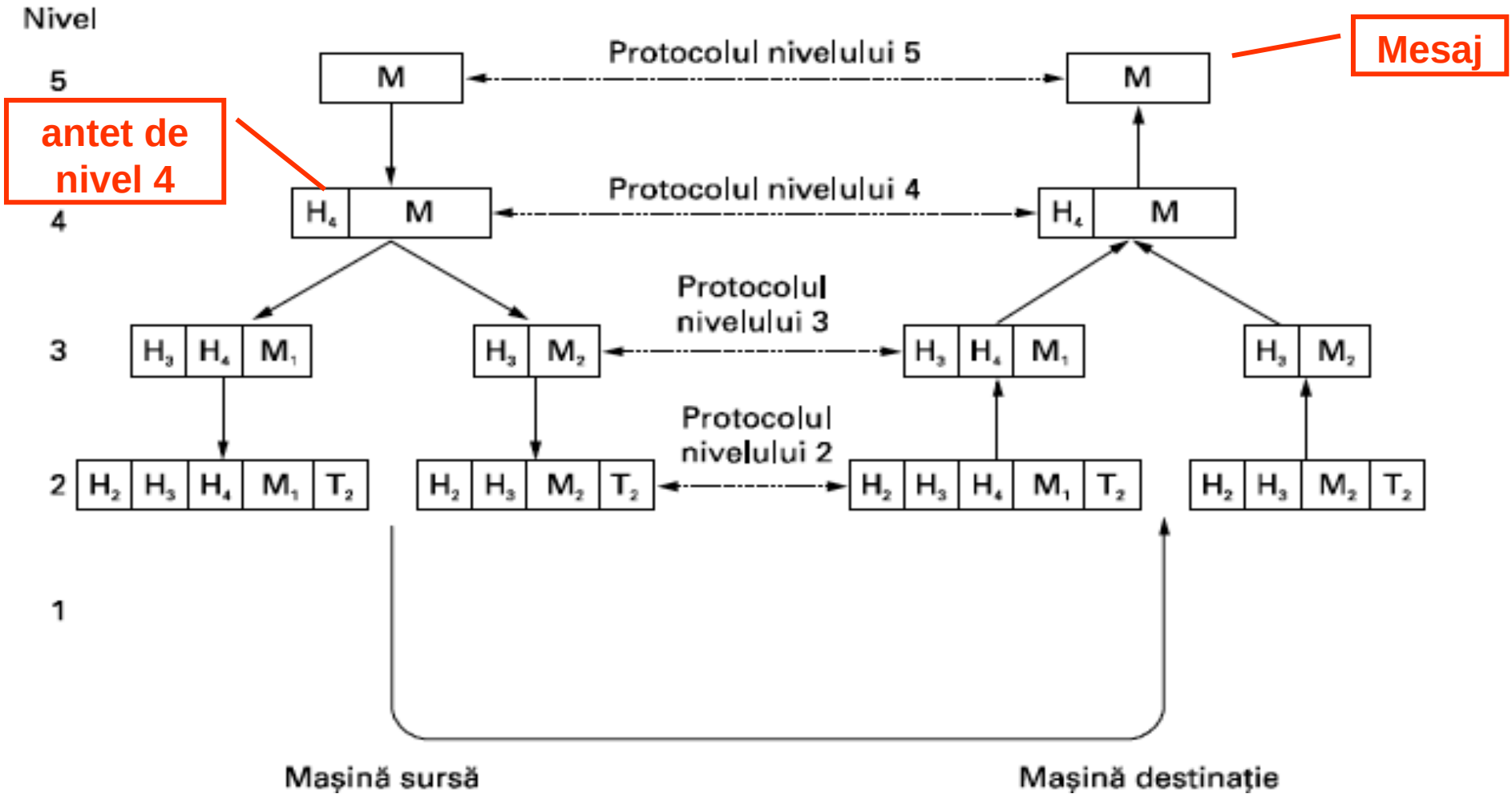
- **Nivel Sesiune**
  - Controlul dialogului intre aplicatii
  - Sincronizarea transferurilor
  - Stabilirea unor puncte de verificare si reluare a transferurilor

- **Nivel Prezentare**
  - Conversia formatului datelor intre
    - sintaxa folosita de aplicatii si
    - sintaxa de transfer

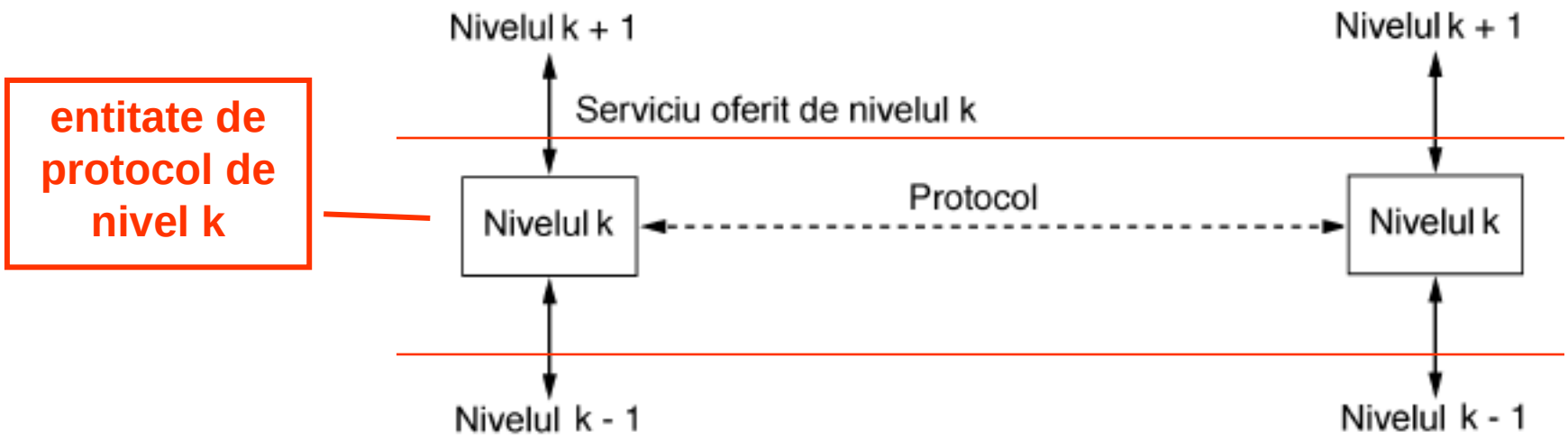
Nivel Aplicaţie

- Servicii comune unor categorii de aplicaţii
  - Schimb de mesaje
  - Transfer de Fişiere
  - Terminal Virtual
  - Serviciu de Directoare

Fiecare protocol are propriul antet



Relaţia între servicii şi protocoale



Entitatile de nivelul k

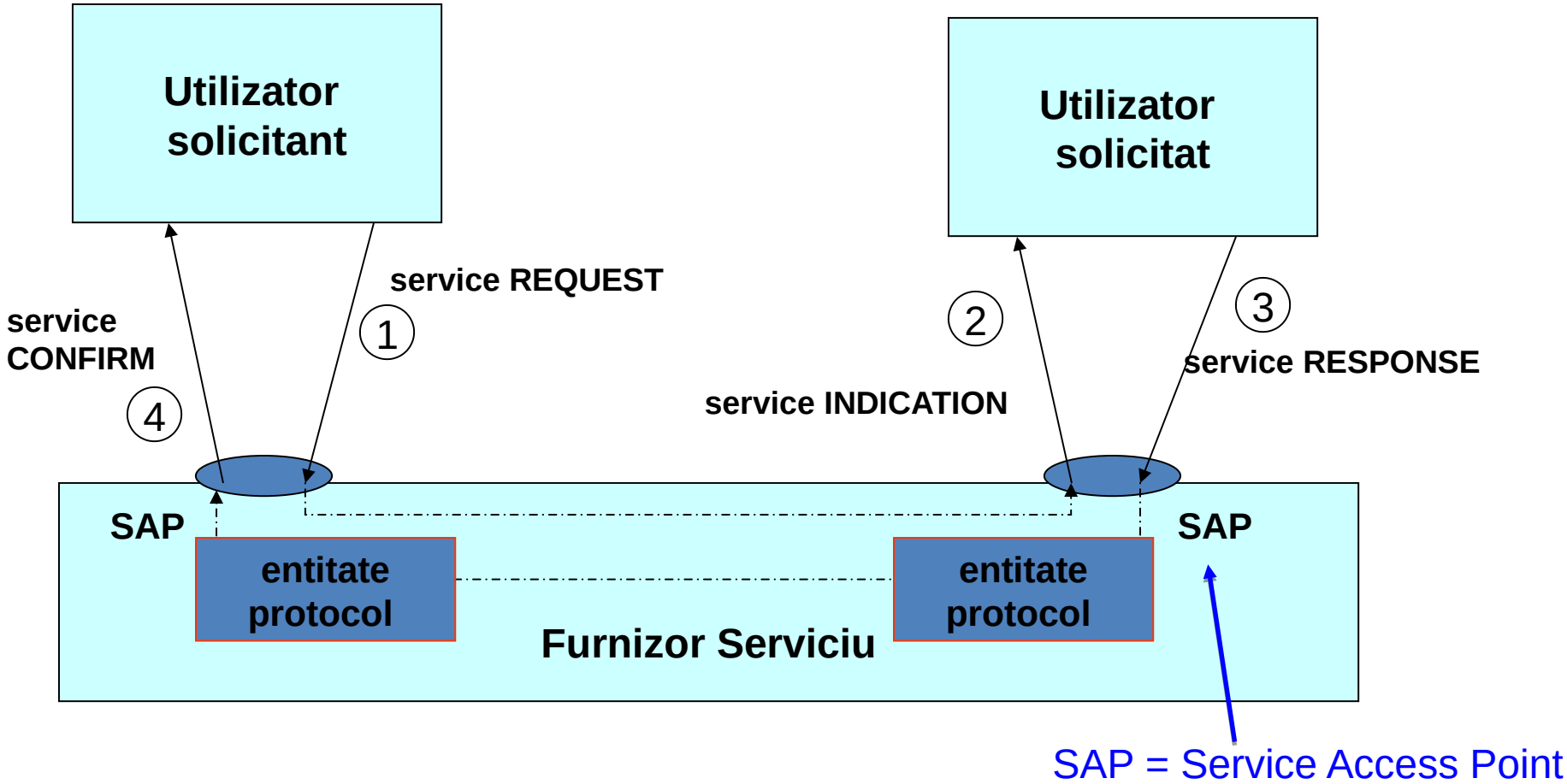
- folosesc protocolul de nivel k si serviciul de nivel k-1
- pentru a furniza serviciul de nivel k

Primitive de serviciu

- Un serviciu este specificat de un set de **primitive** (operaţii accesibile, printr-o interfata, utilizatorului serviciului)

- Patru clase de primitive
  - REQUEST cere un serviciu
  - INDICATION anunţa primirea unei cereri
  - RESPONSE răspunde cererii
  - CONFIRM confirmă cererea

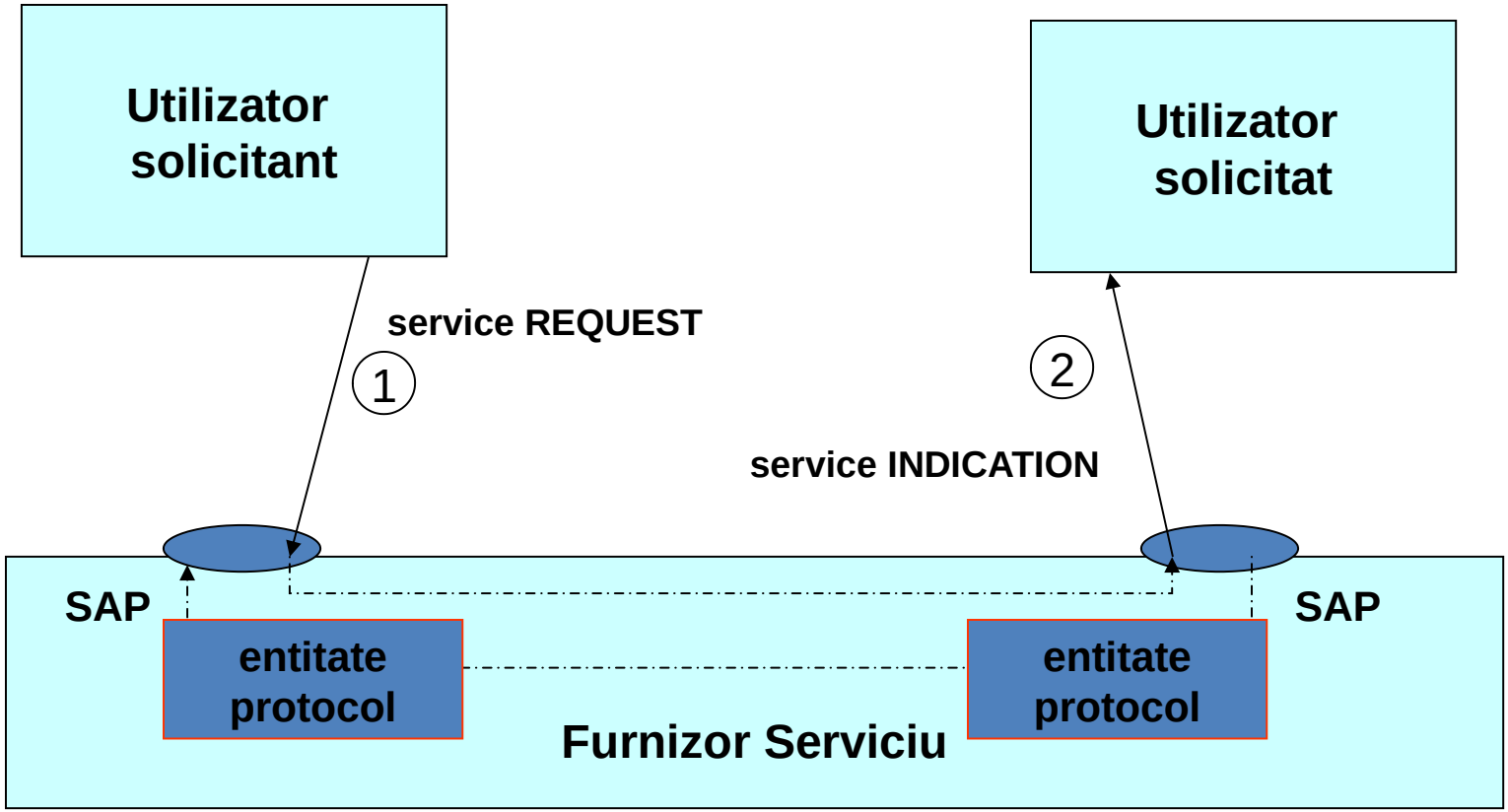
Servicii confirmate



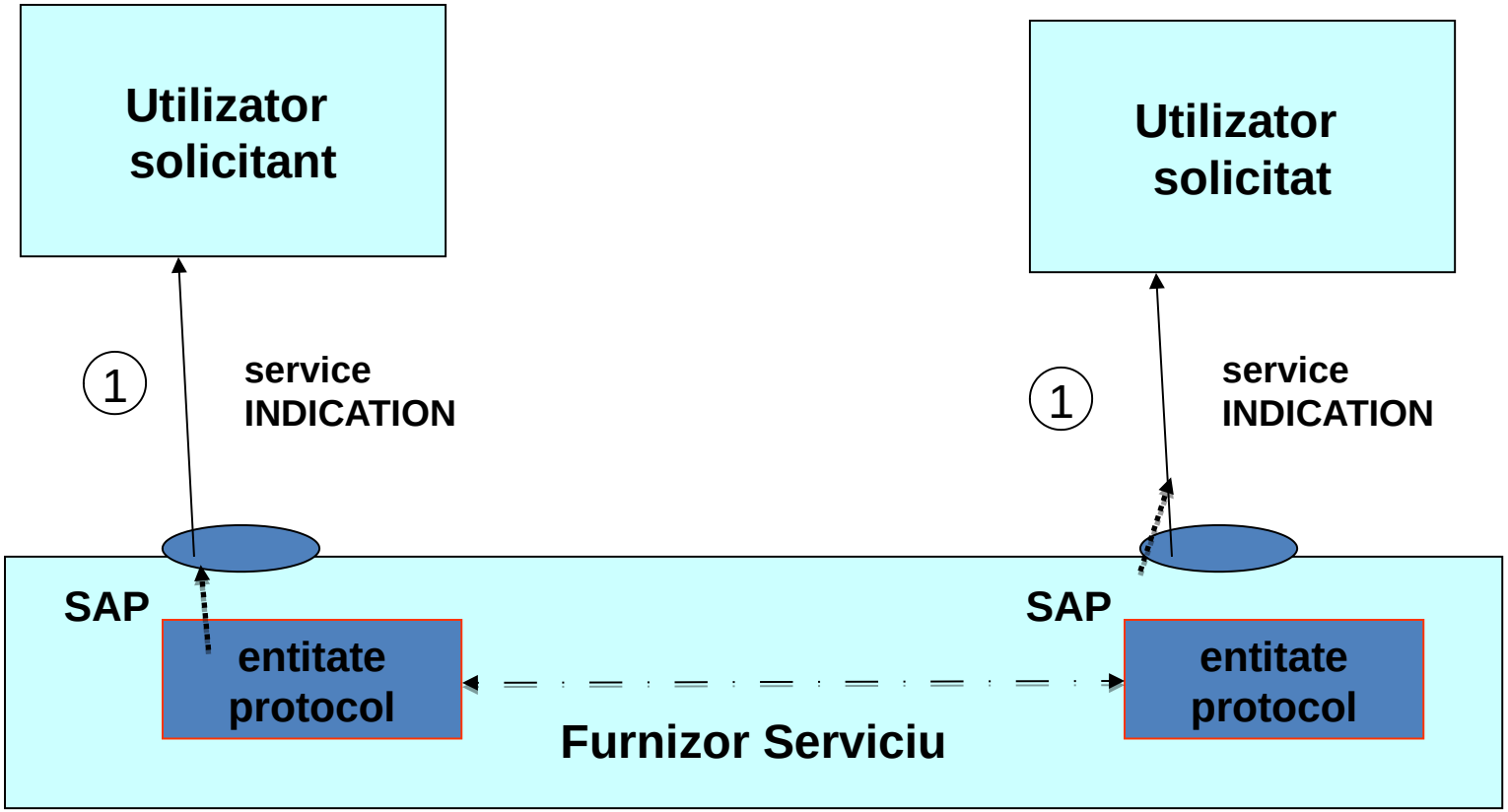
Entitatile de protocol pot inter-schimba mai multe mesaje pentru un singur serviciu confirmat

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

Servicii ne-confirmate



Servicii initiate de furnizor



Ce conţin specificaţiile ?

Specificaţie Serviciu

- primitive (operaţii)
- parametri
- reguli asupra ordinii operaţiilor (state machine)

Specificaţie Protocol

- scop şi funcţii
- servicii oferite
- servicii utilizate din nivel inferior
- structura internă (entităţi şi relaţii)
- tipuri şi formate mesaje schimbate între entităţi
- reguli de reacţie a fiecărei entităţi la comenzi, mesaje şi evenimente interne

Protocoale OSI

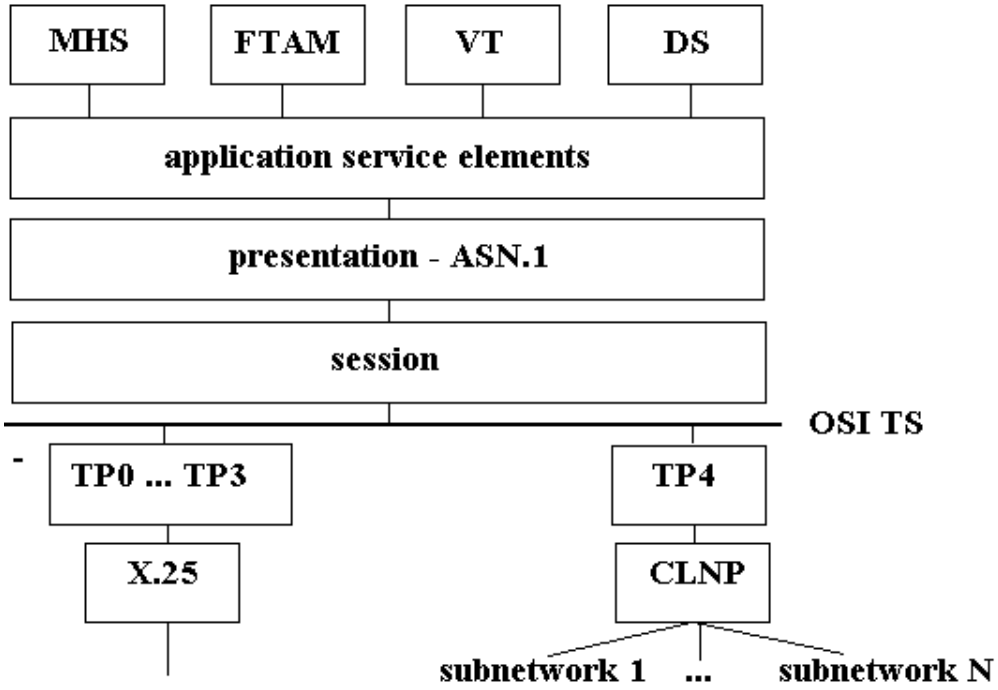
Physical layer: V10, V11, V24, V35
X.21, EIA RS-232-D
MAC for LANs
ISDN physical interface

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 referinţă TCP/IP

Nivelul inferior este legatura gazda-retea (host - network)

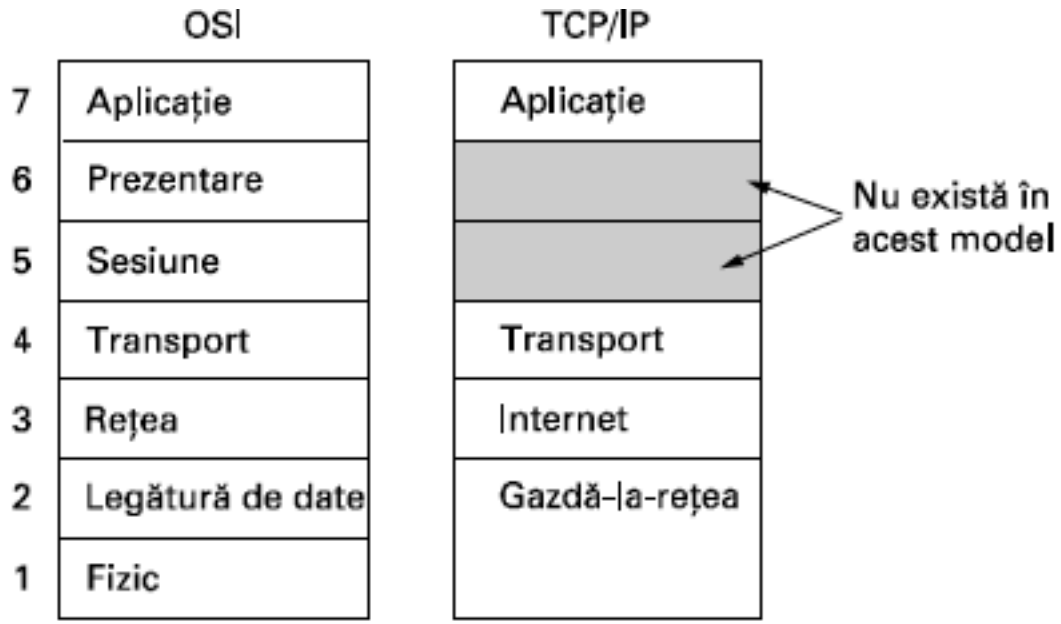
- mare varietate de protocoale
- modelul nu impune reguli despre acest nivel

Nivelul Internet

- un singur protocol, IP
- ptr. retele interconectate

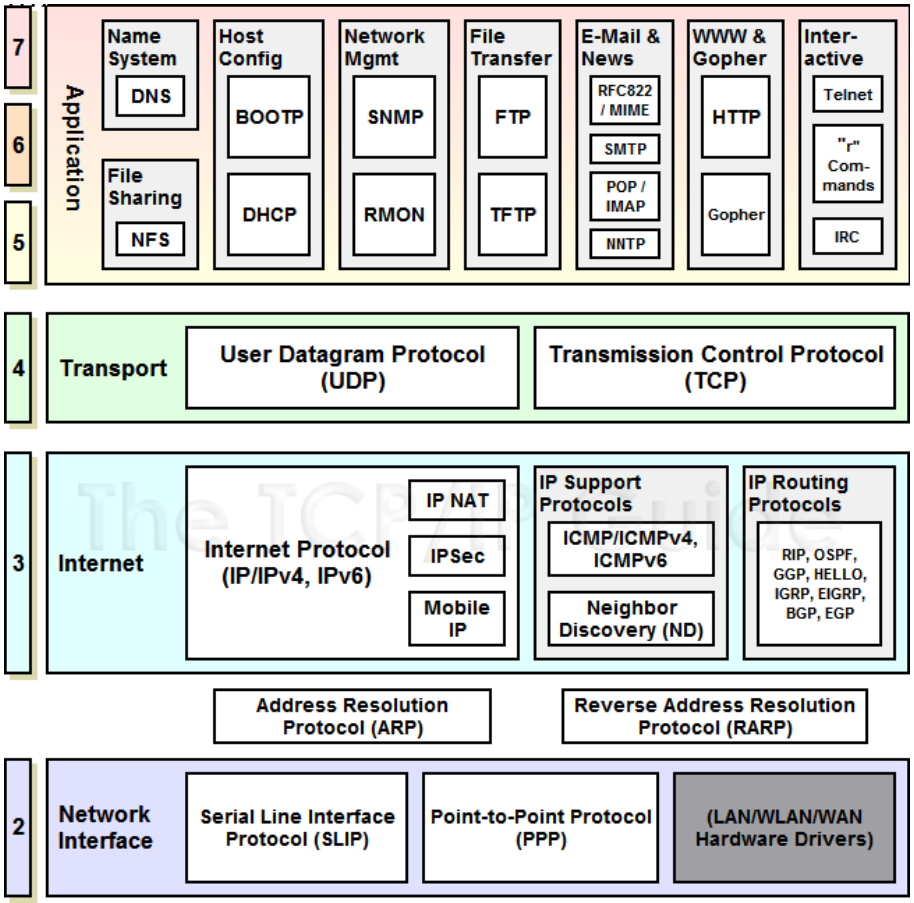
Nivelul Transport

- TCP – canal sigur pentru siruri de octeti
- UDP – canal nesigur pentru livrarea datagramelor (user datagram – sinonim pentru mesaje)



Protocoale în modelul TCP/IP

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



Source: http://www.tcpipguide.com/free/t_TCPIPProtocols.htm

Alte protocoale 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

Comparaţie OSI şi TCP/IP

Contra OSI

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

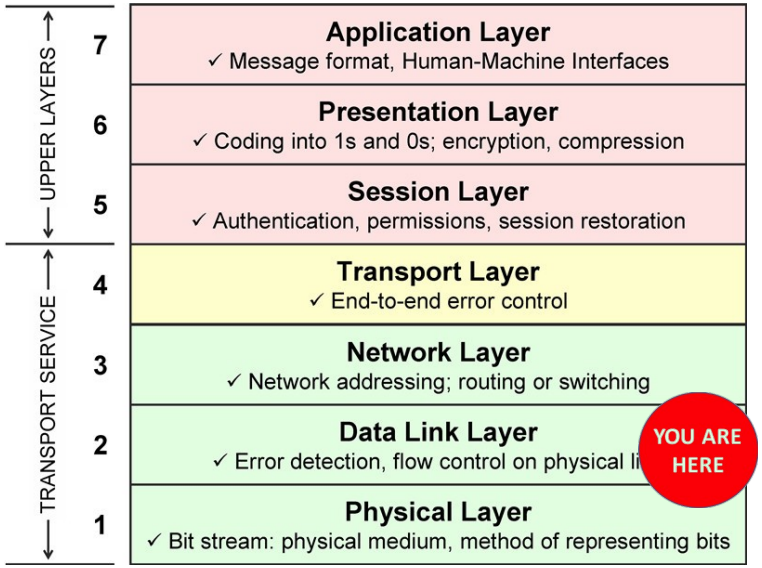
Contra TCP-IP

- Nu distinge între servicii, interfeţe, protocoale
- Nu este un model general
- “Nivelul” gazdă-reţea nu este un nivel
- Nu menţionează nivelele fizic şi legătură de date
- Protocoale minore bine înrădăcinate – ex Telnet (ptr. teletype)

Nivelul legăturii de date

Cuprins

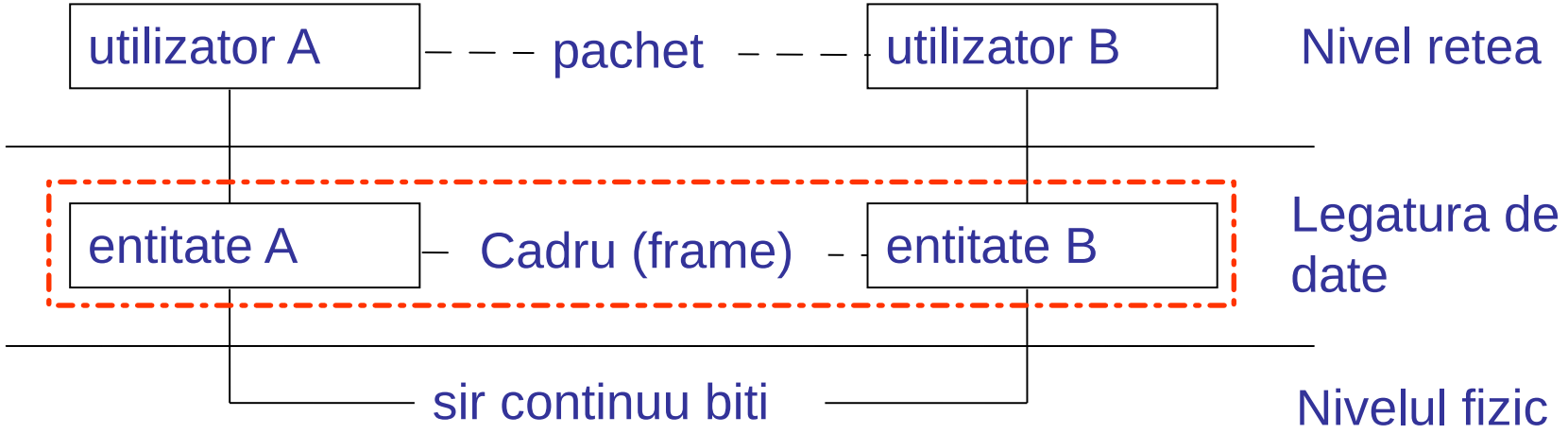
- Functiile legaturii de date
- Incadrarea datelor
- Transmisia transparenta
- Detectia erorilor de transmisie
- Protocoale start-stop
- Protocoale cu fereastra glisanta
- Exemple de protocoale:
- Ethernet, PPP



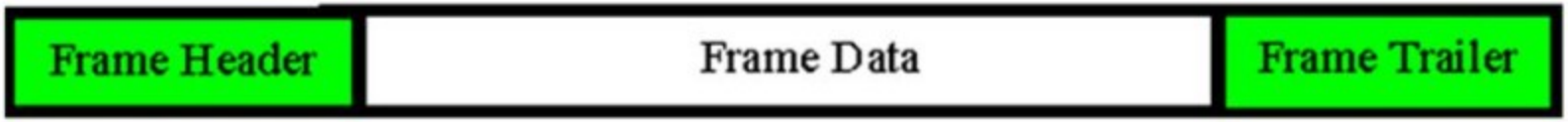
Foarte importante!

- implemantarea protocoalelor (programarea)
- orientarea pe evenimente
- tratarea erorilor (nu exista in programarea “obisnuita”)
- mecanismul ferestrelor glisante
- performanta (retransmitre neselectiva / selectiva)

Funcţiile nivelului legăturii de date



1. Încadrarea datelor
   - nivelul fizic transmite un sir continuu de biti
   - legatura de date partitioneaza sirul in cadre
   - delimiteaza cadrele
2. Transmisia transparentă
   - permite trimiterea in cadru a oricarei combinatii de biti
     - ex. care coincid cu delimitatorii cadrului



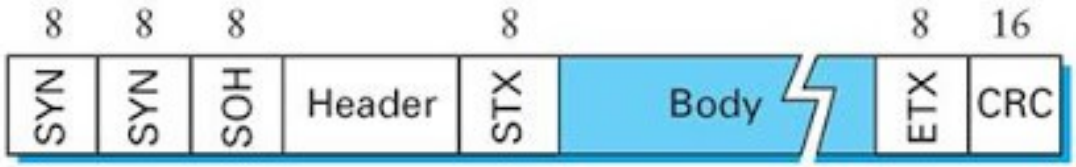
Funcţiile nivelului legăturii de date

3. Controlul erorilor
   - Coduri detectoare si corectoare de erori
   - Mecanisme de protocol
     - mesaje de confirmare
     - ceasuri
     - numere de secvenţă
4. Controlul fluxului – protocol
   - mesaje de permisiune pentru transmiţător
5. Gestiunea legăturii – protocol
   - stabilirea şi desfiinţarea legăturii
   - re-iniţializare după erori
   - configurarea legăturii (ex. legături multipunct)

- Moduri de transmisie:
  - **Full duplex** = Comunicația poate avea loc în ambele sensuri simultan
  - **Half duplex** = Comunicația poate avea loc în ambele sensuri, dar nu simultan
  - **Simplex** = O entitate poate doar transmite informații și cealaltă poate doar să primească

Metode de încadrare a datelor (Exemple)

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



BISYNC (Binary Synchronous Communication) protocol

SOH - start of heading
STX - start of text
ETX - end of text
ETB - end of transmission block
EOT - end of transmission
ENQ - enquiry
ACK - acknowledge
NAK - not acknowledge
SYN - synchronous idle
DLE - data link escape
CRC - cyclic redundancy check

- Începutul unui cadru este delimitat prin caracterul special **SYN.**
- Segmentul de date este încadrat de caracterele speciale **STX** și **ETX**.
- Începutul header-ului este marcat prin caracterul special **SOH.**
- **CRC** – folosit pentru detecția erorilor

Transmisia transparentă

Probleme cu abordarea bazată pe caractere de control:

- Datele trimise pot conține caractere de control în interior

În cazul transmisiei de conținut binar:

STX text… ETX …text ETX → ETX fals ?!

Soluție:

- umplere cu caractere *(Byte/Character stuffing)* - escaparea caracterelor de control, folosind caracterul special DLE

Umplere cu caractere

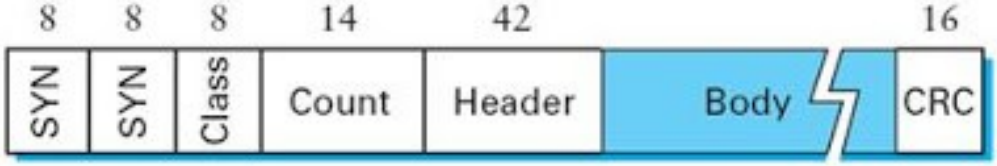
- Toate caracterele de control din corpul cadrului sunt escapate cu DLE (data link escape) înainte de transmisie
- Caracterul DLE este escapat și el (dublat) dacă apare în corpul cadrului.

- Sunt definite transformări admise în datele trimise
  - STX – > DLE STX
  - ETX – > DLE ETX
  - DLE – > DLE DLE

- Se consideră că a avut loc o eroare la receptie dacă se primește caracterul DLE urmat de altceva decât STX, ETX, DLE

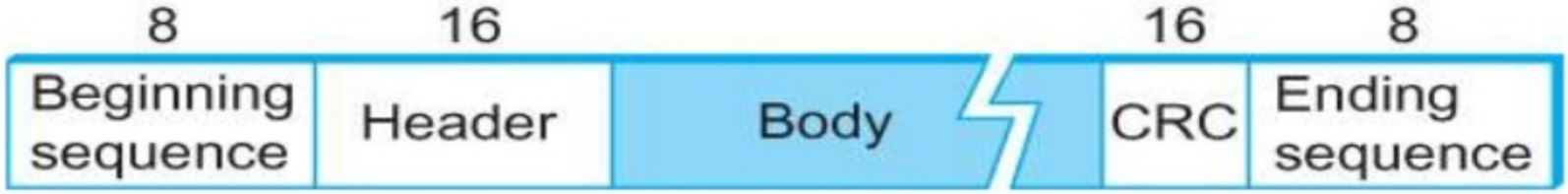
Metode de încadrare (Exemple)

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



DDCMP (Digital Data Communications Message Protocol)

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



Umplere cu biţi (Bit stuffing)

În cazul protocolului HDLC secvențele de început și de sfârșit al cadrului sunt 01111110.

Datele de transmis pot include un flag fals de terminare a cadrului

Date de trimis: 011011111101111111111010

Solutia: se adauga un zero dupa fiecare 5 unitati consecutive

01111110 011011111010111110111110010 01111110

Bitul 0 inserat se elimină la recepție

- Adaugarea se face indiferent daca dupa 5 unitati urmeaza 0 sau 1
- Simplifica regula receptorului: elimina zerouri aflate dupa 5 unitati

Date primite: 011011111010111110111110010

După eliminarea biților inserați: 011011111101111111111010

Nivelul legăturii de date

Detecţia și corectarea erorilor

Detecţia și corectarea erorilor

Descrierea problemei:

- Fie A = {0,1} un alfabet binar
- W_n multimea sirurilor w de lungime n peste A

  W= w[0] w[1]……..w[n-1]
- **Ponderea Hamming** a lui w = numarul de unitati (1) continute de w
- **Distanta Hamming**, d(u,v) dintre cuvintele u si v reprezinta numarul de simboluri diferite dintre cele doua. Este echivalent cu numarul minim de erori de 1 bit de care ar fi nevoie pentru a transforma unul din cuvinte in celalalt.

Detecţia erorilor si corectarea erorilor

Se considera ca pentru transmiterea de mesaje corecte se foloseste doar un subset din W_n

Multimea cuvintelor posibile W_n se imparte in:

- S_n – multimea cuvintelor cu sens
- F_n – multimea cuvintelor fara sens
- Pentru <u>a detecta</u> a cel mult *k* erori se alege S_n astfel ca:

  d(u,v) >= k+1 pentru orice u,v din S_n
- Pentru <u>a corecta</u> cel mult *k* erori se alege S_n astfel ca:

  d(u,v) >= 2*k+1 pentru orice u,v din S_n

Exemplu

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

d(u,v) = 5 ====>>> putem detecta erori de maxim 4 biti
putem corecta erori de maxim 2 biti

Biți de paritate

- Metodă foarte simplă.
- Se adaugă un singur bit pentru fiecare caracter
- Paritate pară – numărul de biți 1 este par după adaugarea bitului
- Paritate impară – numărul de biți 1 este impar după adaugarea bitului

- Receptorul recalculează bitul de paritate
- Se consideră eroare dacă bitul primit diferă de cel recalculat

- Nu este o metodă foarte sigură. Dacă are loc un număr par de erori, verificarea bitului de paritate nu le va detecta.

7 biți de date	nr. de biți 1	8 biți cu paritate	
		pară	impară
0000000	0	0000000**0**	0000000**1**
1010001	3	1010001**1**	1010001**0**
1101001	4	1101001**0**	1101001**1**
1111111	7	1111111**1**	1111111**0**

Suma de control (checksum)

- Se calculează suma valorilor zecimale a fiecărui caracter din mesaj
- Se împarte rezultatul la 255
- Restul împărțirii se salvează și se adaugă la mesaj

- Are o eficiență de ~ 95%

Metoda Hamming

Se considera un sir de n biti numerotati de la 1 la n.
Bitii 1,2,4,8…..(puteri ale lui 2) sunt **biti de control**

Exemplu: pentru un payload de 8 biti este nevoie sa trimitem 4 biti de control (1,2,4,8)

Paritatea pentru bitii de control poate fi **pară** sau **impară**, dupa urmatoarea regula:

Bit 1 – controleaza bitii 1,3,5,7,9,11
Bit 2 – controleaza bitii 2,3,6,7,10,11
Bit 4 – controleaza bitii 4,5,6,7,12
Bit 8 – controleaza bitii 8,9,10,11,12

Regulă: fiecare bit k este controlat de bitii ale caror pozitii insumate dau k.

Metoda Hamming

Exemplu: Se dorește trimiterea șirului de biți 1100001

Se adaugă biții de paritate la trimitere:

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

Obs: Codul Hamming poate corecta erorile de 1 bit

Metoda Hamming-Corectarea erorilor

Codul Hamming se poate folosi pentru corectarea erorilor de 1 bit.

Se pot corecta erorile in rafala (en. burst) in situatia urmatoare:

- Matricea de biti este transmisa coloana cu coloana
- Se pot corecta erori in rafala dintr-o coloana daca exista 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

Detecţia erorilor cu coduri polinomiale

k biţi de informaţie (date) ~ coeficientii unui polinom **i(X)** cu k termeni

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

n-k biţi de control ~ polinom **r(X)** cu n-k termeni
concatenarea de **n** biţi ~ polinom de n termeni

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

Pentru detectie erori
se foloseste un polinom generator **g(X)**
se calculeaza **r(X)** astfel ca **w(X)** sa fie multiplu de **g(X)**
se verifica daca proprietatea este pastrata la receptia cadrului
Calcul r(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 împărţirii lui $\mathbf{X^{(n-k)}}$ **i(X)** la **g(X)**

Calculul sumei de control

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

Calculul sumei de control pentru un cod polinomial

10 biti informatie + 4 biti control

Imparte **11010110110000**

la 10011

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

$g(X)$

```
                1 1 0 0 0 0 1 0 1 0
10011 | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
        1 0 0 1 1
          1 0 0 1 1
          1 0 0 1 1
            0 0 0 0 1
            0 0 0 0 0
              0 0 0 1 0
              0 0 0 0 0
                0 0 1 0 1
                0 0 0 0 0
                  0 1 0 1 1
                  0 0 0 0 0
                    1 0 1 1 0
                    1 0 0 1 1
                      0 1 0 1 0
                      0 0 0 0 0
                        1 0 1 0 0
                        1 0 0 1 1
                          0 1 1 1 0
                          0 0 0 0 0
                            1 1 1 0   Remainder
```

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

Transmitted frame: 1 1 0 1 0 1 1 0 1 1 $\boxed{1\ 1\ 1\ 0}$

Ce erori pot fi detectate?

- Probabilitatea de detectie depinde de lungimea Codului cu Redundanta Ciclică

  CRC–8–Bluetooth = $x^8 + x^7 + x^5 + x^2 + 1$

  CRC-8-CCITT = x^8+x^2+x+1

  CRC-12 = $x^{12}+x^{11}+x^3+x^2+x+1$

  CRC-16 = $x^{16}+x^{15}+x^2+1$

  CRC-32 = $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$ (Ethernet)

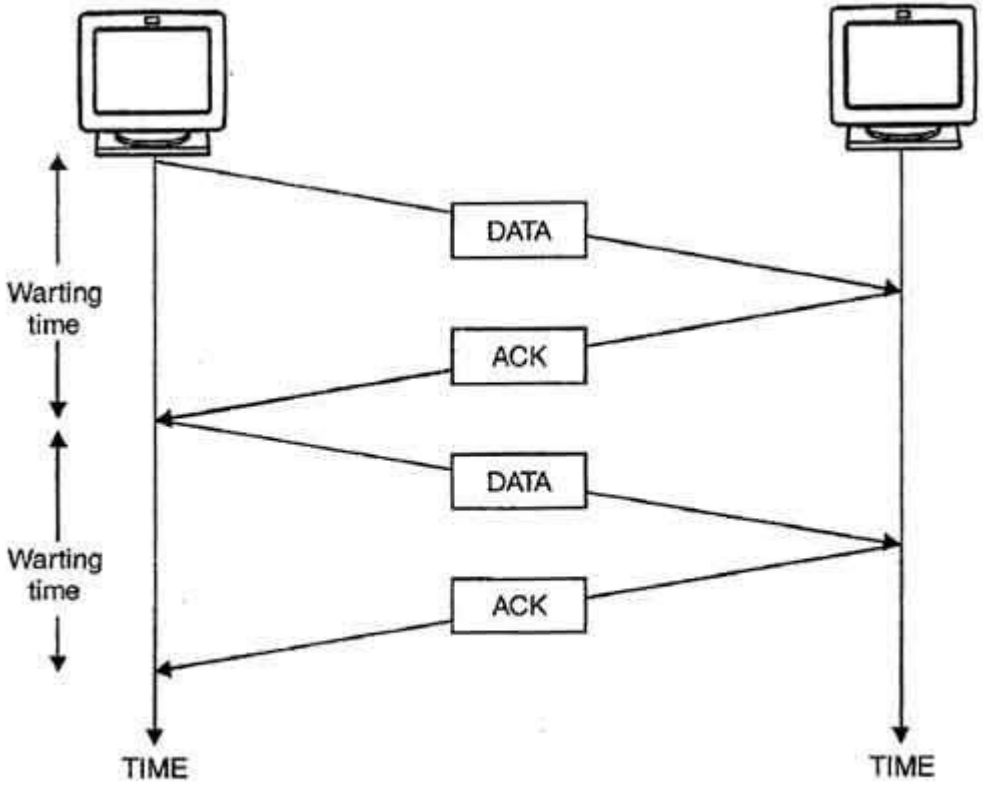
- CRC si sume de control pe
  - 8 biti detecteaza 99.6094% din erori (CRC-8-CCITT)
  - 16 biti detecteaza 99.9985% din erori (CRC-16)
  - 32 biti detecteaza 99.9999% din erori

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

Nivelul legăturii de date

Protocoale start-stop

Protocoale Stop and Wait

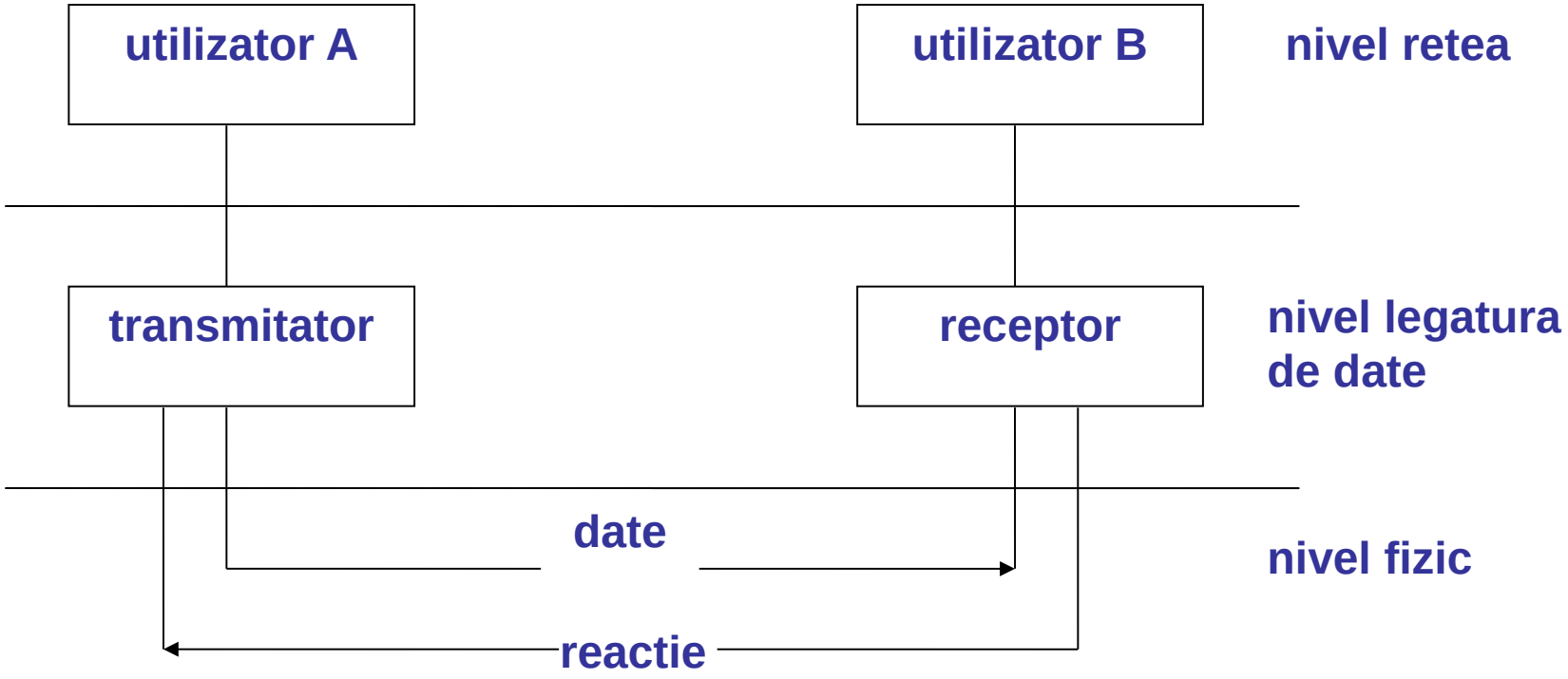


Stop & Wait Method.

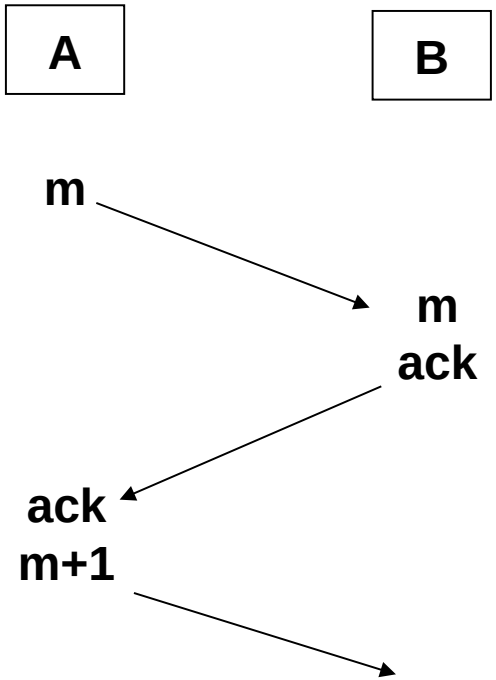
Protocol Stop and wait

- Sender-ul trimite un singur cadru si asteapta primirea unei confirmari (ACK)
- Transfer bidirectional de informatie, se poate realiza printr-un canal half-duplex.
- Se doreste impartirea informatiei trimise in cadre de dimensiune mai mica deoarece:
  - Buffer-ul receiver-ului ar putea fi limitat
  - Cadrele mai mici au o probabilitate mai mica de eroare
  - In cazul unei erori retransmisia unui cadru mic va consuma mai putine resurse.
  - In cazul folosirii unui mediu partajat de transmisie (LAN), un singur sender nu va ocupa mediul pentru un timp prea lung.

Protocol start-stop

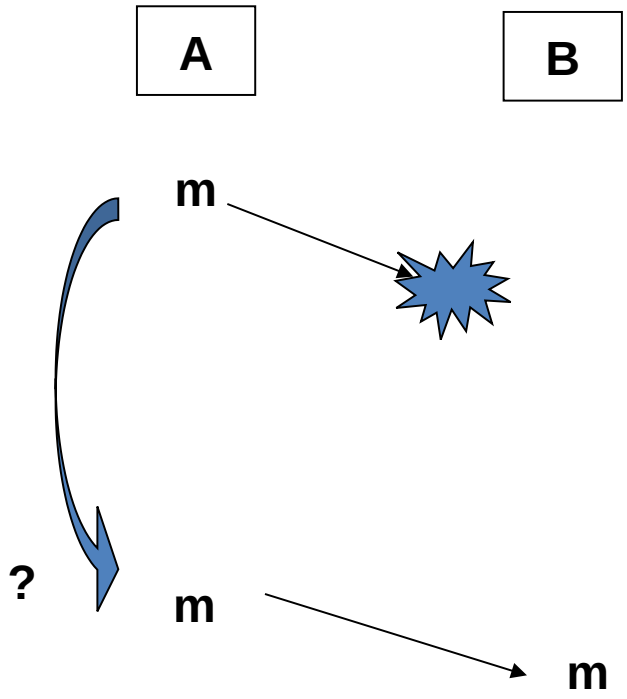


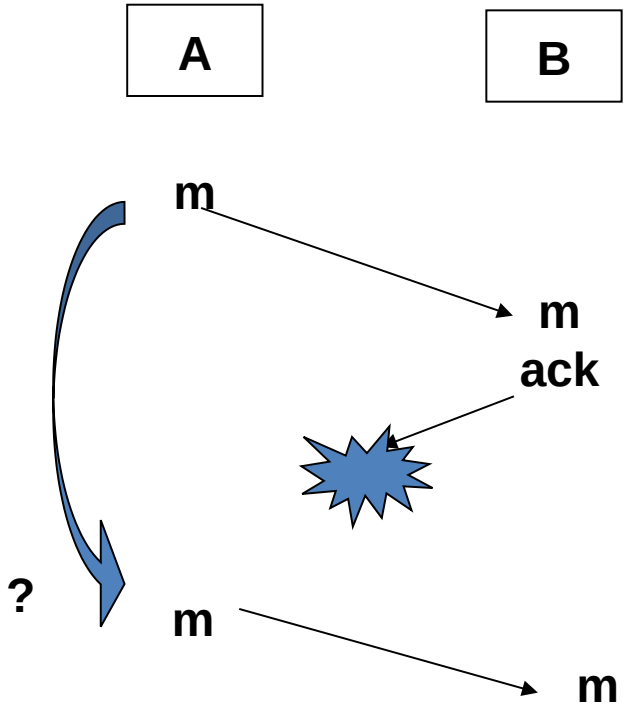
Protocol simplex pentru un canal cu erori



Entitaţi legătură de date

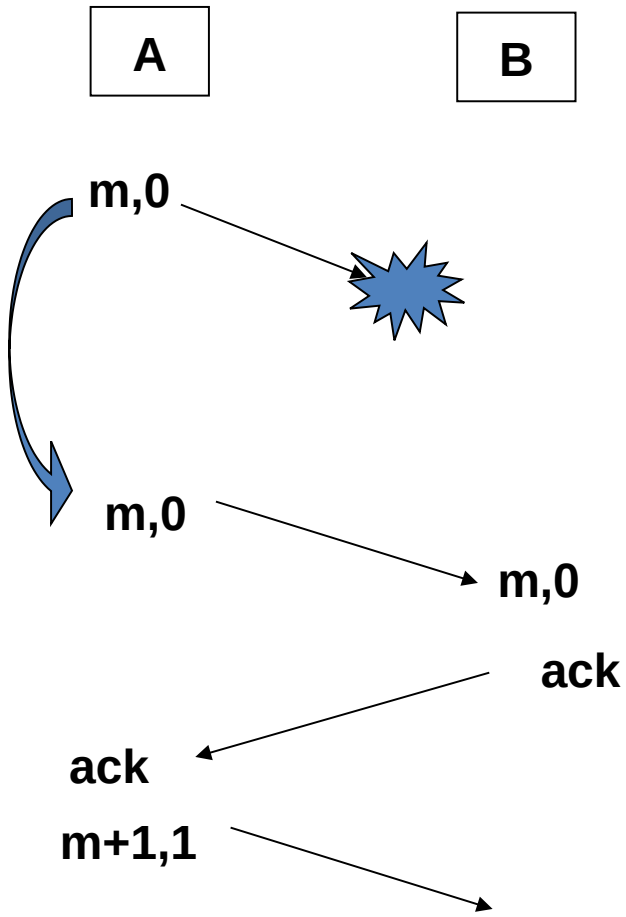
Transmisie corecta

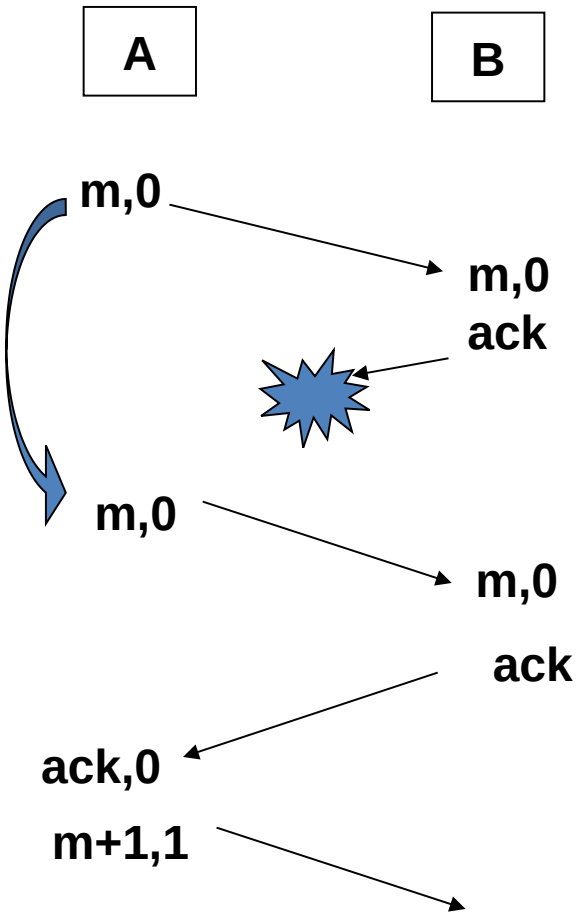




Pierdere m
La time-out A retransmite m
Care este acceptat corect de B

Pierdere ack
La time-out se retrimite m
Care este acceptat incorect, ca mesaj nou de B !





- accepta

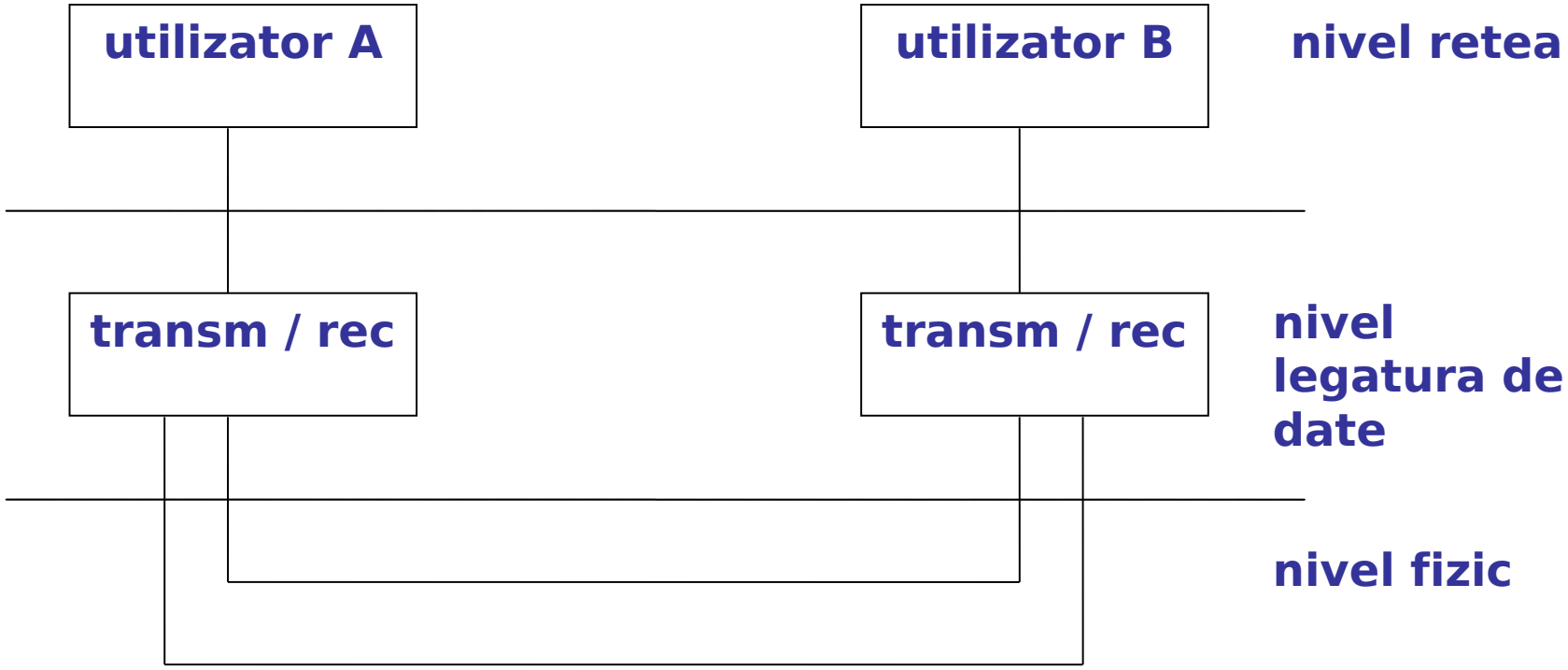
- ignora

se adauga un numar de secventa
la time-out se re-transmite ultimul cadru
B accepta daca este corect

B ignora daca este dublura
dar re-trimite ack !

Protocoale cu fereastră glisantă

Configuraţia generala



2 legaturi pentru date+confirmare

Protocoale cu fereastra glisanta

- Daca este nevoie de comunicatie full duplex se folosesc 2 canale separate de comunicatie: unul pentru trimiterea datelor si al doilea pentru confirmari ACK/NACK.

- Sender-ul pastreaza o lista de cadre ce trebuie sa fie trimise. Receiver-ul pastreaza o lista de cadre pe care le poate accepta.

- Numerele de secventa pentru cadre trimise dar pentru care inca nu s-a primit un ACK sunt pastrate in fereastra glisanta.

- Numerele de secventa din ferestrele glisante sunt consecutive.

Fereastra glisanta de 1 bit

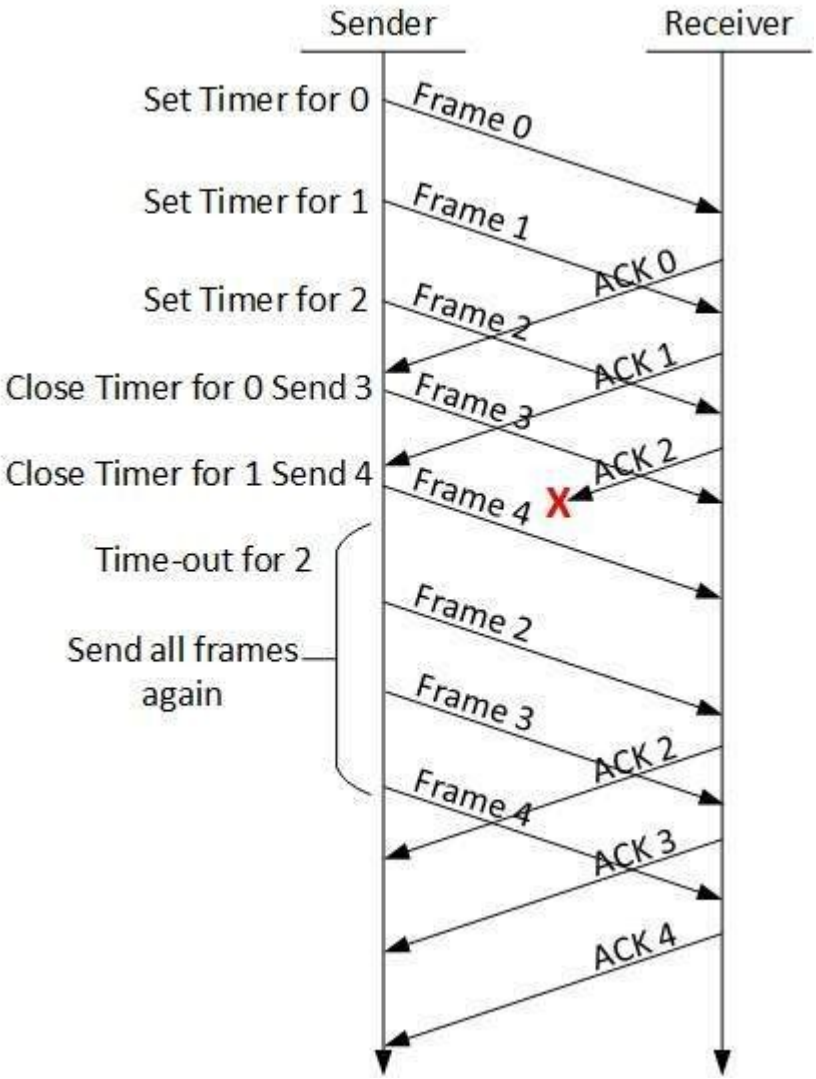
- **Fiecare entitate a**re o secventa de initializare in care trimite un prim cadru

  si realizează ciclic următoarele operaţii:

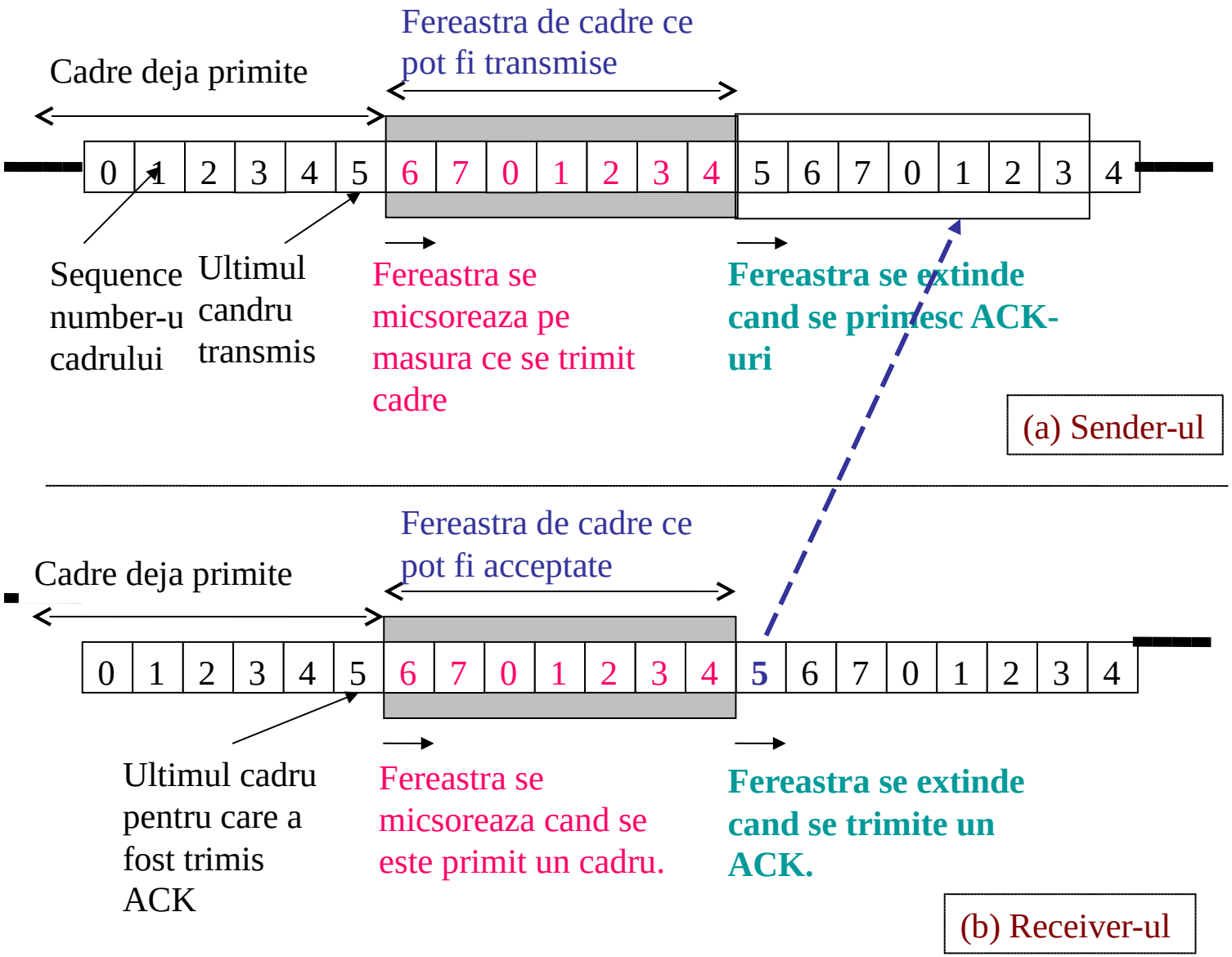
  - receptia unui cadru,
  - prelucrarea sirului de cadre receptionate,
  - prelucrarea sirului de cadre transmise,
  - transmiterea sau retransmiterea unui cadru impreuna cu confirmarea cadrului receptionat corect.

  Se ajunge la modul de transmisie Stop and wait, in care sender-ul trimite un cadru si asteapta primirea unui ACK inainte sa il trimita pe urmatorul.

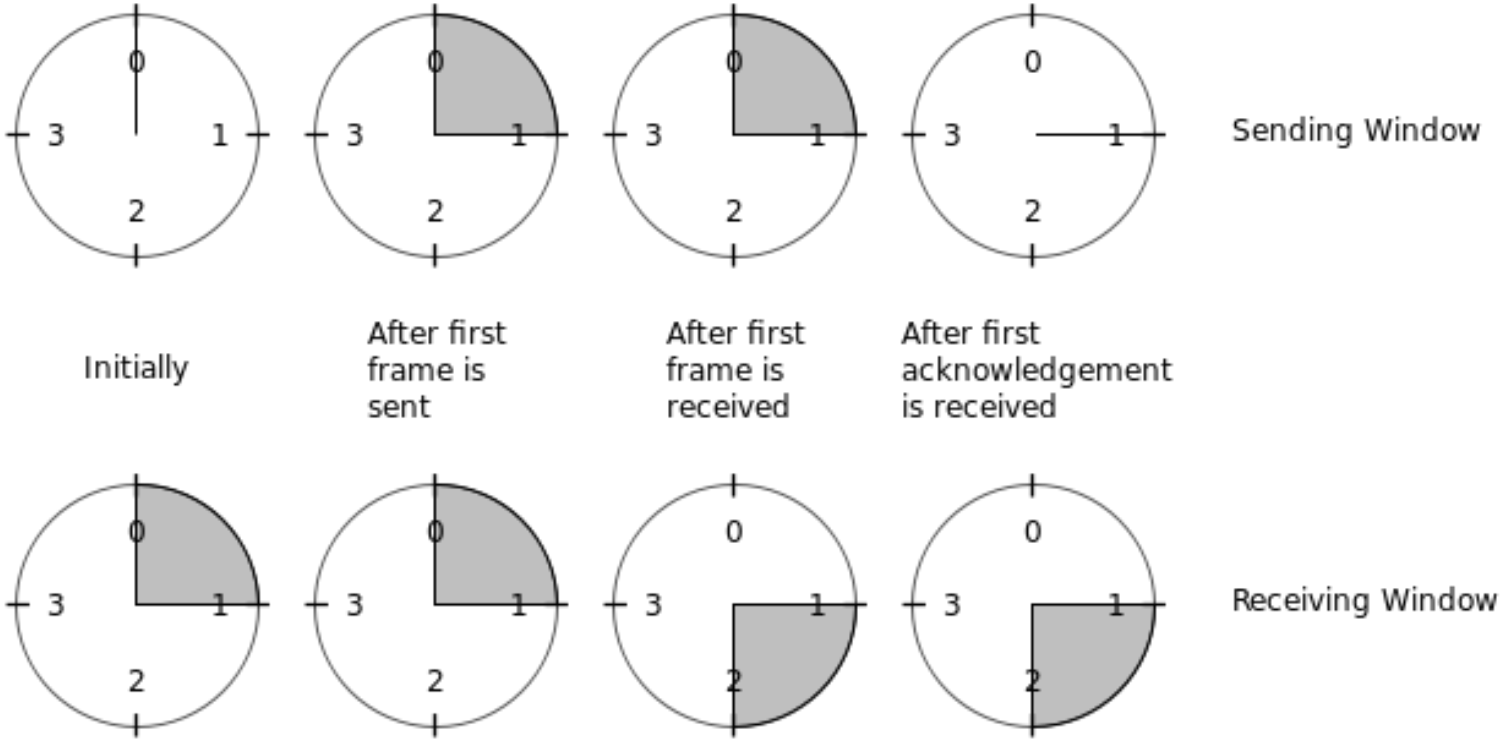
Protocoale cu fereastra glisanta



Protocoale cu fereastra glisanta

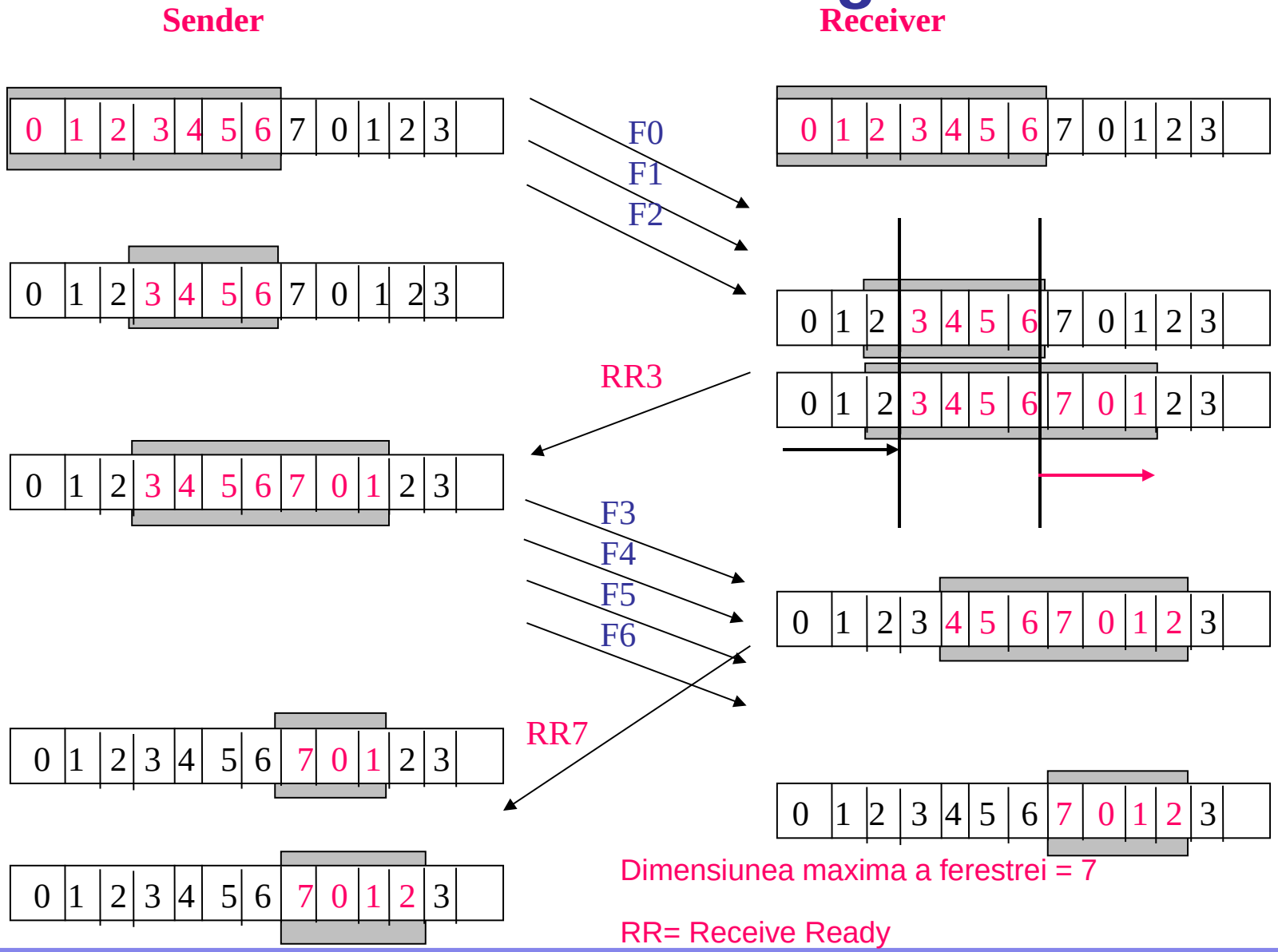


Fereastra transmitatorului



A sliding window with a 2-bit sequence, of size 1

Protocoale cu fereastra glisanta



Dimensiunea maxima a ferestrei = 7

RR= Receive Ready

Protocoale cu fereastră supraunitară de transmisie

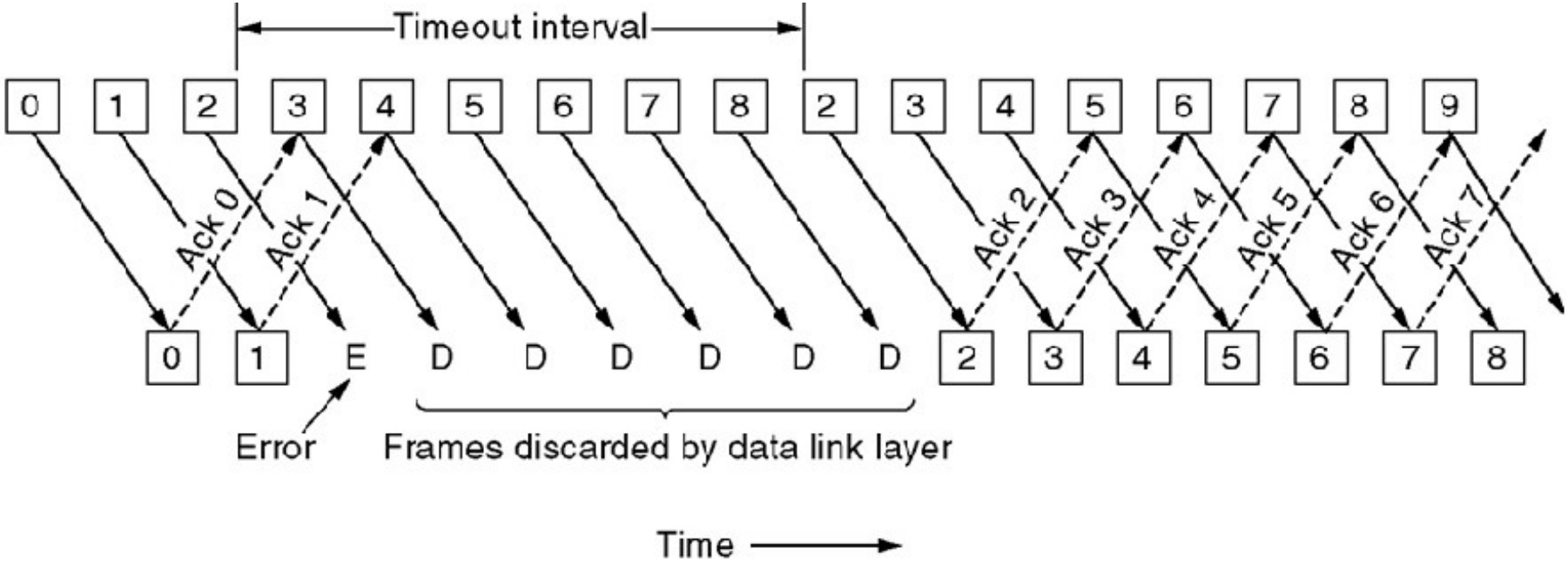
Protocol cu retransmitere neselectivă (Go back n)

Sunt **`MaxSecv + 1`** numere de secventa diferite
Fereastra maxima a transmitatorului poate fi de **`MaxSecv`** cadre
Demonstratie pe scenariu cu **`MaxSecv = 7`** si fereastra de 8

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

Un protocol cu retransmitere neselectivă "Go Back N"

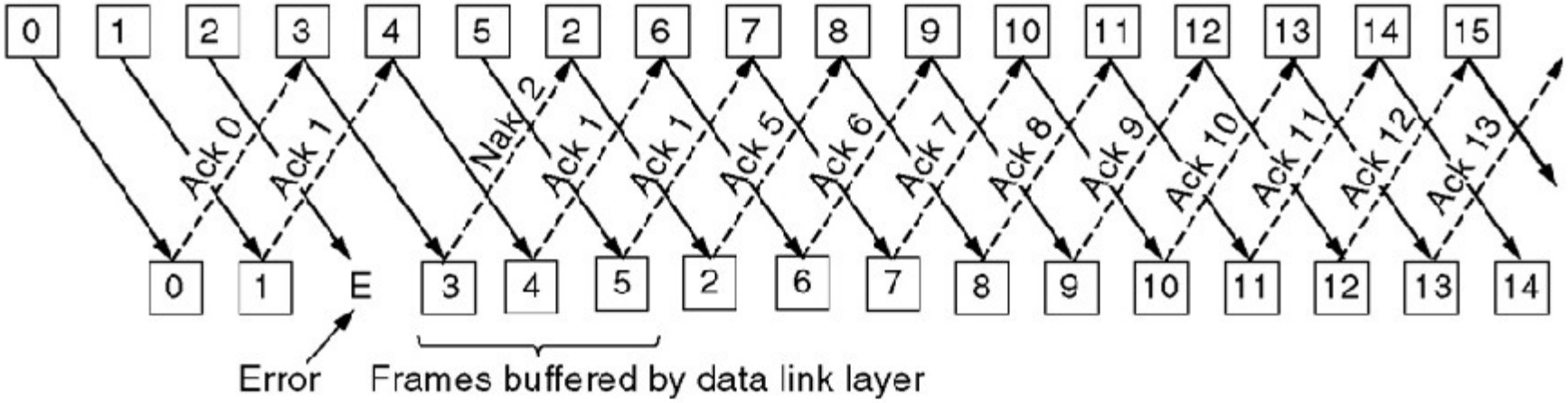
Efectul erorii cand fereastra receptorului este 1.



Protocol cu retransmitere selectiva "selective repeat"

Fereastra receptorului este supraunitara

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



Optimizari protocoale cu fereastra glisanta

- Receiver-ul poate confirma un cadru si sa anunte ca asteapta urmatoarele (**Receive Ready**) sau poate sa nu permita trimiterea de alte cadre (**Receive Not Ready**). Este necesara trimiterea unui Acknowledgement normal pentru a relua trimiterea datelor dupa un cadru RNR.
- Este ineficienta trimiterea unui cadru ACK separat – se foloseste **piggybacking**. ACK-ul este atasat intr-un cadru de date (camp din header pentru ACK)

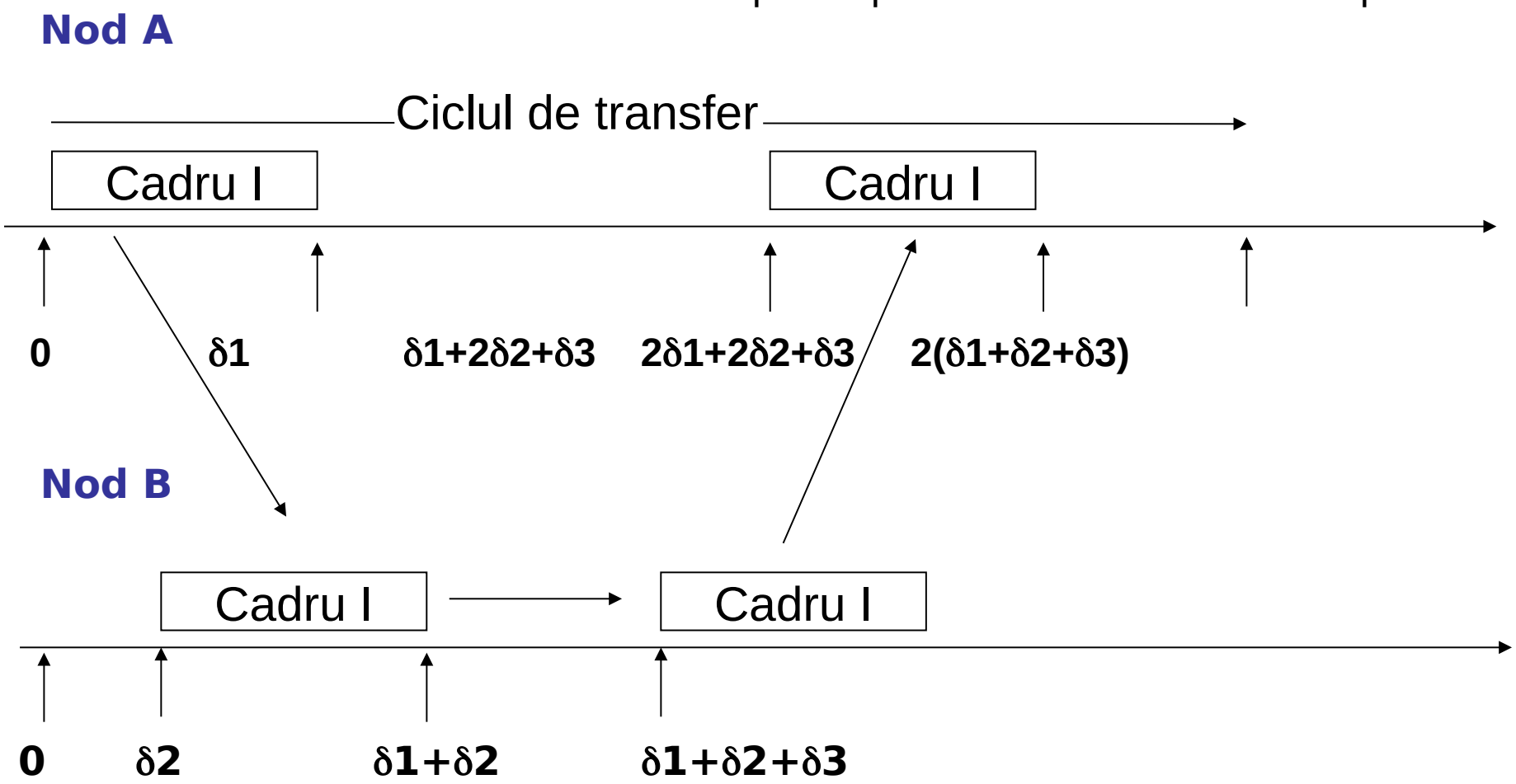
Nivelul legăturii de date

Metrici de performanta

- Analiza performanţelor protocoalelor start-stop

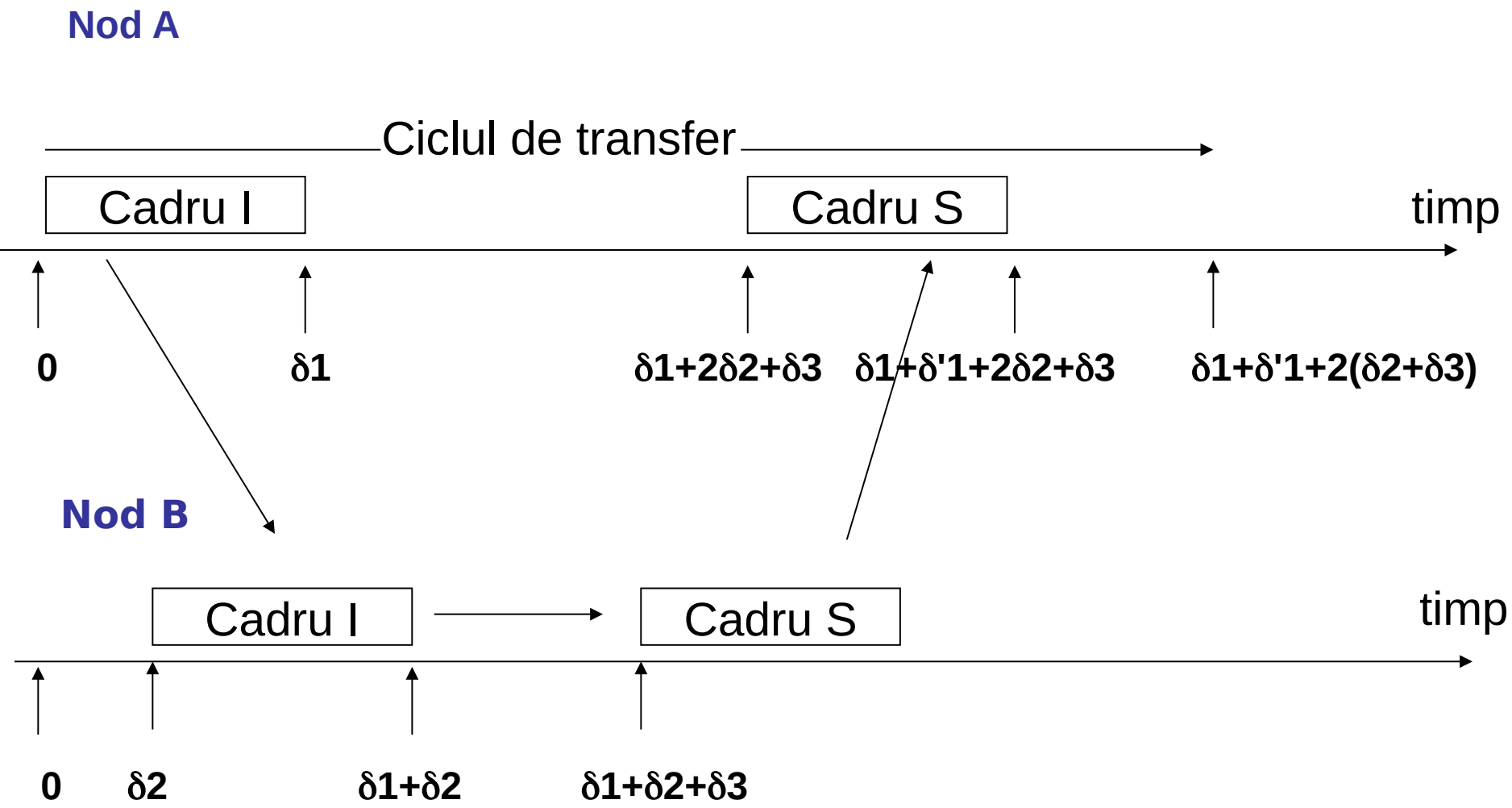
Transmitere cu confirmare în cadre de informatie I

∂1 – durata de transmitere a unui cadru I (sec)
∂2 – intarzierea de transmisie
∂3 – timpul de prelucrare a cadrului la receptor.



Transmitere cu confirmare în cadru supervizor S

∂'1 – durata de transmitere a unui cadru S (sec)



Eficienţa în absenţa erorilor

Cazul confirmării prin cadre S

ρ = timpul de transmitere a informaţiei / durata unui ciclu de transfer

ρ = **$\delta 1$ / ($\delta 1+\delta' 1 + 2(\delta 2+\delta 3)$)**

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 = $2(\delta 2+\delta 3)$, **latenţa** (sec)

• Exemple

1) Canal lent, distanţă mică

D = 352 biţi H = 48 biţi

$\delta 2$ = 5 msec

$\delta 3$ = 1 msec

Capacitatea canalului C = 9600 biţi / sec

Rezultă:

$L = 2(\delta 2+\delta 3) = 12$ msec

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

(2) Canal rapid, latenţă mare

D = 104 biţi H = 48 biţi

$\delta 2$ = 100 msec

$\delta 3$ = 1 msec

Capacitatea canalului C = $150*10^6$ biţi / sec

Rezultă:

L = 202 msec

ρ **=** D / (LC+2H+D) = **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 aropiate unul de altul) este de aproximativ 55 Gm ($55*10^9$ m).

Datele traverseaza legatura la viteza luminii = $3*10^8$ 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 trimite pe Pamant. In cat timp poate ajunge poza la Centrul de Control al Misiunii de pe Pamant? Fisierul are volumul de 5 MB.

Conventii

1 B = 8 biti

In transmisii de date

1 kb = 10^3 b = 1 000 b

1 Mb = 10^6 b = 1 000 000 b

1 Gb = 10^9 b = 1 000 000 000 b

In calculatoare, volumul datelor

1 kb = 2^{10} b = 1 024 b

1 Mb = 2^{20} b = 1 048 576 b

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

Raspuns

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

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

RTT este 368 secunde.

(b) Produsul intarziere * largime_banda pentrul legatura

$184 \text{ s} * 128*10^3 \text{ b/s} = 23552000 \text{ b} \approx 3 \text{ MB}$

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

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

Intarzierea de propagare + timp transmitere

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

Throughput

- O metoda mai buna decat eficienta
- Numarul total de biti transmisi, tine cont de
  - Bitii de overhead
  - Nevoia de a retransmite unele cadre
- Complex de calculat, depinde de
  - Eficienta transmisiei
  - Rata de erori
  - Numarul de retransmiteri
- Transmission Rate of Information Bits (TRIB)
  - Folosit ca o metoda de a masura throughput-ul

TRIB

= Numarul de biti de informatie primiti / timpul total necesar

$$\frac{\text{(numarul de biti de informatie) (Probabilitatea de transmisie corecta)}}{\text{timpul necesar de transmisie + intarzierea de propagare}}$$

$$\text{TRIB} = \frac{K\,(M - C)\,(1 - P)}{(M / R) + T}$$

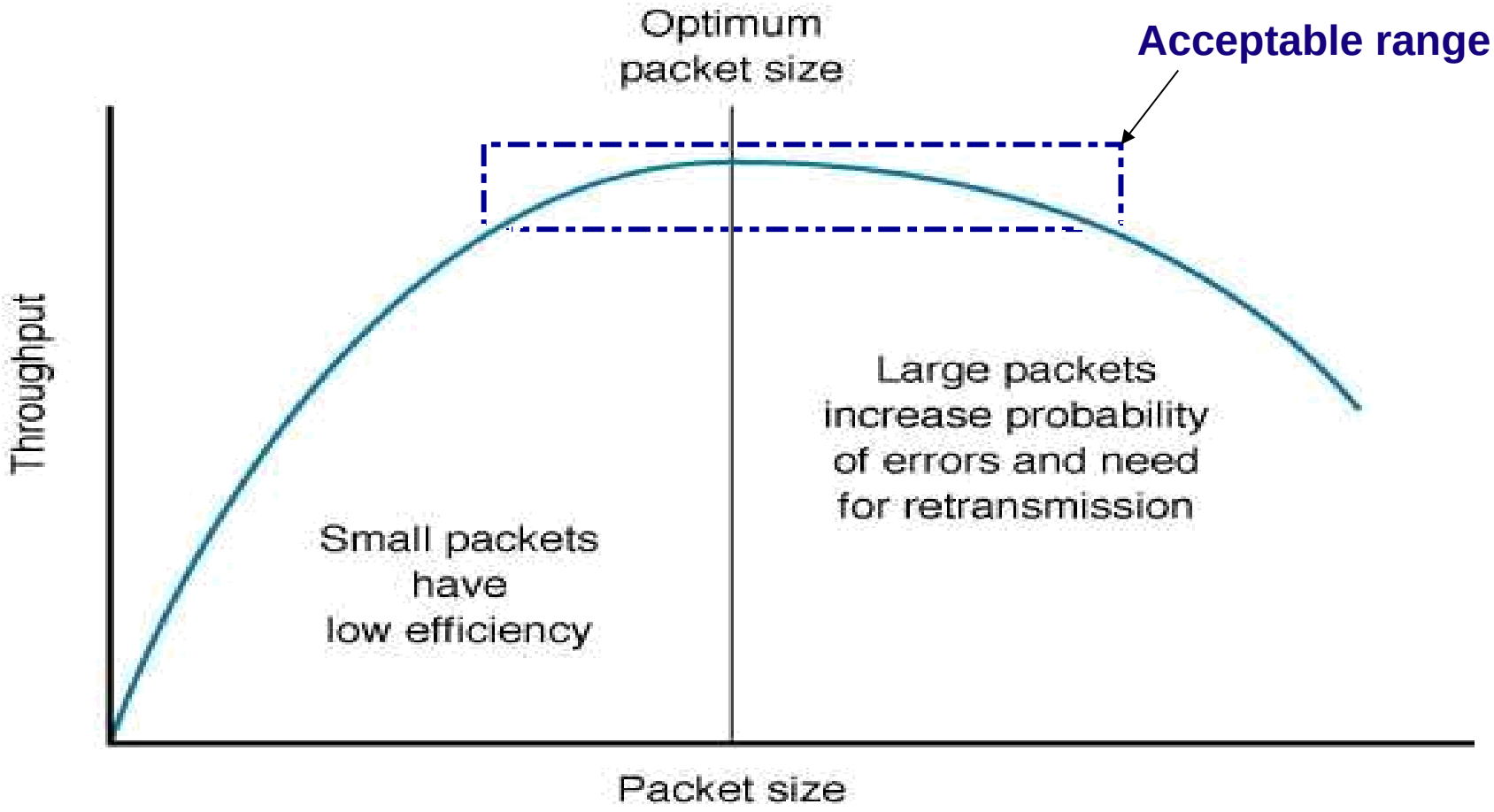
- K: Biti de informatie per caracter
- C: Media numarului de biti care nu sunt biti de informatie
- P: Probabilitatea ca un cadru va avea nevoie de retransmisie
- M: Lungimea unui cadru
- R: Rata de transmisie pe secunda
- T: Timpul intre cadre (intarziere, timp de procesare, etc.)

Ex:
K=7 bits/character
M = 400 char/cadru
R= 4.8 Kb/s
C = 10 char/block
P = 1%
T = 25 ms

$$\text{TRIB} = \frac{7(400-10)(1-0.01)}{(400/600)+0.025)}$$

= 3.908 Kb/s

Dimensiunea unui cadru

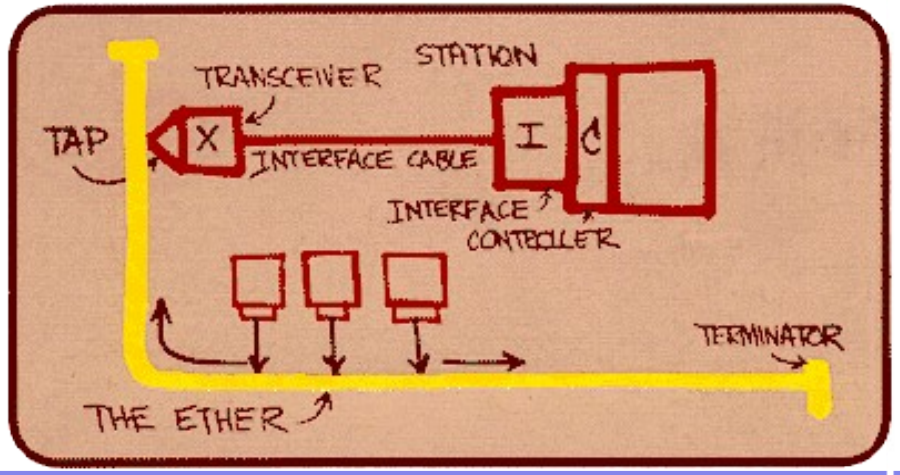


Exemple Protocoale Legatura de date

- Ethernet
- PPP
- PPPoE

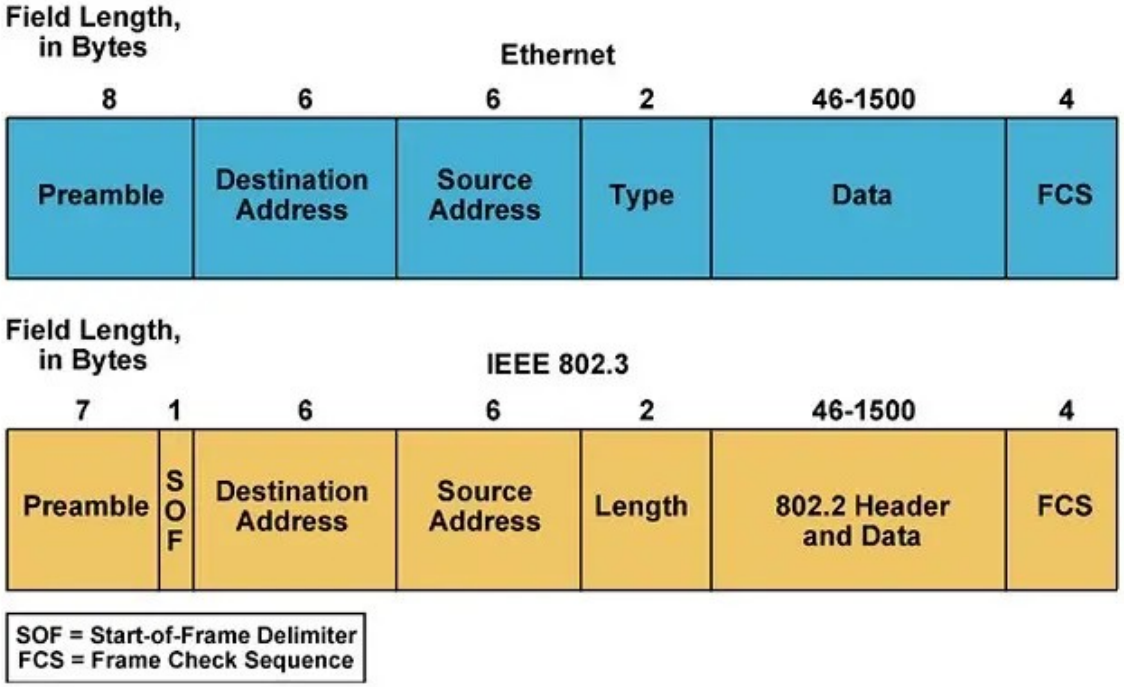
Protocolul Ethernet (IEEE 802.3)

- Cel mai folosit protocol LAN, dezvoltat de catre Digital, Intel si Xerox in anii 1970 de catre Bob Metcalfe si altii.
- Acum este un standard IEEE
- Protocol de nivel legatura de date, proiectat ca protocol broadcast
- Nu are probleme de transmisie transparenta a datelor
  - Foloseste un camp care contine numarul de bytes pentru a delimita cadrele
- Corectia erorilor este se realizeaza folosind CRC-32



Bob Metcalfe –
schita initiala pentru Ethernet

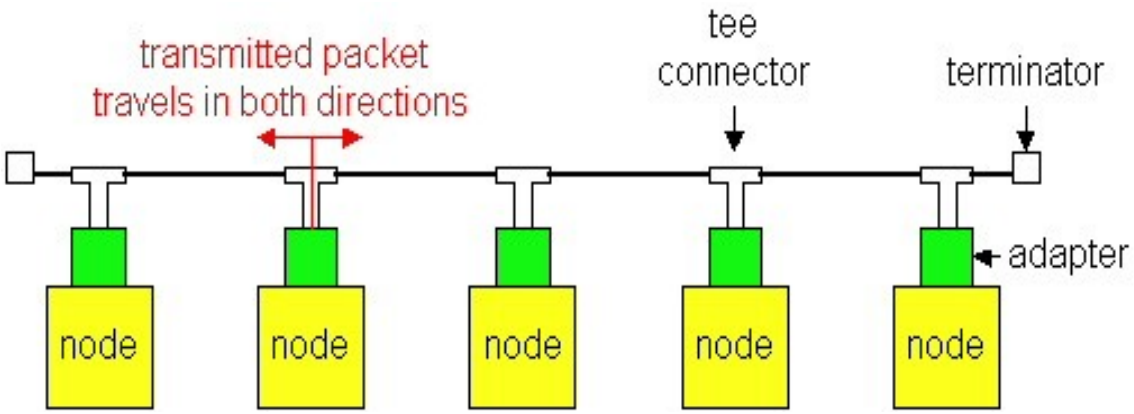
Protocolul Ethernet (IEEE 802.3)



- **Preamble** – un pattern de biti 0 si 1 care alterneaza (01010101….)
- **SOF** – delimiteaza inceputul cadrului (1010101**1**)
- **FCS** – verificarea corectitudinii transmisiei folosind CRC-32

Protocolul Ethernet (IEEE 802.3)

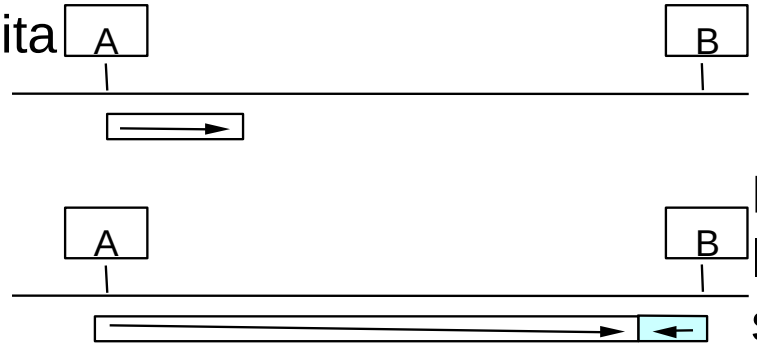
- Cadrul poate sa aiba minim 64 bytes si maxim 1516 bytes (fara preambul) – 12 bytes adrese, 2 bytes lungimea cadrului, 46 bytes date, 4 bytes CRC
- Daca un host are de trimis mai putin de 46 de bytes, transmitatorul va adauga padding.



Coliziuni

- Coliziunile apar cand 2 adaptoare incearca sa trimita in acelasi timp
- Daca este detectata o coliziune se asteapta si se incearca din nou dupa un timp random
- Modul de trimitere se bazeaza pe Carrier-sense multiple access with collision detection (CSMA/CD).
  - MA = multiple access
  - CS = carrier sense
  - CD = collision detection

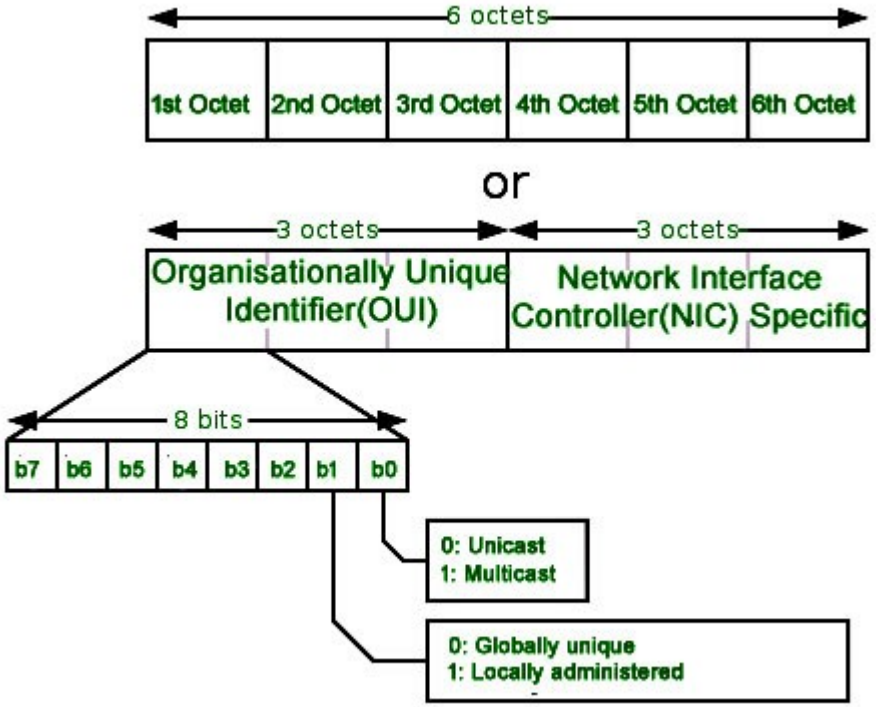
A incepe sa trimita la timpul 0



Mesajul aproape a ajuns la destinatie cand B incepe sa trimita --- > Coliziune!

Protocolul Ethernet (IEEE 802.3)

- **Foloseste adrese MAC de 6 bytes**
- **Dezavantaj**: A fost proiectat ca un protocol de tip broadcast, ceea ce ii limiteaza performanta.
- Nu are o negociere a parametrilor de comunicatie.

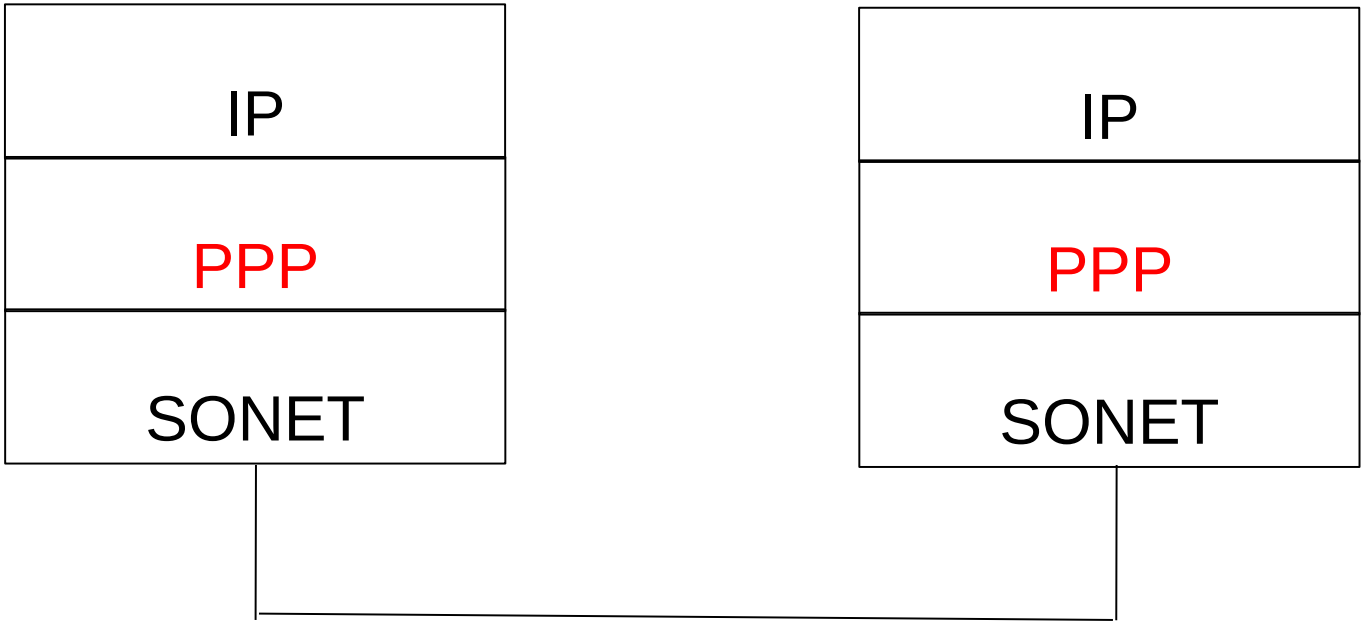


Caracteristici Ethernet

- Protocolul functioneaza cel mai bine pentru incarcare redusa (o incarcare de 30% este considerata foarte mult)
- Este un protocol simplu, usor de administrat.
- Foloseste algoritmul CDMA/CD pentru managementul coliziunilor
- Tipuri de Ethernet
  - Fast Ethernet 100 Mbps
  - Gigabit Ethernet 1Gbps

Legatura de date in Internet

Comunicarea pe fibra optica



Fibra optica

PPP – Point to Point Protocol

Ofera incadrare
Link Control Protocol, LCP
Network Control Protocol, NCP

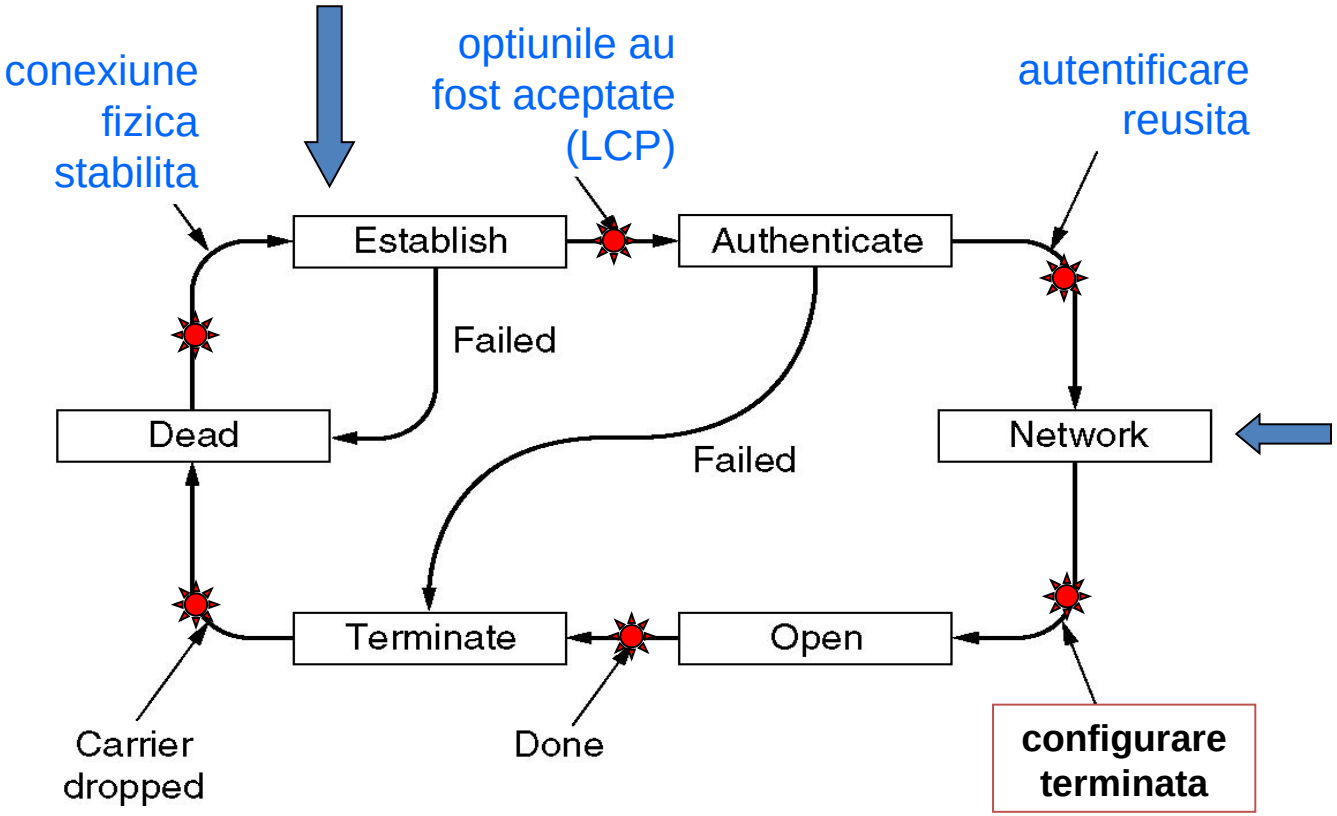
Bytes	1	1	1	1 or 2	Variable	2 or 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Payload	Checksum	Flag 01111110

Format de cadru PPP pentru modul nenumerotat

Addresa 11111111 = toate statiile accepta cadrul
Control 00000011 = nenumerotat
Lungimea default a datelor este 1500 bytes
Protocol = selecteaza dintre
- LCP, NCP
- IP, IPX (Internetwork Packet eXchange), OSI CLNP, XNS (Xerox Network Services)

PPP – Point to Point Protocol (2)

Tipuri de cadre LCP
Configure-request **Configure-reject**
Configure-ack **Code-reject**
Configure-nak **Protocol-reject**



protocolul de retea este configurat folosind NCP (ex. ptr IP – alocare adresa IP)

Inchide conexiune
Terminate-request
Terminate-ack

PPP transporta pachete (de ex. IP)

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**

Point To Point Protocol over Ethernet (PPPoE)

- Publicat in RFC 2516 (1999)

- Extinde capabilitatile protocolului PPP pentru a putea crea o conexiune punct-la-punct peste o retea Ethernet

- Este folosit de catre ISP (Internet Service Provider) pentru a gestiona accesul la retea pentru clienti. Companiile care ofera servicii de Internet pot folosi aceleasi mecanisme de autentificare pentru sesiunile PPPoE si PPP.

- Protocolul este configurat ca o legatura de date punct-la-punct intre doua porturi Ethernet.

Point To Point Protocol over Ethernet (PPPoE)

Pasii protocolului:

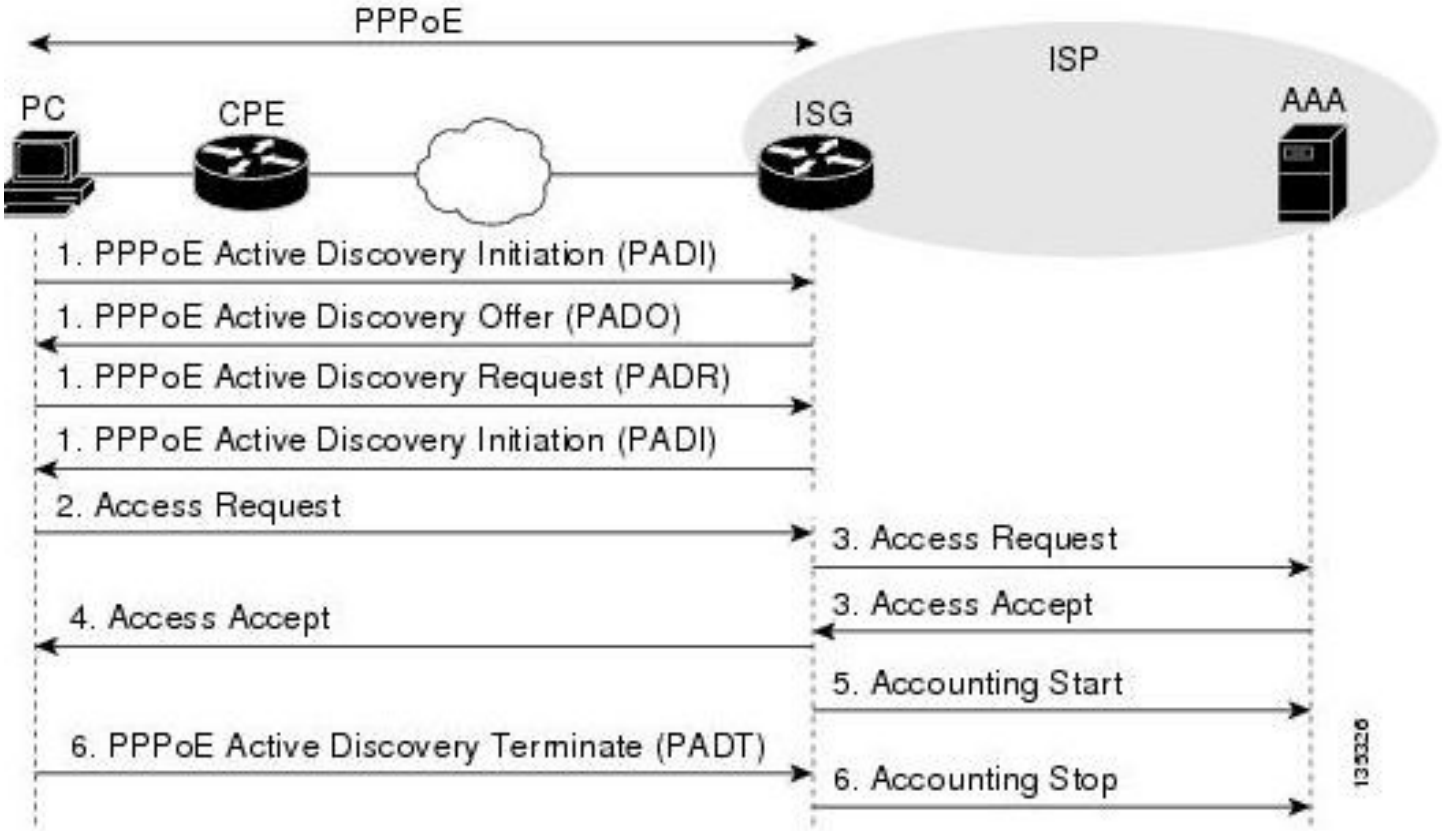
1) Initiation – Clientul trimite un pachet **PPPoE Active Discovery Initiation (PADI)** pentru a initia sesiunea.

2) Offer – Serverul raspunde cu un pachet **PPPoE Active Discovery Offer (PADO).**

3) Request – Dupa primirea unui pachet PADO, clientul raspunde prin trimiterea unui pachet **PPPoE Active Discovery Request (PADR)** catre server.

4) Confirmation – Dupa primirea unui pachet PADR, serverul raspunde prin generarea unui ID unic pentru sesiunea PPP si trimite un pachet **PPPoE Active Discovery Session (PADS)** de confirmare catre client.

Point To Point Protocol over Ethernet (PPPoE)



Point To Point Protocol over Ethernet (PPPoE)

Ethernet packet encapsulation format

Destination_address (48bits)	Source_address (48bit)	Ether_type (16bit)	PPPoE Packet	Checksum (16bits)

PPPoE packet encapsulation format

Ver (0x01)	Type (0x01)	Code (8bits)	Session_ID (16bits)	Length (16bits)	PPP Packet

PPP packet encapsulation format

Flag 01111110	Address 11111111	Control 00000011	Protocol 8/16bits	Information	FCS 16 bits	Flag 01111110

Point To Point Protocol over Ethernet (PPPoE)

Field	Description
Destination_address	Indicates either a unicast Ethernet destination address or the Ethernet broadcast address (0xffffffffffff). For Discovery packets, the value is either a unicast or broadcast address. A PPPoE client uses a broadcast address to search for a PPPoE server and uses a unicast address after choosing a PPPoE server. For PPP session traffic, this field must contain the peer's unicast address as determined from the Discovery stage.
Source_address	Indicates the Ethernet MAC address of a source device.
Ether_type	Set to either 0x8863 (Discovery stage) or 0x8864 (Session stage).
Ver	Indicates a PPPoE version number. The **Ver** field is four bits and must be set to 0x1.
Type	Indicates a PPPoE type. The **Type** field is four bits and must be set to 0x1.
Code	Indicates a PPPoE packet type. The **Code** field is eight bits and has different values: • 0x00: session data • 0x09: PADI packet • 0x07: PADO or PADT packet • 0x19: PADR packet • 0x65: PADS packet
Session_ID	The **SESSION_ID** field is sixteen bits. The value is fixed for a given PPP session and defines a PPP session along with Ethernet source and destination addresses. A value of 0xffff is reserved for future use and must not be used.
Length	Indicates the length of the PPPoE payload. The **Length** field is 16 bits, excluding the length of the Ethernet or PPPoE header.
PPP Packet	For PPP packet format, see PPP Basic Concepts-PPP Packet Format.

Studiu individual

A. S. Tanenbaum Reţele de calculatoare, ed 4-a, BYBLOS 2003
3.1 ASPECTE ALE PROIECTĂRII NIVELULUI LEGĂTURĂ DE DATE
3.2.2 Coduri detectoare de erori
3.3 PROTOCOALE ELEMENTARE PENTRU LEGĂTURA DE DATE
3.4 PROTOCOALE CU FEREASTRĂ GLISANTĂ
3.6 EXEMPLE DE PROTOCOALE ALE LEGĂTURII DE DATE
A. S. Tanenbaum Computer networks, 5-th ed. PEARSON 2011
3.1 DATA LINK LAYER DESIGN ISSUES
3.2.2 Error-Detecting Codes
3.3 ELEMENTARY DATA LINK PROTOCOLS
3.4 SLIDING WINDOW PROTOCOLS
3.5.1 Packet over SONET

Nivelul retea

Cuprins

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

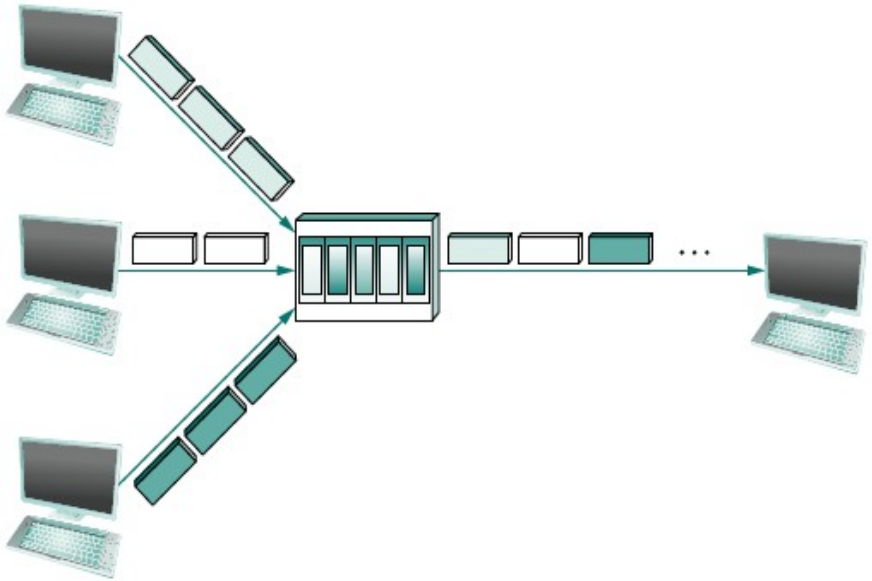
De ce este nevoie de pachete?

Legaturile unei retele pot fi folosite simultan de transmisii paralele intre mai multe perechi de noduri

Multiplexare – se aloca transmisiilor

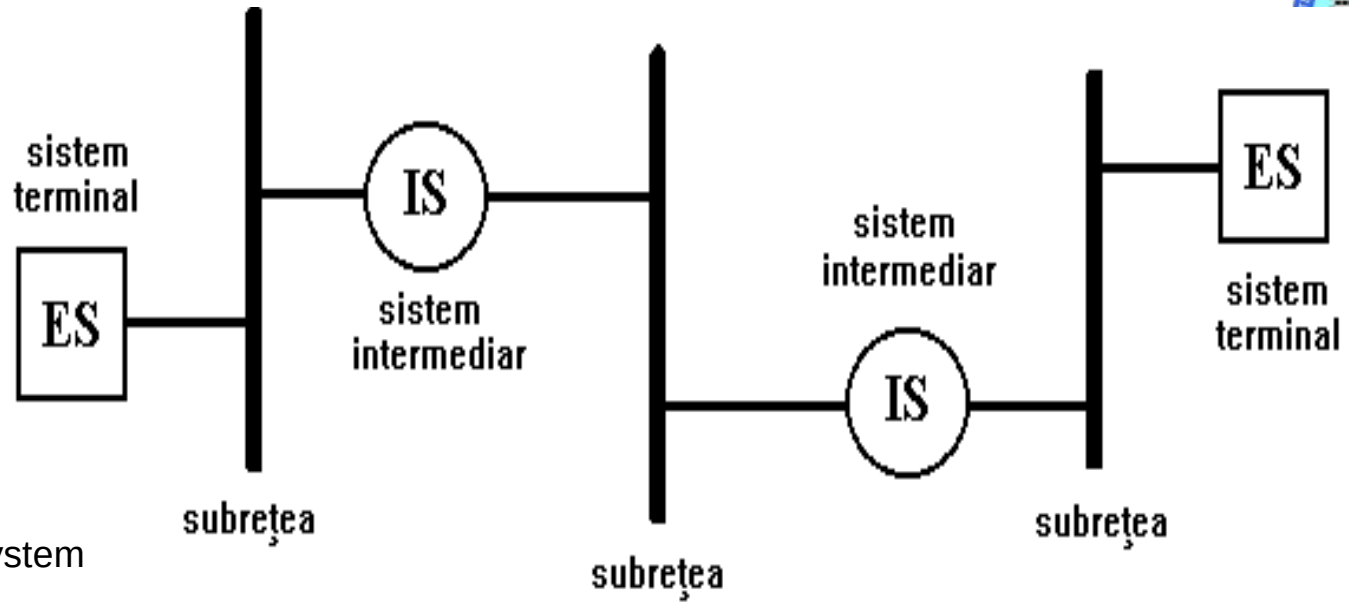
- sloturi de timp - STDM – Synchronous Time Division Multiplexing
- subcanale de frecvente diferite – FDM – Frequency Division Multiplexing

- multiplexare statistica – STMD – sloturile sunt alocate la cerere; daca o singura pereche de noduri are de transmis, nu asteapta ptr. slot;
- pentru a evita acapararea legaturii, transmisia se face in pachete cu dimensiune limitata

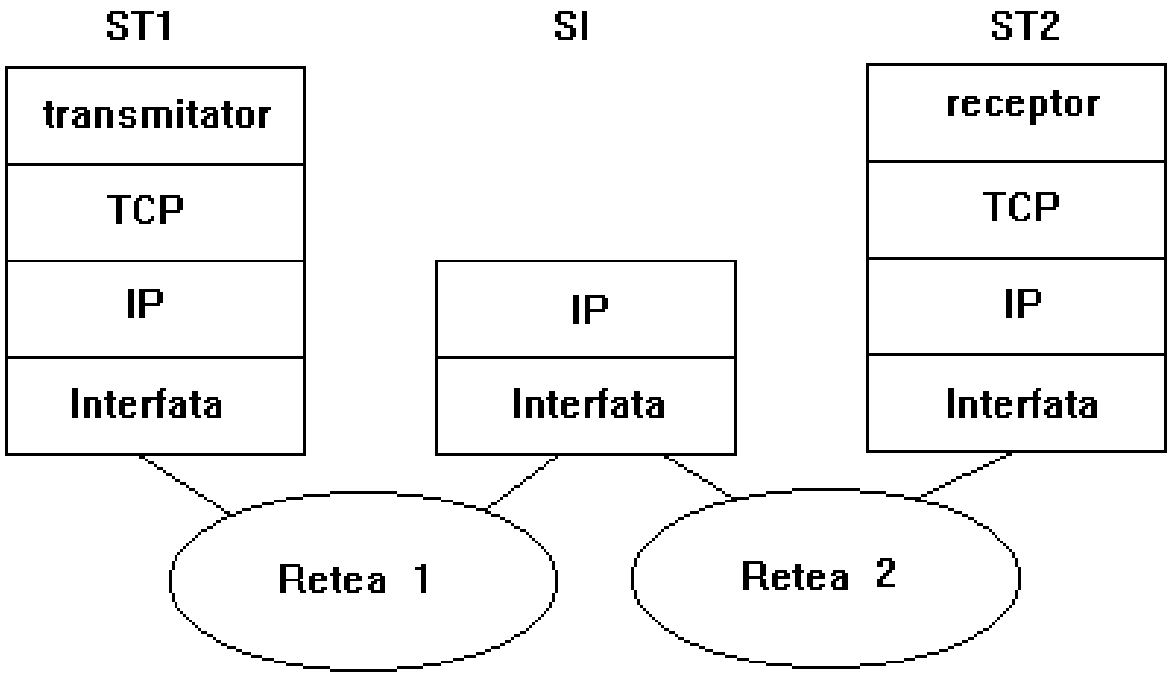


Modelul Internetului

- ES – End system
- IS – Intermediate system



Nivelele străbătute de pachete



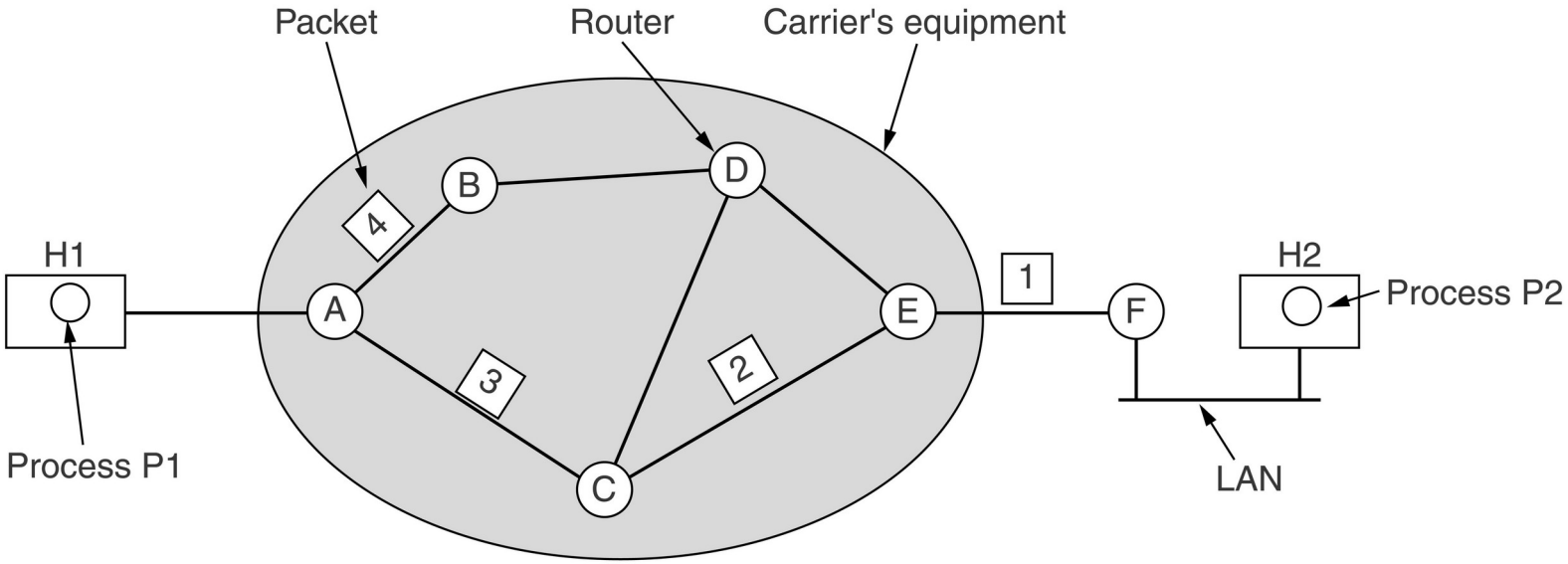
Funcţiile nivelului reţea

- 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



A's table

A	A
B	A
C	–
D	D
E	E
F	E

C's table

A	A
B	A
C	–
D	D
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

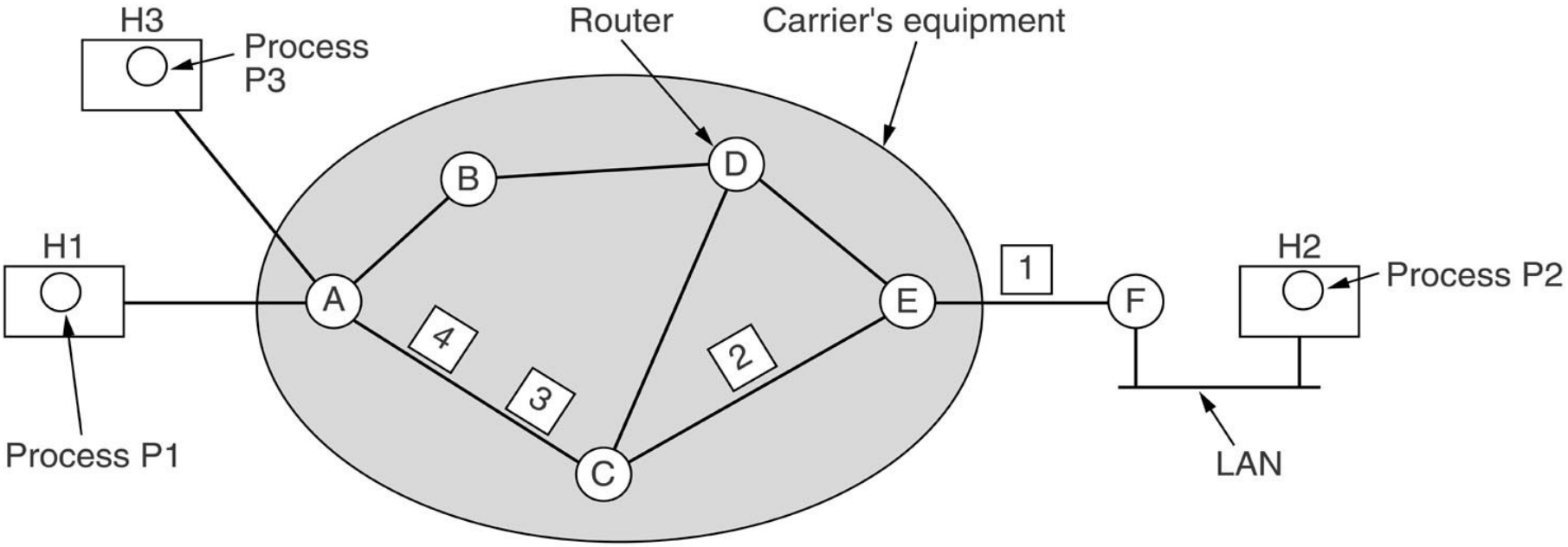
Folosita de pachetele 1, 2 si 3

Folosita de pachetul 4

Caracteristici datagrame

- Pachetul contine toate informatiile necesare ruterelor pentru “pasarea” lui catre destinatie (inclusiv adresele sursa si destinatie)
- Organizare fara conexiune
  - Un calculator gazda (*host*) poate trimite pachetul oricand si oriunde in retea
- Nu asigura corectitudinea
  - Transmitatorul nu poate sti daca pachetul este livrat sau daca destinatarul mai este conectat
- Nu pastreaza ordinea pachetelor
  - Pachetele sunt dirijate independent unele de altele
- Robusta
  - La defectarea unei legaturi se gasesc rute alternative
- Utilizare larga in Internet

Organizarea internă – circuit virtual



A's table

H1	1
C	1

C's table

A	1
E	1

E's table

C	1
F	1

In Out

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 (buffer 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 oricarui 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” sa livreze pachetele la destinație
  - nu face nici o încercare să corecteze erorile

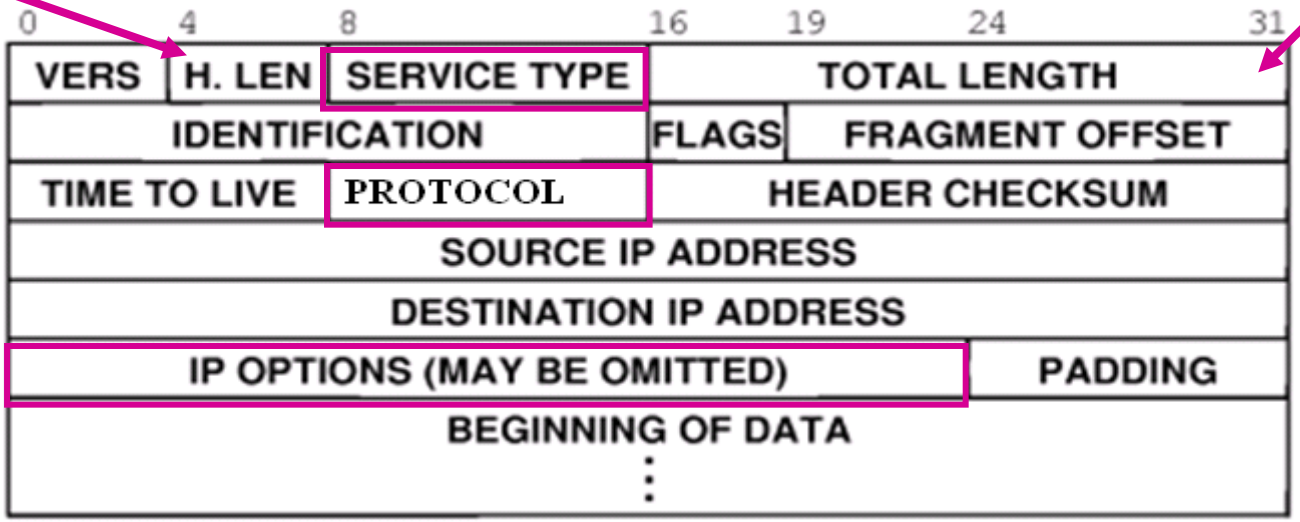
Protocol IPv4 – formatul pachetului

VERS = IPv4, IPv6….
SERVICE TYPE = precedence (3), delay, throughput, reliability, cost
PROTOCOL = (TCP, UDP, etc.)

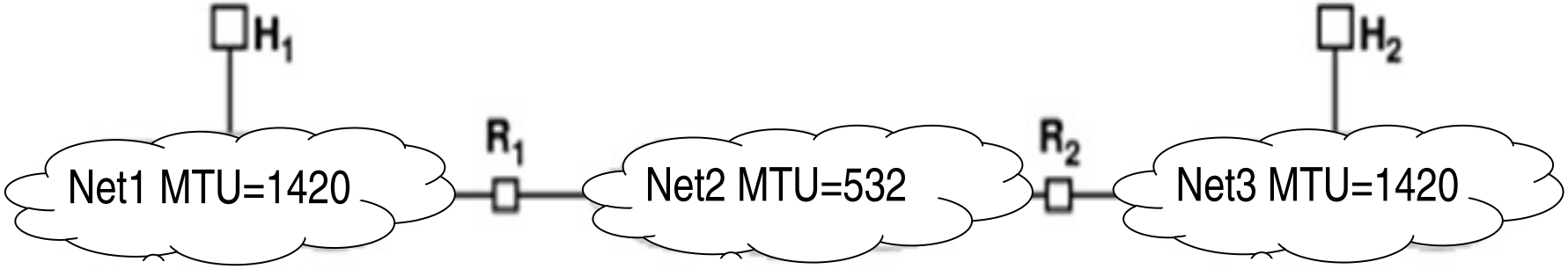
OPTIONS:
- Security
- Strict source routing
- Loose source routing
- Record route
- Timestamp

numar cuvinte de 32 biti

max 65535 octeti

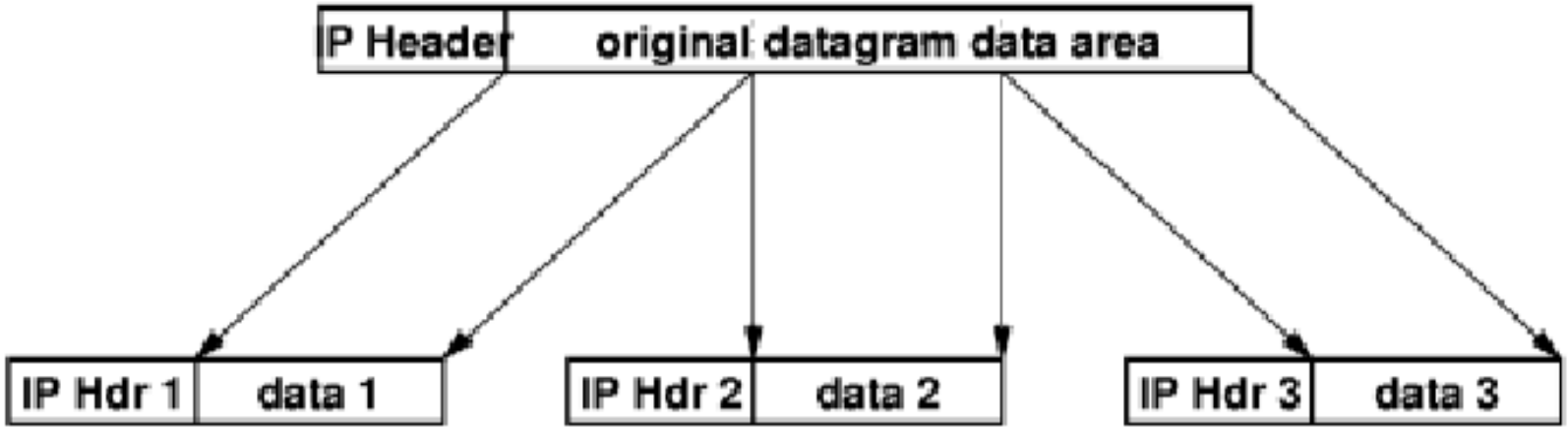


MTU – Maximum Transmission Unit



MTU – Maximum transmission unit

Fragmentarea se face la R_1



Reasamblarea se face la H_2 – de ce?

Protocol IPv4 – formatul pachetului

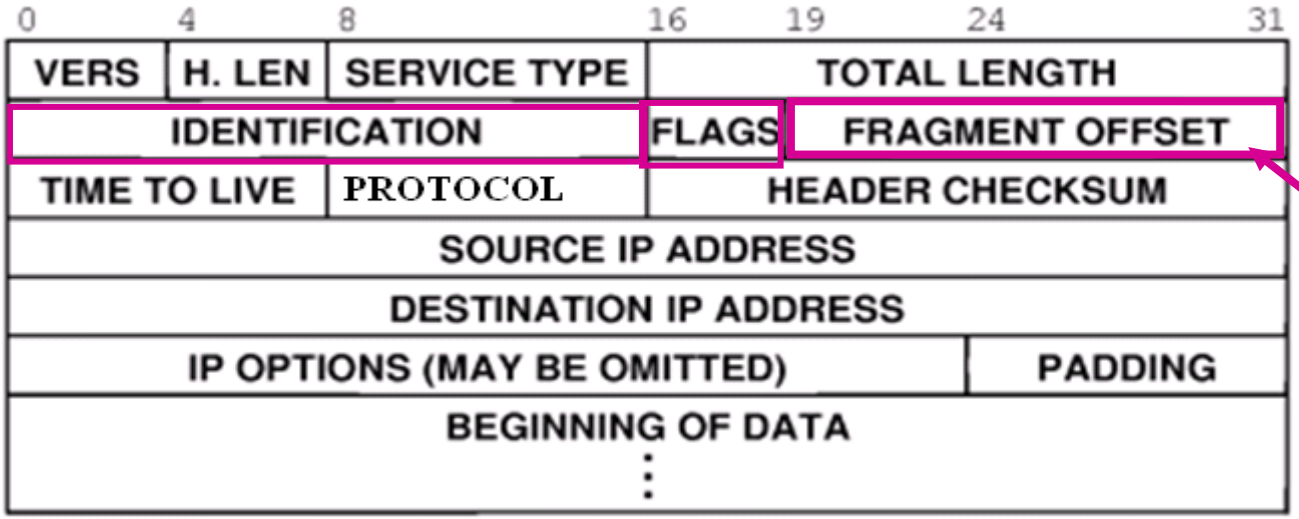
IDENTIFICATION Id datagrama de care aparţine fragmentul

FRAGMENT OFFSET pozitia fragmentului in pachet

FLAGS

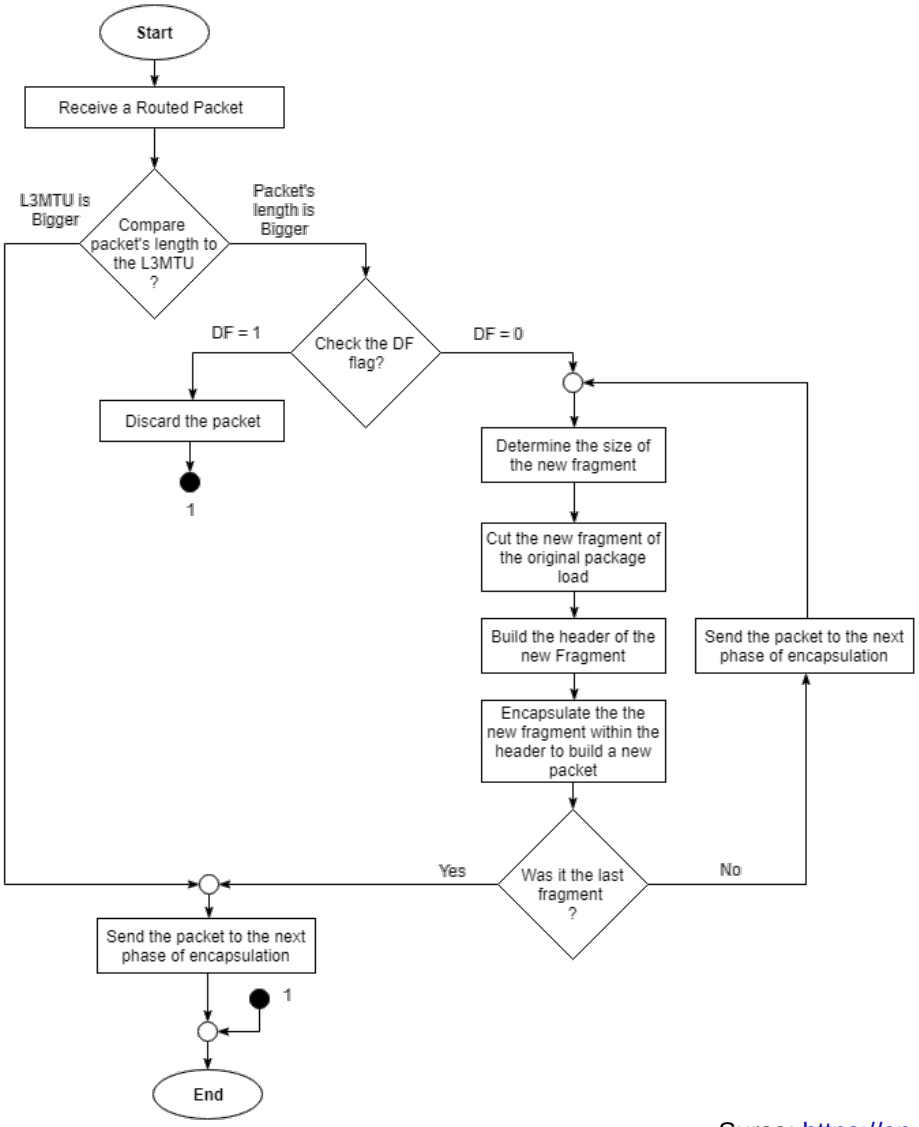
DF = Don't Fragment

MF = More Fragments



max 8192 fragmente a 8 octeti

Protocol IPv4 – fragmentarea



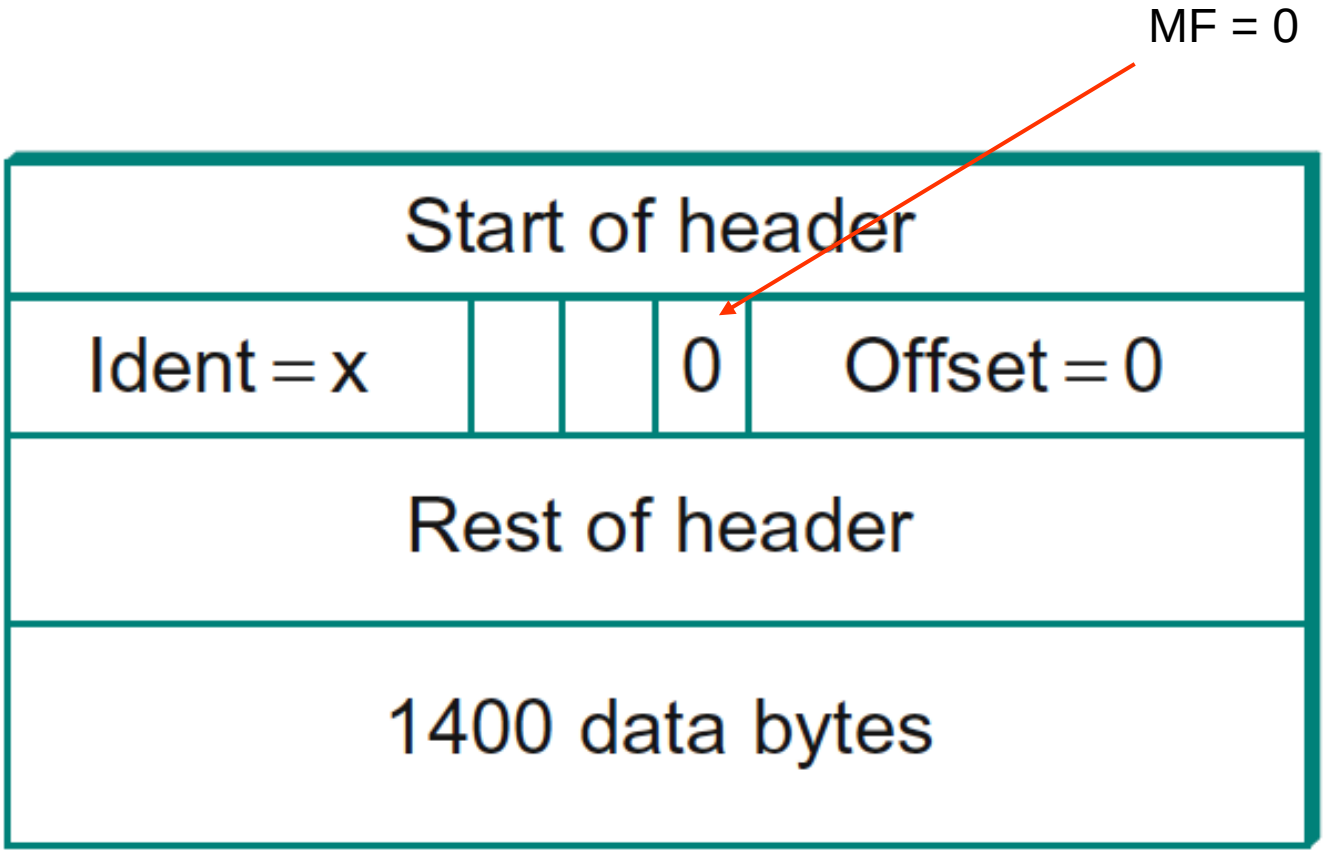
Sursa: https://en.wikipedia.org/wiki/IP_fragmentation

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 este restul pana la 532 octeti?

b2

Start of header				
Ident = x			1	Offset = 64
Rest of header				
512 data bytes				

b1

Start of header				
Ident = x			1	Offset = 0
Rest of header				
512 data bytes				

b3

Start of header				
Ident = x			0	Offset = 128
Rest of header				
376 data bytes				

ARP

Conversie adresa IP – adresa fizică

La livrarea directă (destinatar in aceeasi retea) se foloseste adresa fizică a receptorului (ptr care se cunoaște adresa IP)
Nu exista o asociere directa între adresa fizica si adresa IP

De ex.

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

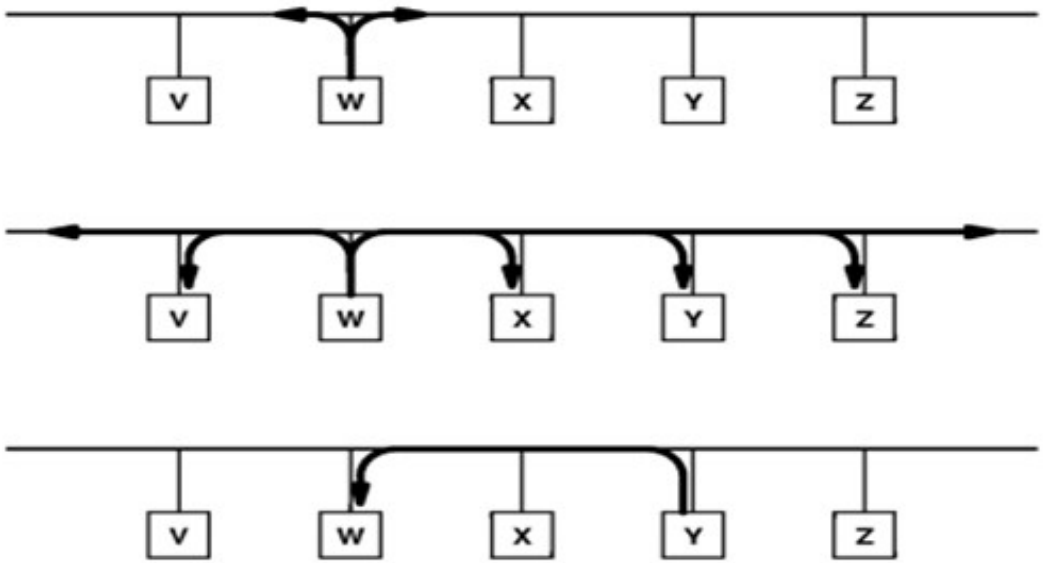
Pentru mapare se pot folosi

- tabele de corespondenţă
- formule de calcul
- schimb de mesaje
  - ex. ARP - Address Resolution Protocol
    - Face maparea intre adresa de protocol şi adresa hardware

ARP - Address Resolution Protocol

Secvența de mesaje pentru a afla adresa fizică

- w difuzeaza o cerere ARP cu adresa IP cunoscută
- cererea este primită de toate gazdele din reteaua locala
- y – care are adresa IP respectivă – trimite ca raspuns adresa fizică (ex. Ethernet)



Conversie adresa IP – adresa fizică

0 – 8	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

ICMP

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

Test accesibilitate (**ping** trimite **ICMP** Echo şi aşteaptă un timp răspunsul)

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

ICMP- Internet Control Message Protocol

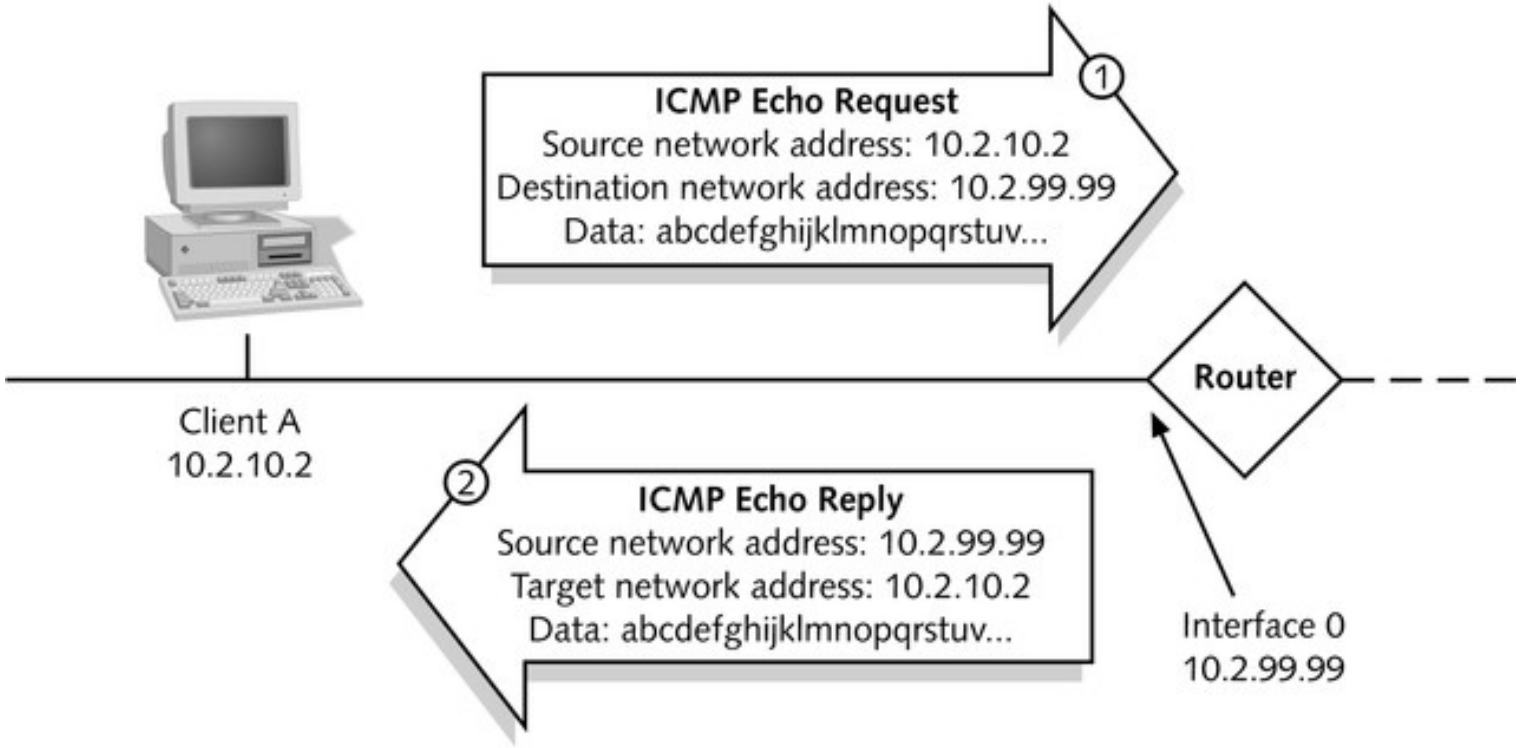


Figure 4-1 PING utility uses ICMP echo requests and replies

Functia PING foloseste mesaje de tip **ICMP echo request** si **ICMP Echo Reply**

Folosire ICMP pentru aflare path MTU

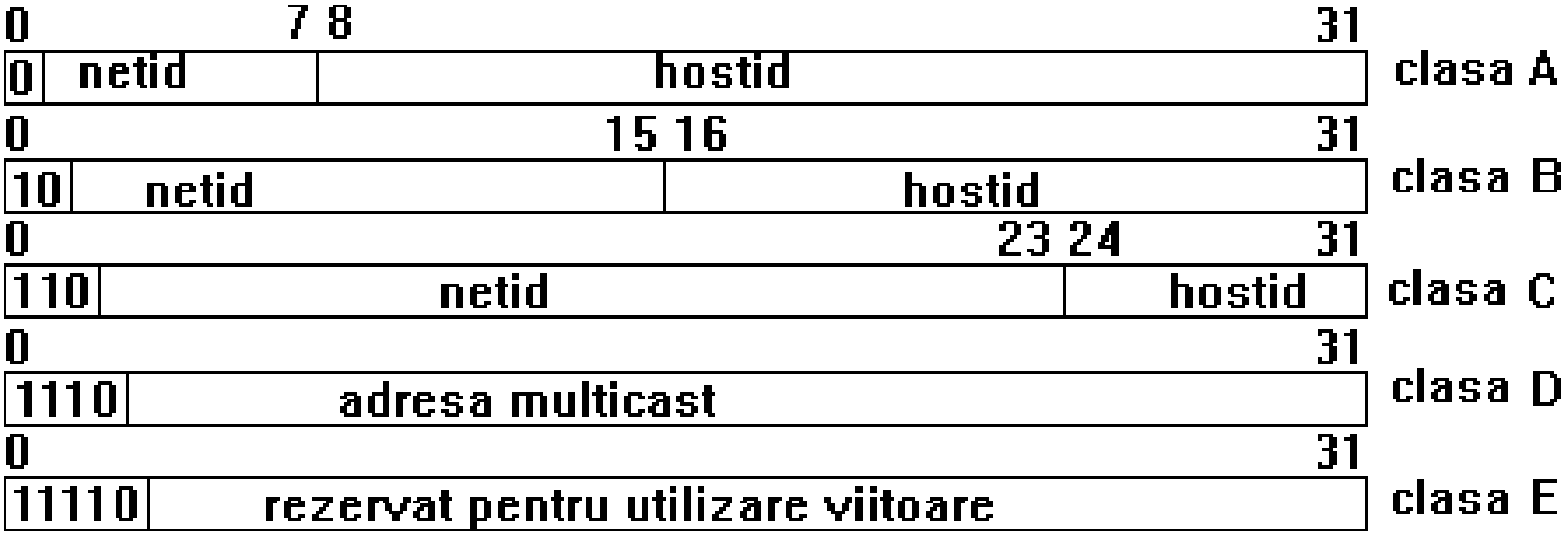
Path MTU = Maximum Transmission Unit minimă pentru o cale

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

Adrese IP

Clase de adrese IP

Fiecare nod (gazda/ruter) are o adresa IP asociata cu interfata lui de retea
Un nod cu mai multe interfete are mai multe adrese IP (ex. rutere)



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

Câteva adrese speciale

Prefix (rețea)	Suffix (gazdă)	Semnificație	Scop
toţi 0	toţi 0	acest calculator	Folosită cand nodul inca nu are o adresa
network	toţi 0	network	Identifică reţeaua
network	toţi 1	broadcast	broadcast în reţeaua specificată
toţi 1	toţi 1	broadcast	broadcast în reţeaua locală
127	orice	loopback	testare

Notații 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 in aceeași rețea)
  - Toate nodurile care au aceeași adresa_retea sunt situate in aceeasi retea fizica 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> destinatie din datagrama
Caută o intrare cu adresa_retea în tabela de dirijare

if *adresa_retea apare in tabela de dirijare*
  if *adresa_retea indica o retea direct conectata* ***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 numarul de intrari in tabela de dirijare (o intrare pentru o adresa_retea)
In practica, tabelele de rutare sunt separate pe clase de adrese
  - Căutare prin: indexare (A şi B) sau hashing (C)

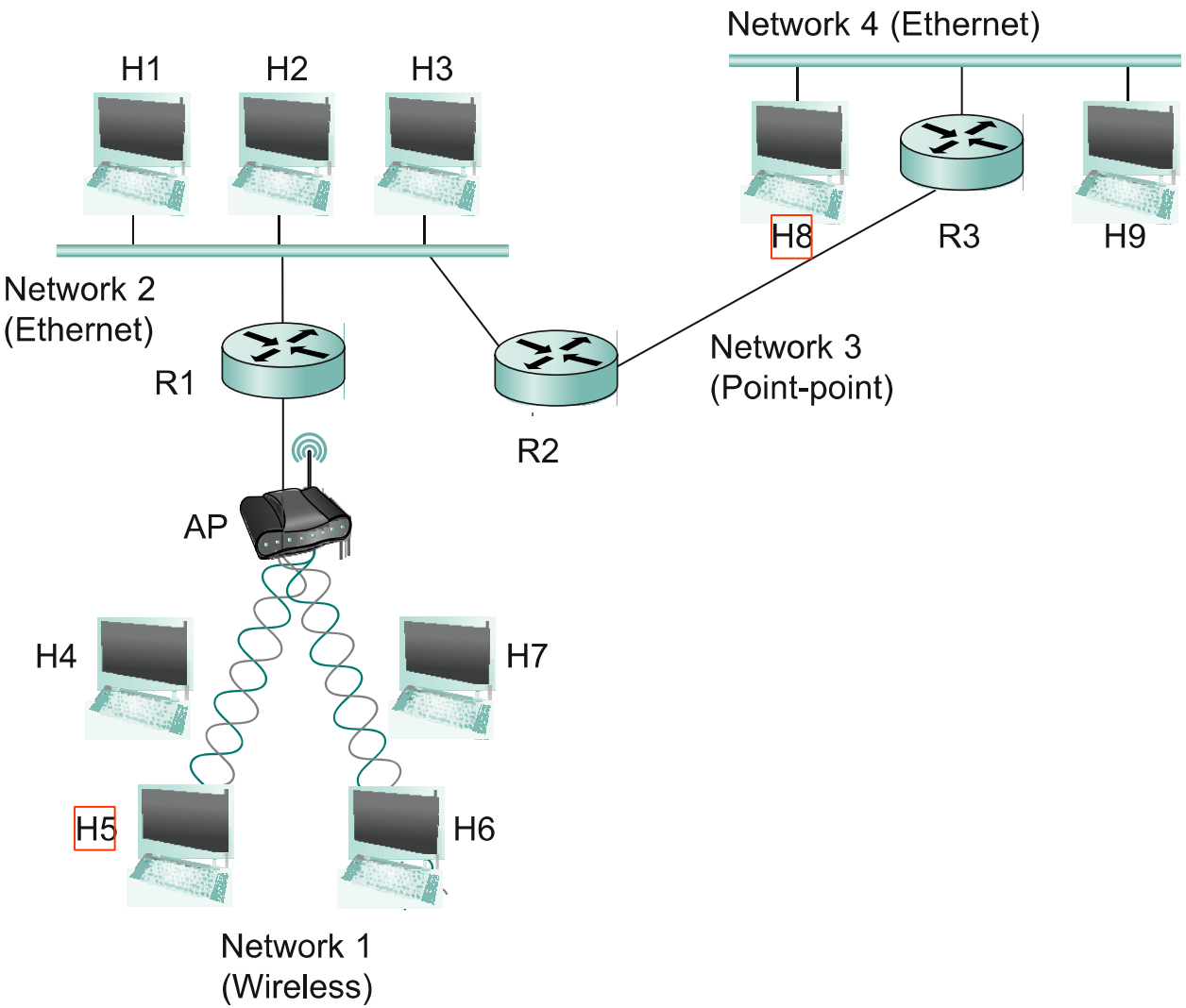
Exemplu

Transferul unei datagrame intre H5 si H8

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

Pachetul trece prin ruterele R1, R2 si R3

R3 este in aceeasi retea cu H8 si ii livrează direct datagrama



Subreţele

- Regula “o adresă pentru fiecare rețea fizică separată”
  - foloseste ineficient spatiul de adrese
    - o retea de clasa C cu 2 noduri consuma 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 daca le foloseste pe toate sau nu
- Solutia 1 – subretele
  - de ex. o retea clasa B este impartita in mai multe subretele apropiate geografic
- Solutia 2 - adrese fara clase

CIDR – Classless InterDomain Routing

Ideea: alocă spaţiul de adrese IP în blocuri de lungimi diferite

Notaţia specială pentru adresa de rețea CIDR

194.24.0.0/21 -> din cei 32 de biţi ai adresei IP

adresa_retea ocupă 21 biţi

adresa_gazda ocupă 11 biti

University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7. **255**	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. **255**	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. **255**	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 sa fie o putere a lui 2

⇒ toate adresele din bloc au aceeasi adresa_retea ⇒ adresa de inceput a blocului de adrese trebuie sa fie multiplu de dimensiunea acestuia

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

0 **2048** **3072** **4096**

Cambridge	**Edin.**		**Oxford**

Algoritm *forwarding*

Intrare in tabela de rutare - (adresa_retea, Masca, NextHop)

Algoritmul alege intrarea pentru care

(Adresa_IP AND Masca) = adresa_retea

Ex. Sosește pachet cu adresa_IP = 194.24.17.4

compara cu Cambridge /21 – adresa_rețea = 194.24.0.0

adresa_rețea: **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” (“Longest prefix match”)

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 in tabela de dirijare același NextHop

Tabela de dirijare poate include o singură intrare comuna pentru

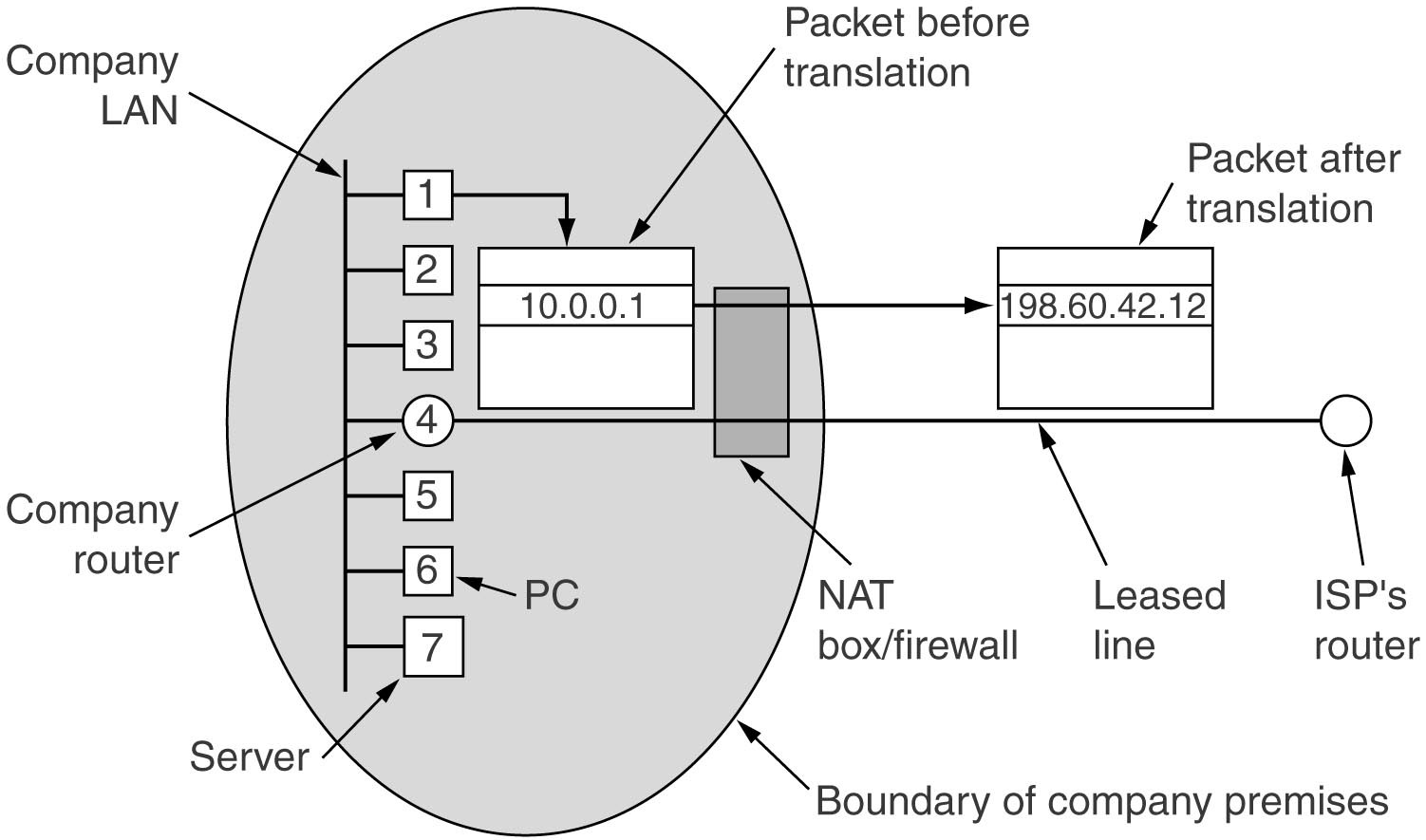
adresa_retea **11000010 00011000 00000000 00000000**

adica 194.24.0.0/19

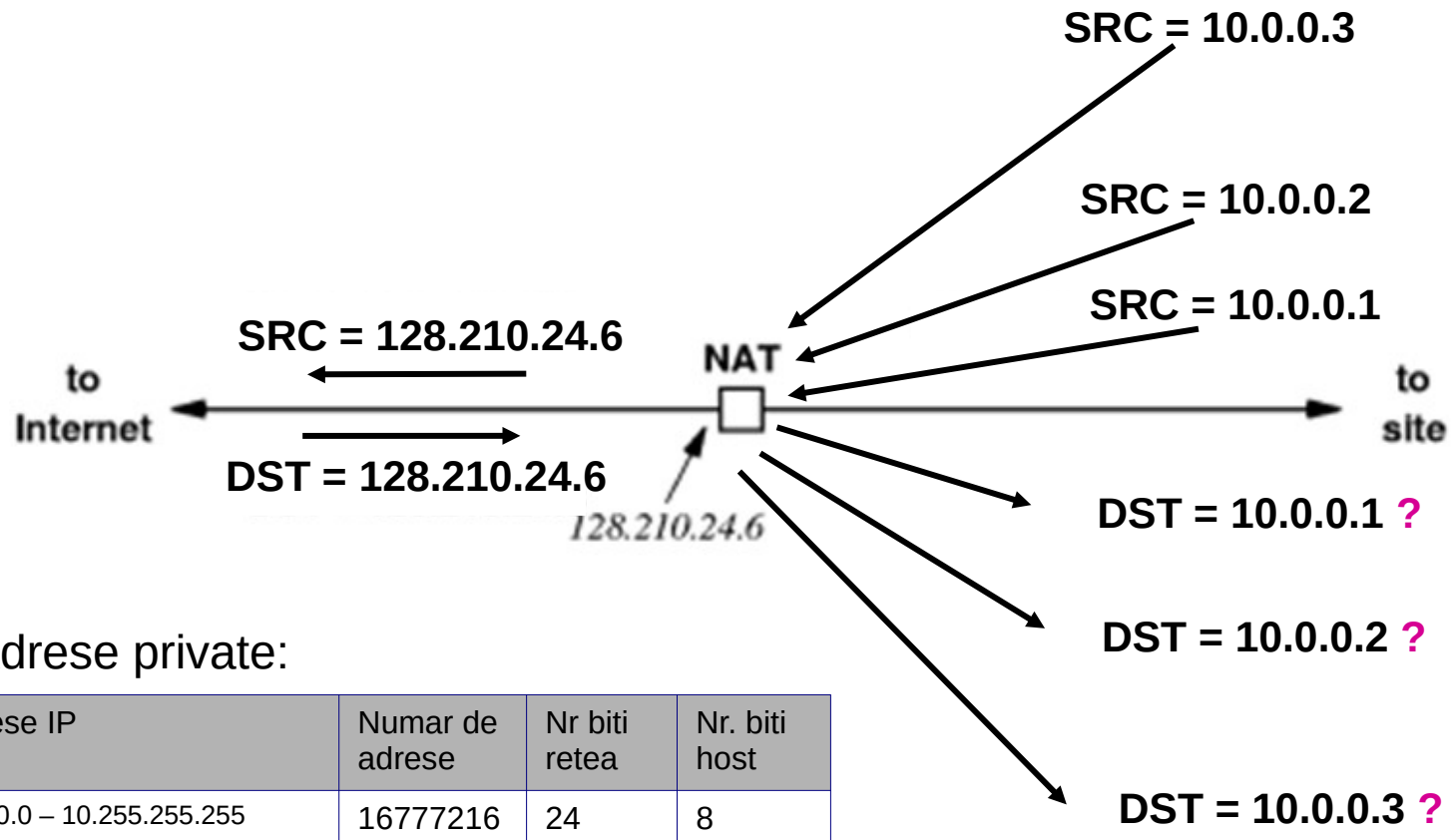
NAT

NAT – Network Address Translation

O adresă este asignata pentru mai multe calculatoare
Foloseşte adrese locale (private sau non-rutabile)
NAT translatează între adresa privată şi o adresă globală



Translatarea adresa globală ⇒ adresa privată



Adrese private:

Adrese IP	Numar de adrese	Nr biti retea	Nr. biti host
10.0.0.0 – 10.255.255.255	16777216	24	8
172.16.0.0 – 172.31.255.255	1048576	20	12
192.168.0.0 – 192.168.255.255	65536	16	16

Principiul NAT

Foloseşte

adresa IP + număr port transmitator

tabela de translatare

Transmisie

înlocuieşte adresa IP locală cu o adresă IP globală

memorează (in tabela de translatare) corespondenţa şi număr port

inlocuieşte număr port cu index în tabela translatare

re-compune sumele de control IP şi TCP

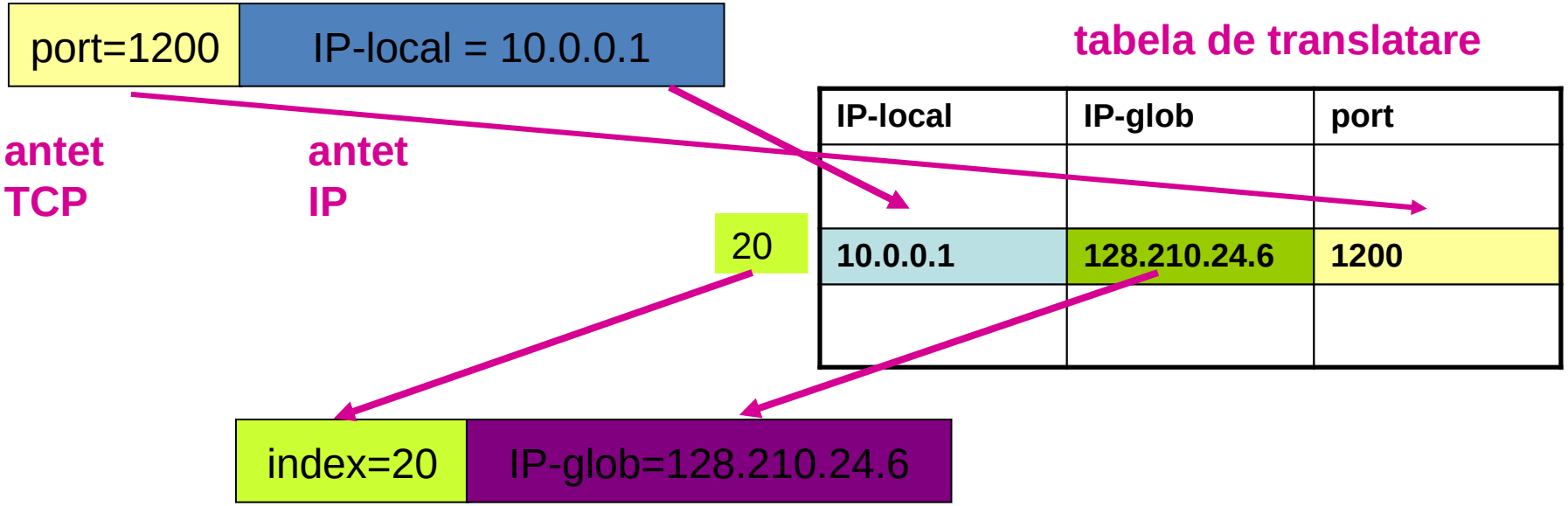
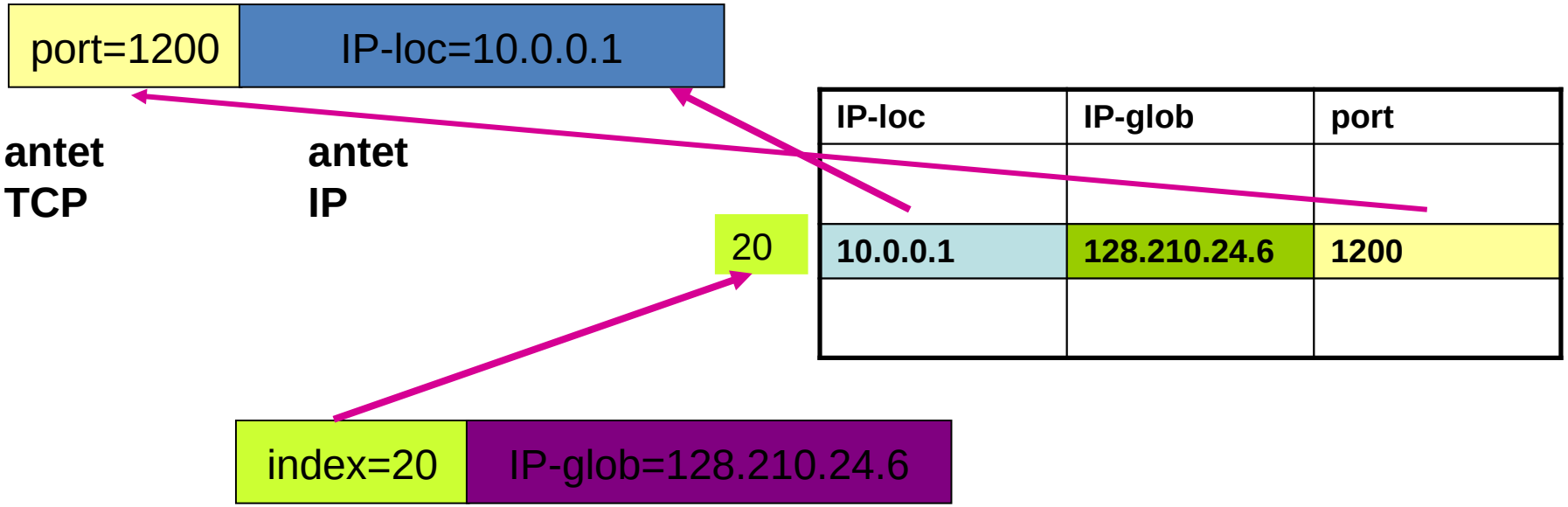


tabela de translatare

IP-local	IP-glob	port
10.0.0.1	128.210.24.6	1200

Receptie

obţine număr port din pachet (= index în tabela translatare)
extrage adresa IP locală şi număr port
înlocuieşte adresa IP şi număr port din pachet
re-calculează sumele de control IP şi TCP



IPv6

IPv6 - Motivaţii

Spaţiul de adrese

32 biţi = peste un milion de reţele

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

Tip servicii

Aplicaţii diferite au cerinţe diferite de livrare, siguranţă şi viteză

IPv4 are tip de serviciu dar adesea nu este implementat

IPv6 - format Base header

lungime fixă = 40 octeti

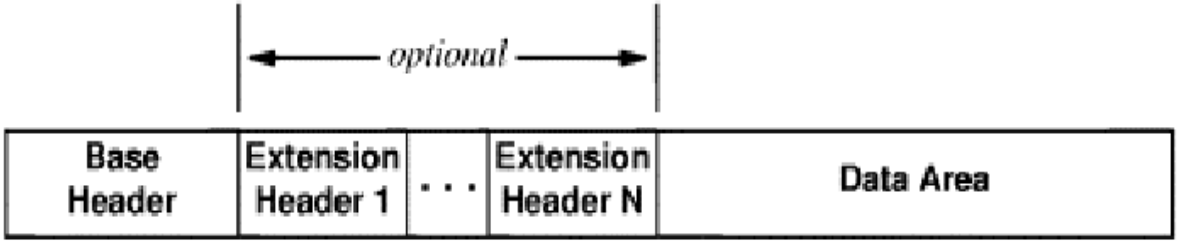
Priority - clasa de trafic

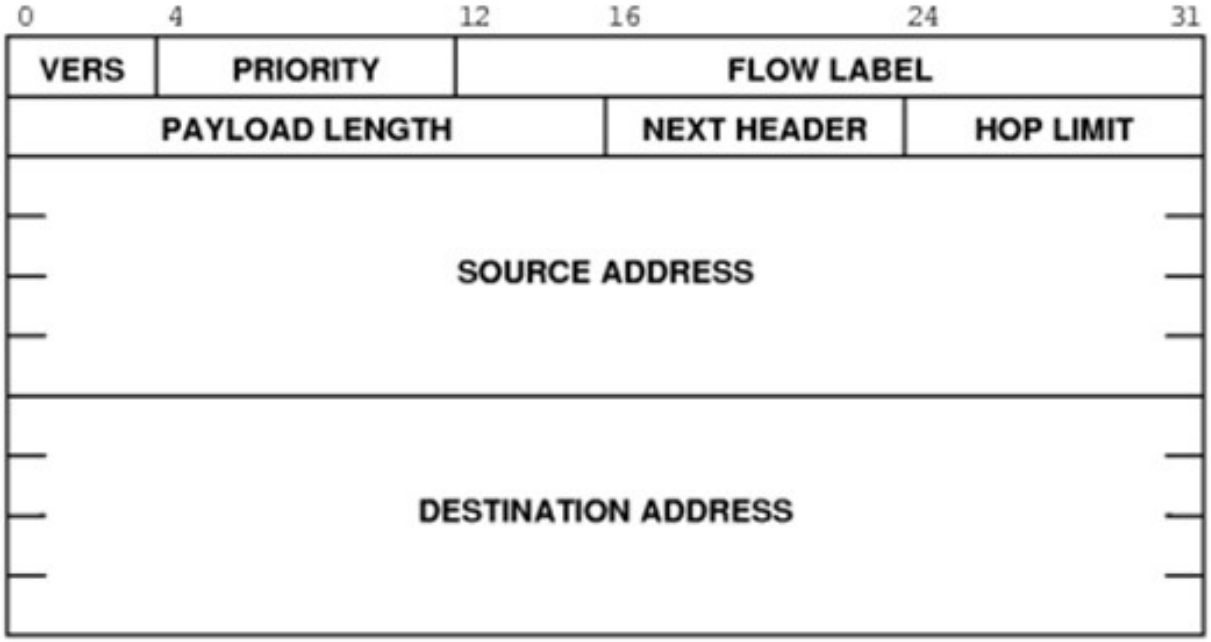
FLOW LABEL - asociază datagramele unui flux

Diferenţe circuit virtual

două fluxuri cu aceeaşi etichetă se diferentiaza prin adr sursă + adr dest

aceeaşi pereche sursă+dest poate avea mai multe fluxuri





Conţine mai puţine info decât antet IPv4

Restul de info în extensii

NEXT HEADER defineşte tipul datelor (ex. TCP)

NEXT HEADER defineşte 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 destinaţie nefolosit

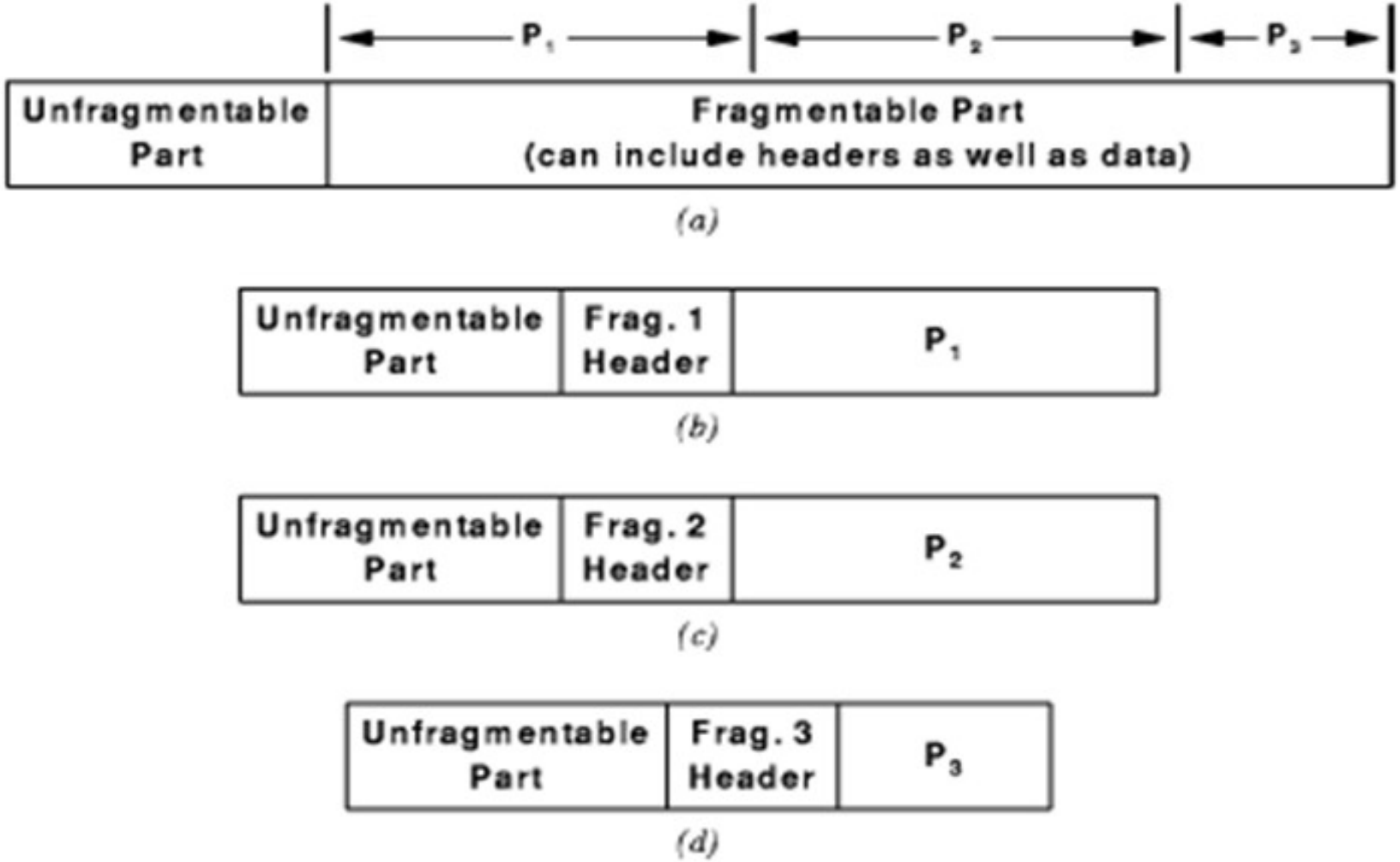
Routing – lista rutere de vizitat

Fragmentation – identificare fragmente

Authentication – verificare identitate transmiţător

Encrypted security payload – info despre conţinut 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

Eficienţa – antet nu are spaţiu pierdut

Flexibilitate – noi antete pentru noi caracteristici

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

Adrese de 128 biti

Includ prefix reţea şi suffix gazdă

Fără clase de adresă – limita prefix/suffix oriunde

Tipuri speciale de adrese:

- unicast
- multicast
- cluster – colecţie de calculatoare cu acelaşi prefix; datagrama livrată unuia din ele (permite duplicare servicii)

Notaţia adresei

- 16 numere
  105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255

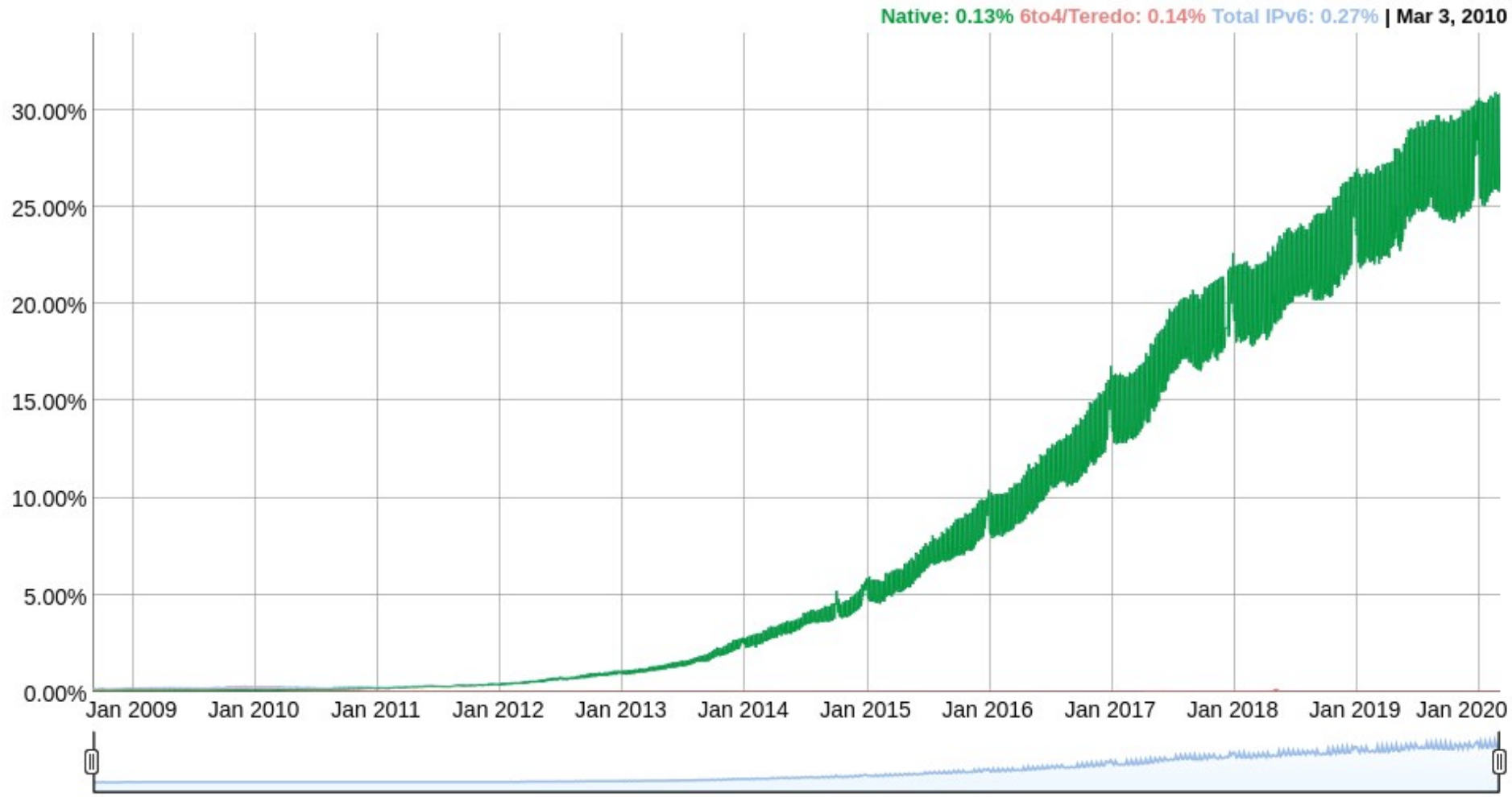
- Notaţie hexazecimală
- 69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF

- Compresie zerouri
- FF0C:0:0:0:0:0:0:B1
- FF0C::B1
-
- Adrese IPv6 cu 96 zerouri prefix sunt interpretate ca adrese IPv4

IPv6 adoption rate

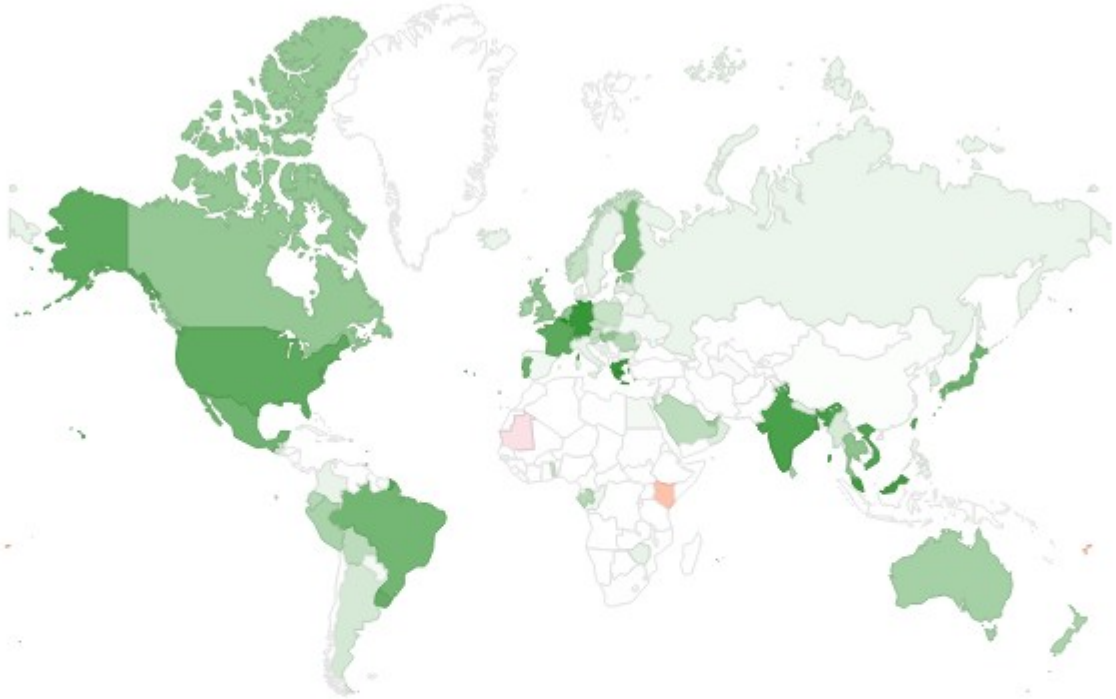
IPv6 Adoption

We are continuously measuring the availability of IPv6 connectivity among Google users. The graph shows the percentage of users that access Google over IPv6.



IPv6 adoption rate

Per-Country IPv6 adoption



World | Africa | Asia | Europe | Oceania | North America | Central America | Caribbean | South America

The chart above shows the availability of IPv6 connectivity around the world.

- Regions where IPv6 is more widely deployed (the darker the green, the greater the deployment) and users experience infrequent issues connecting to IPv6-enabled websites.
- Regions where IPv6 is more widely deployed but users still experience significant reliability or latency issues connecting to IPv6-enabled websites.
- Regions where IPv6 is not widely deployed and users experience significant reliability or latency issues connecting to IPv6-enabled websites.

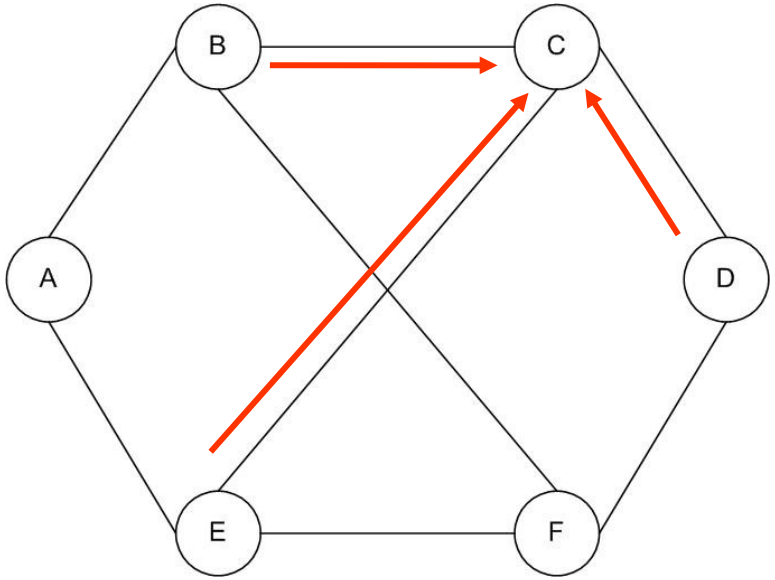
Dirijarea

Dirijarea - clasificare

- Fără tabele de dirijare
  - inundarea
  - hot potato
- Cu tabele de dirijare – criterii diverse
  - adaptarea la condiţiile 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
  - **informaţii schimbate între noduri**
    - **starea legăturii**
    - **vectorul distanţelor**
  - tipul reţelei
    - uniformă
    - ierarhică

Vectorii distanţelor (Distance vector)

Algoritm distribuit !
Fiecare nod trimite periodic vecinilor sai o lista cu distantele de la el la celelalte noduri.



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

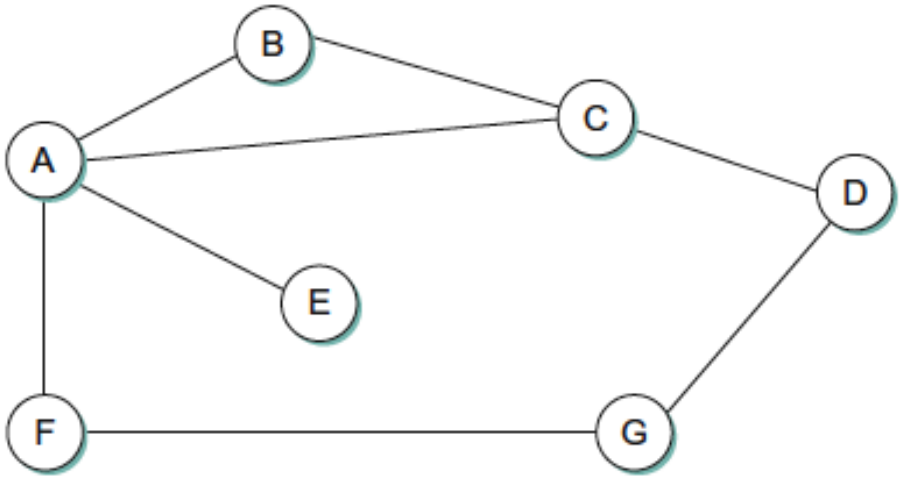
De la B: (5, 0, 8, 12, 6, 2);
De la D: (16, 12, 6, 0, 9, 10);
De la E: (7, 6, 3, 9, 0, 4).

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

In plus, intârzierea măsurată de la C la B, D si E este 6, 3 şi 5 respectiv.

Costurile de la C Prin -> La	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)
La E:	1	2	2	3	0	2	3	distante initiale la E
	∞	**2**	**2**	3	0	2	3	legatura A – E cade

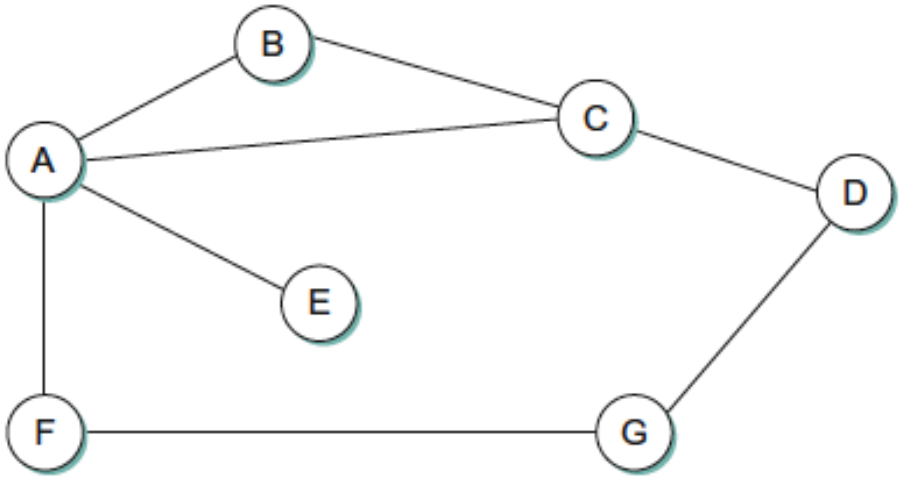
distantele catre E anuntate de: **A = ∞, B = 2, C = 2**

In functie de ordinea evenimentelor, se pot face modificarile:

∞	**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

distantele 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

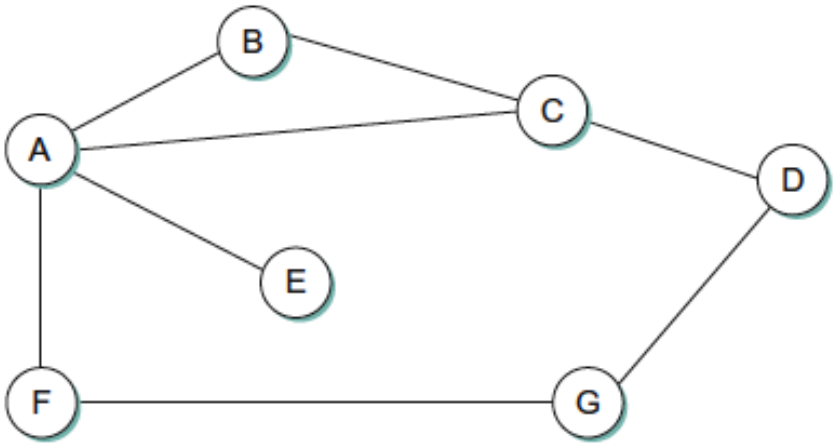
- B are ruta de distanta 2 catre E prin A
- B nu include noua distanta catre E in actualizarea trimisa lui A

split horizon with poison reverse: trimite o valoare f. mare

- B trimite distanta ∞ catre A
- A nu va mai alege o cale prin B

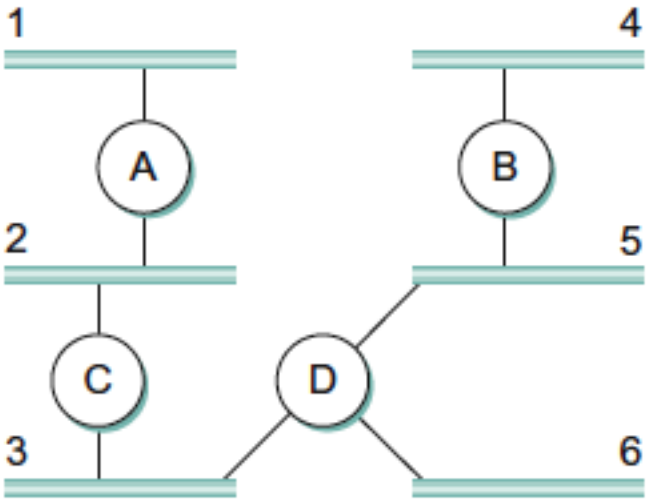
Neajuns: solutiile nu functioneaza in toate cazurile

- de ex. pentru bucle cu mai multe noduri



RIP - Routing Information Protocol

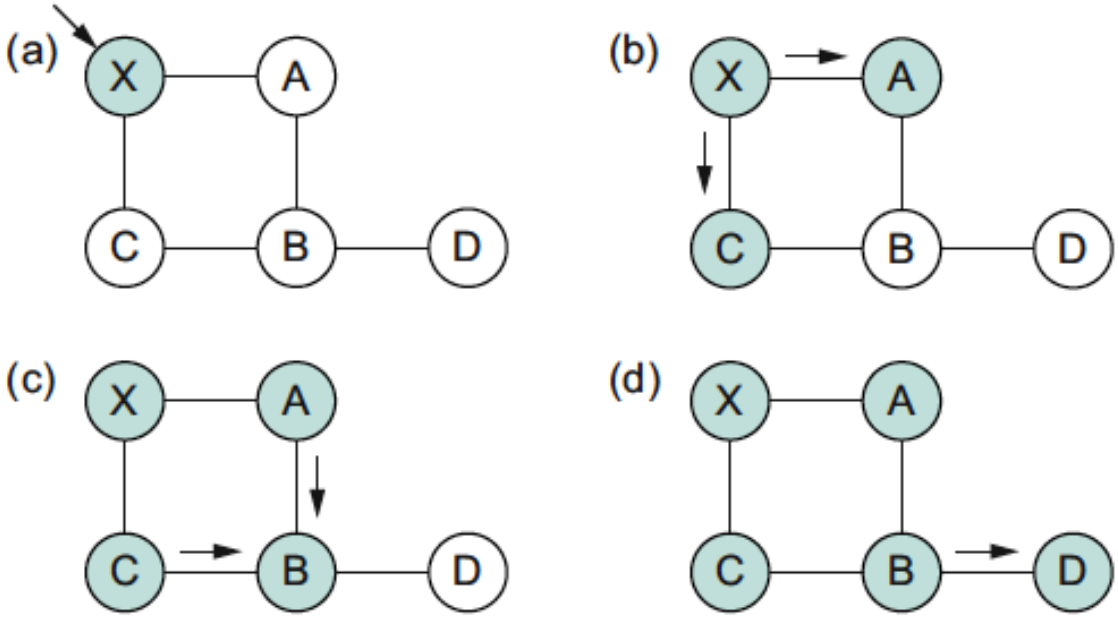
- fiecare legatura are cost 1
- foloseste distante la retele (nu la noduri)
  - ruterul C are distanta 0 la reteaua 2 si 2 la reteaua 4
- transmit vectorii distantelor la fiecare 30 secunde
- distante maxime de 15 hop-uri (16 inseamna infinit)
  - rețele de mici dimensiuni



Starea legaturii (Link State)

- Presupune ca fiecare nod poate gasi legaturile cu vecinii si costul fiecarei legaturi
- Informatiile sunt diseminate prin inundare tuturor celorlalte noduri
  - LSP – Link State Packet transmis prin inundare;
  - Pachetul contine
    - Id-ul nodului care creaza pachetul
    - lista nodurilor conectate cu costul fiecarei legaturi
    - un numar de secventa
    - durata de viata a pachetului (numar)
- Cu informatiile primite, fiecare nod va calcula rutele cele mai scurte catre celelalte noduri

Transmiterea prin inundare



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

Algoritmul căii celei mai scurte

Algoritmul lui Dijkstra

nnod	numărul nodurilor reţelei;
sursa	nodul sursă;
l[i][j]	costul legăturii (i,j), având valorile 0 dacă i = j; lungmax dacă i şi j nu sunt adiacente; o valoare între 0 şi lungmax în celelalte cazuri;
D[i]	costul minim al legăturii de la sursă la i;
S	mulţimea nodurilor deja selectate;
V	tabloul de dirijare; V[i] = 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

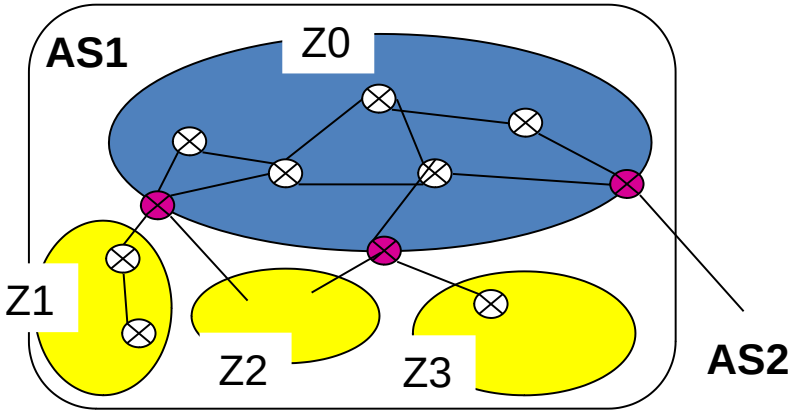
- Internet-ul este partajat in mai multe domenii numite sisteme autonome AS – Autonomous Systems
  - AS-urile sunt retele independente operate de organizatii diferite
- Intr-un AS se foloseste acelasi algoritm de rutare “intra-domeniu”
  - denumit interior gateway protocol
  - ex. OSPF – Open Shortest Path First
- Rutarea intre AS-uri (intre domenii) foloseste, de asemenea, un protocol comun
  - denumit exterior gateway protocol
  - ex. BGP – Border Gateway Protocol (bazat pe RIP)

Structura ierarhica AS

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

- zona Z0 – coloana vertebrală (backbone)
- zone "stub" Z1, Z2 … – legate la backbone
- rutele intre noduri din zone stub diferite trec prin zona Z0

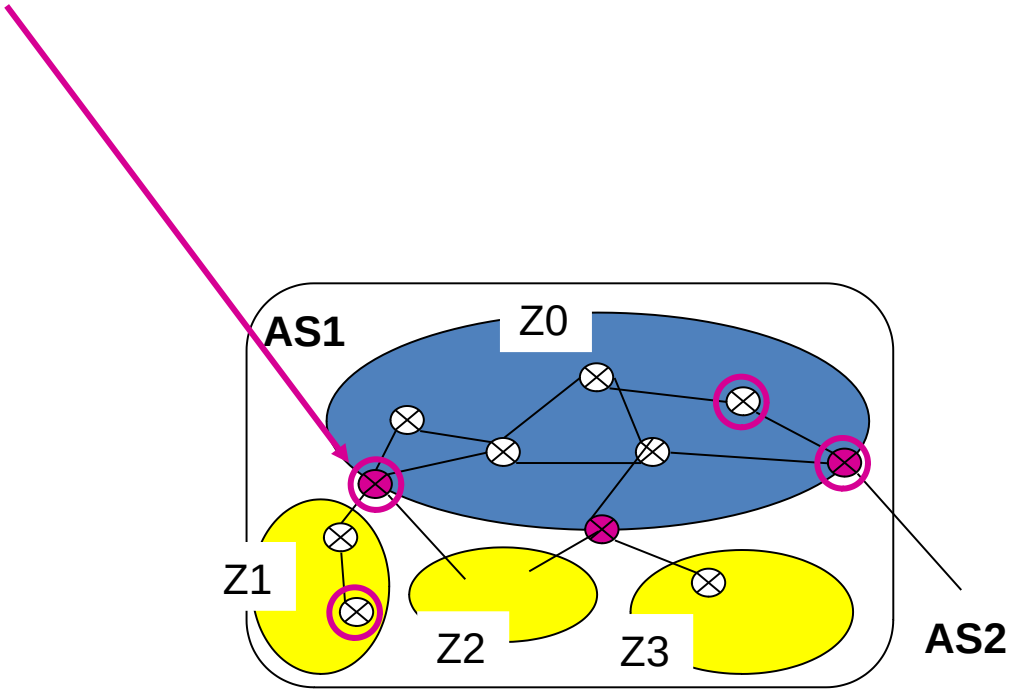
- Ierarhizarea crește scalabilitatea
  - un ruter dintr-un AS nu trebuie sa stie cum se ajunge la fiecare retea din AS, fiind suficient sa stie cum se ajunge la zona in care se afla reteaua respectiva reduce volumul tabelelor de dirijare



Structura ierarhica AS (2)

Tipuri de rutere

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



Mesaje OSPF

Mesajele OSPF permit schimbul de informatii intre noduri

Sunt transmise in pachete IP cu 89 ca numar de protocol

Hello – stabileste si pastreaza legaturi cu vecinii

descopera nodurile (rutere) 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

trimite info despre legaturi, ca raspuns la o cerere

Link state ack - Confirmare stare legătură

confirmă primirea unui mesaj de actualizare

Database description – trimite LSDB – LInk State Data Base

mesajele contin info despre AS sau zonă

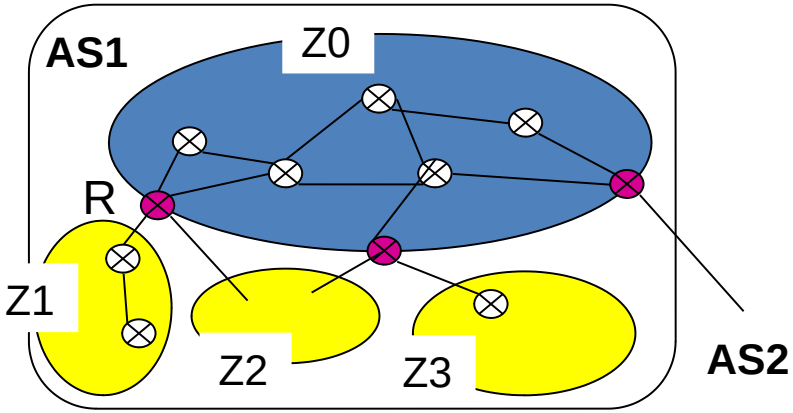
Calcul rute - Nivel 1 (zona stub)

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

- ex. informatiile de starea legaturilor schimbate intre noduri din Z1 nu se transmit in afara acestei zone!

Fiecare ruter (inclusiv cele de granita zonala) din **zonă** calculează separat căile cele mai scurte catre ruterele din aceeasi zona

- in final, un ruterul de graniță zonală R va cunoaste caile cele mai scurte catre oricare rețea din Z1 si Z2



Calcul rute - Nivel 2 (AS)

Ideea: calea unui pachet intre doua zone diferite are trei parti:

- de a nodul sursa la zona backbone
- traverseaza backbone
- de la backbone la reteaua de destinatie

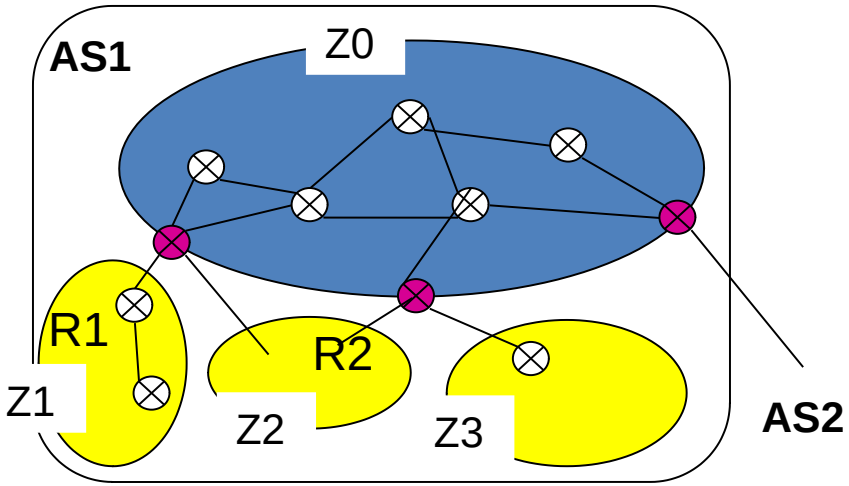
Ruterele de coloana vertebrala (backbone) primesc informatii de la ruterele de granita zonale si calculeaza cele mai bune rute la retele din orice zona

ex. R1 ofera caile cele mai scurte pentru Z1 si Z2

ruterele din Z0 calculeaza caile catre orice retea din Z1,Z2

Rezultatele sunt difuzate de la Z0 inapoi la zonele stub, care actualizeaza caile cele mai scurte la retele din alte zone.

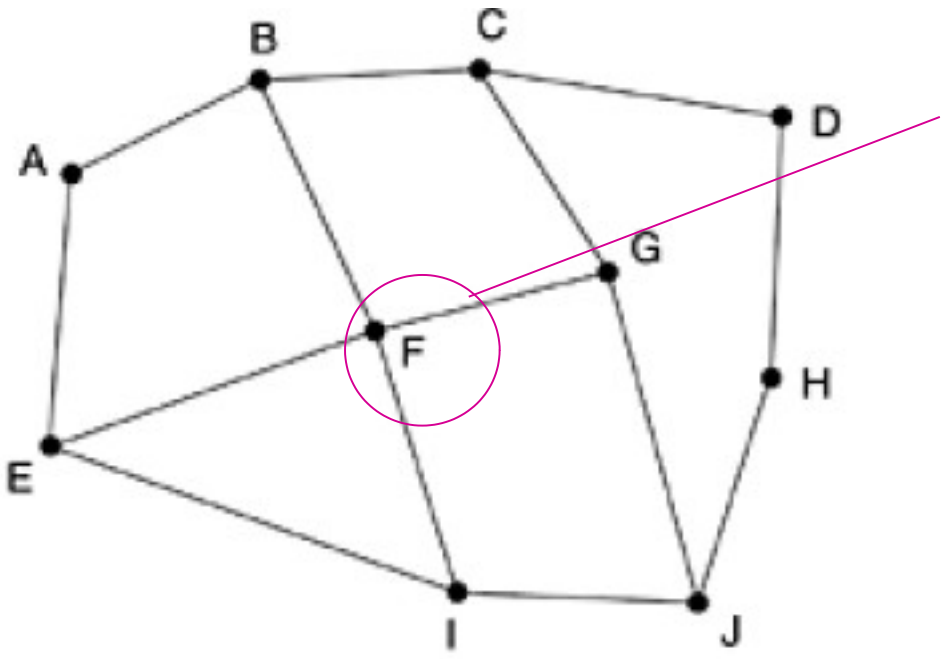
Ex. ruterele din Z2 vor sti sa aleaga intre R1 si R2 pentru rutare spre retele din alte zone



BGP – Border Gateway Protocol

Algoritmi orientaţi pe aspectele politice, de securitate, economice
Reţea = AS-uri şi conexiunile
Protocol = vectorul distanţelor
Tabelele de dirijare conţin şi rutele spre destinaţie
Comunică vecinilor căile utilizate efectiv (ocoleste numaratoarea la infinit)

Ex. pp. F foloseste calea FGCD la D



G se defecteaza
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 reţele ad hoc

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

reţea ad hoc = graf

Muchie = conexiune – nodurile pot comunica direct (radio)

Fiecare nod = ruter + gazdă

Conţine

Tabela dirijare

destinaţie,

pas următor,

distanţă

nr secv destinaţie

altele

Tabela history

identitatile cererilor precedente

Tabela reverse route

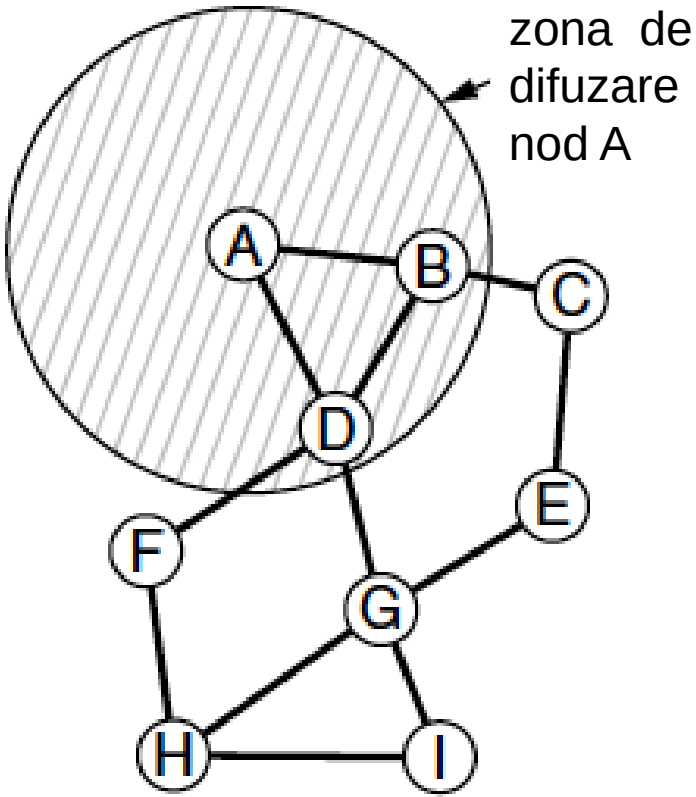
calea spre sursa unui pachet de cerere

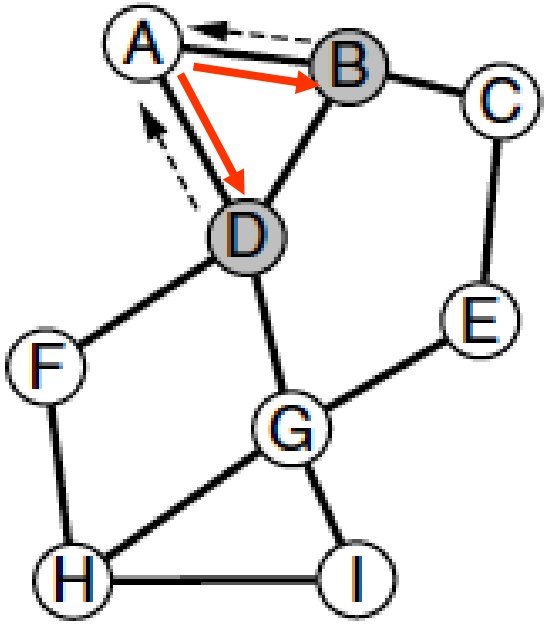
Cum functioneaza?

Exemplu: A vrea sa comunice cu I care nu e în tabela sa

trebuie să descopere ruta

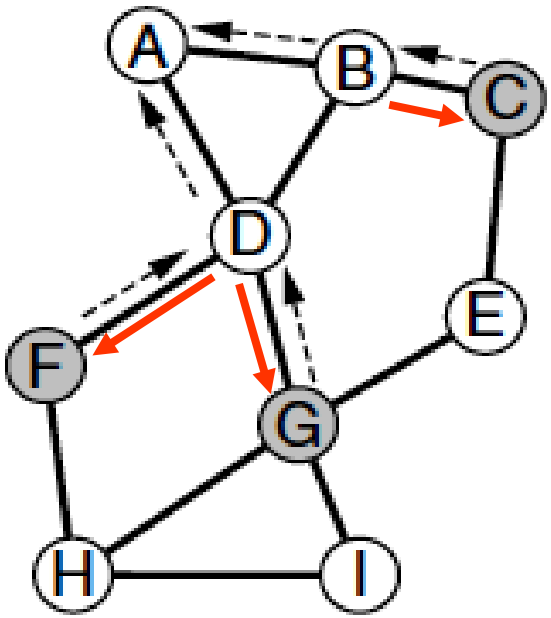
trimite (broadcast) o cerere de ruta, care ajunge la nodurile B si D

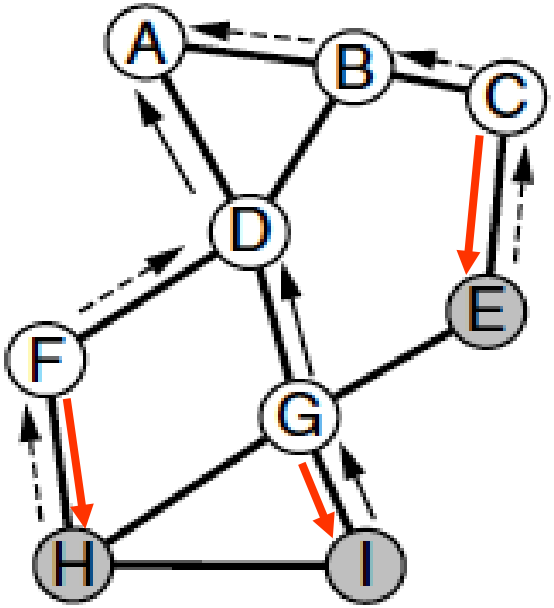




- daca B si D nu cunosc ruta spre I, cererea este re-transmisa, prin inundare, celorlalte noduri
  - mai intai C, F si G
  - apoi E, H si I
- daca cererea ajunge la I, acesta va returna un raspuns catre A
  - raspunsul parcurge calea inversa prin G si D la A (sageti negre pline)

⇒ fiecare nod trimite raspunsul vecinului de la care a primit cererea (sageti punctate)





Probleme

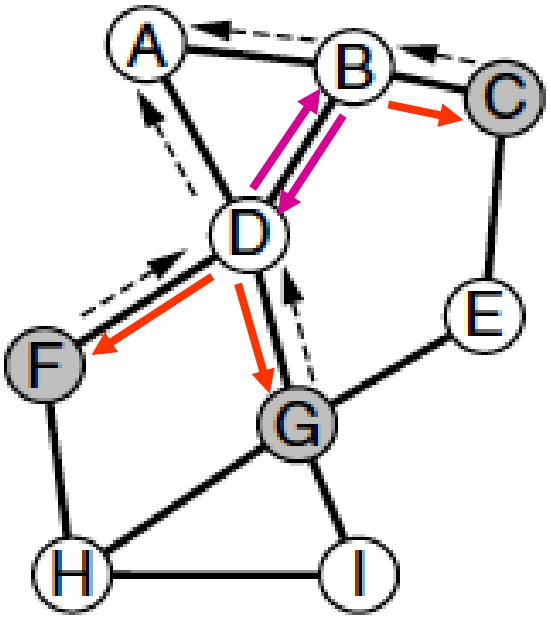
Eliminarea replicilor

- pachete transmise prin inundare
- B primeste de la D o copie a pachetului
- D primeste de la B o copie a pachetului
- pentru a descoperi si elimina copiile
  - pachetul contine un **ID al cererii**
  - nodurile pastreaza cererile primite

in **tabela history**

Lungimea caii A → I

- pachetul de cerere are un camp Hop count
- initializat de A la 0
- incrementat de fiecare nod care re-transmite pachetul
- returnat de I in raspunsul lui



Probleme-2

Nodul A poate avea o ruta veche

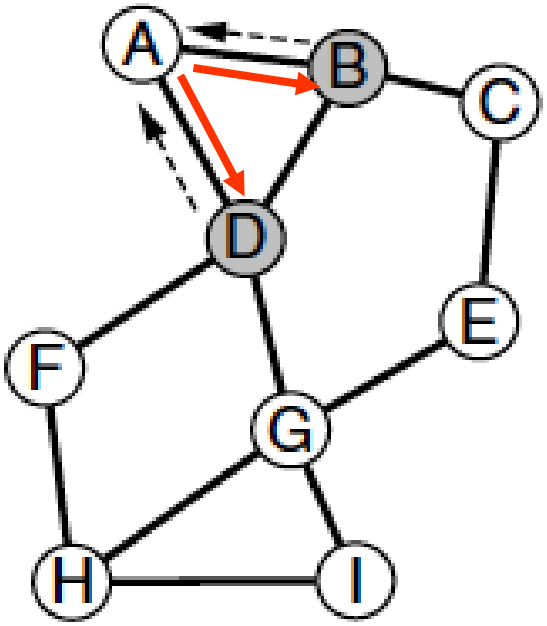
- trimite o cerere pentru actualizarea ei

Se foloseşte Sequence # pentru a deosebi rutele noi de cele vechi

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

Exemplu

- daca nodul D are o ruta mai noua decat cea din cerere, nu va mai re-difuza cererea si va trimite raspunsul pe calea inversa



Informatii pastrate de noduri

Tabela dirijare

- destinaţie,
- pas următor,
- distanţă,
- nr secv destinaţie
- 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
- Foloseşte Sequence # pentru a deosebi rutele noi de cele vechi

Prelucrarea ROUTE REQUEST în fiecare nod

- Verifica duplicat 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 şi re-difuzează ROUTE REQUEST
  - memorează informaţia în reverse route table
- Source sequence # folosit pentru actualizare tabela dirijare locala

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

Pachete ROUTE REPLY

I construieşte ROUTE REPLY şi-l trimite 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

Intreţinerea rutelor

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

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

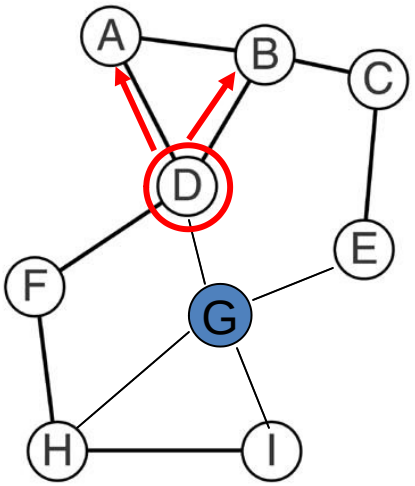
D află că G a fost utilizat pe rute către E, G şi I

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

D goleşte intrările pentru E, G şi 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)

Studiu individual

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

5.1 CERINŢELE DE PROIECTARE ALE NIVELULUI REŢEA

5.2.2 Dirijarea pe calea cea mai scurtă

5.2.3 Inundarea

5.2.4 Dirijare cu vectori distanţă

5.2.10 Dirijarea în reţele AD HOC

5.6.1 Protocolul IP

5.6.2 Adrese IP

5.6.4 Protocoale de control în Internet

5.5.5 Protocolul de dirijare folosit de porţile interioare: OSPF

5.6.5 Protocolul de dirijare pentru porţi externe: BGP

5.6.8 IPv6

Studiu individual

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

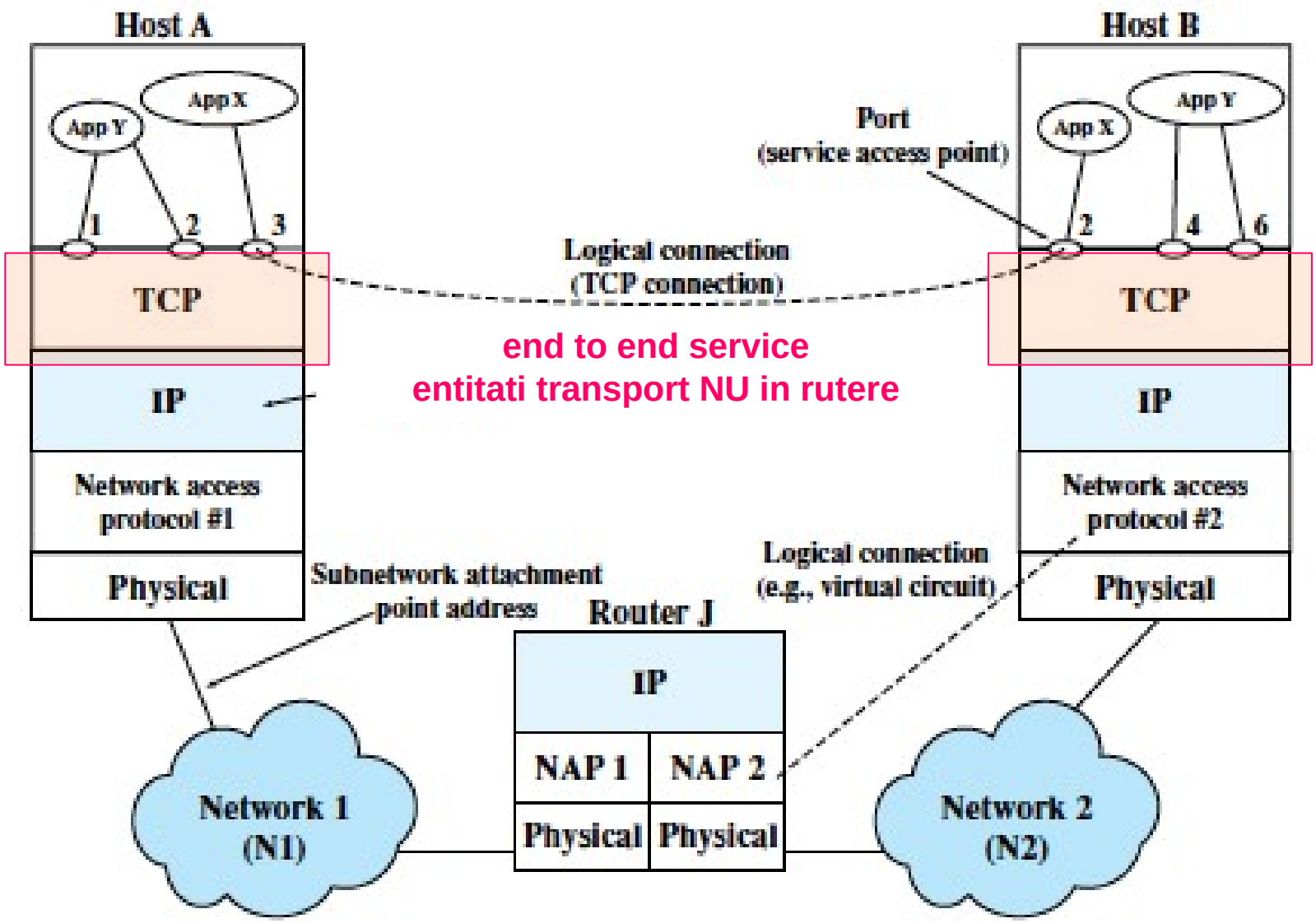
5.1 NETWORK LAYER DESIGN ISSUES
5.2.2 Shortest Path Algorithm
5.2.3 Flooding
5.2.4 Distance Vector Routing
5.2.11 Routing in Ad Hoc Networks
5.6.1 The IP Version 4 Protocol
5.6.2 IP Addresses
5.6.3 IP Version 6
5.6.4 Internet Control Protocols
5.6.6 OSPF—An Interior Gateway Routing Protocol
5.6.7 BGP—The Exterior Gateway Routing Protocol

Nivelul transport

Cuprins

- Servicii si caracteristici
- Socket API
- Protocolul UDP
- ProtocolulTCP
  - gestiunea conexiunilor TCP
  - corectitudinea
  - controlul fluxului de date
  - mai multe numere de secventa
  - dimensiunea ferestrei recceptorului
  - controlul congestiei, ceasurilor, time-out

Nivelul transport in ierarhia TCP/IP

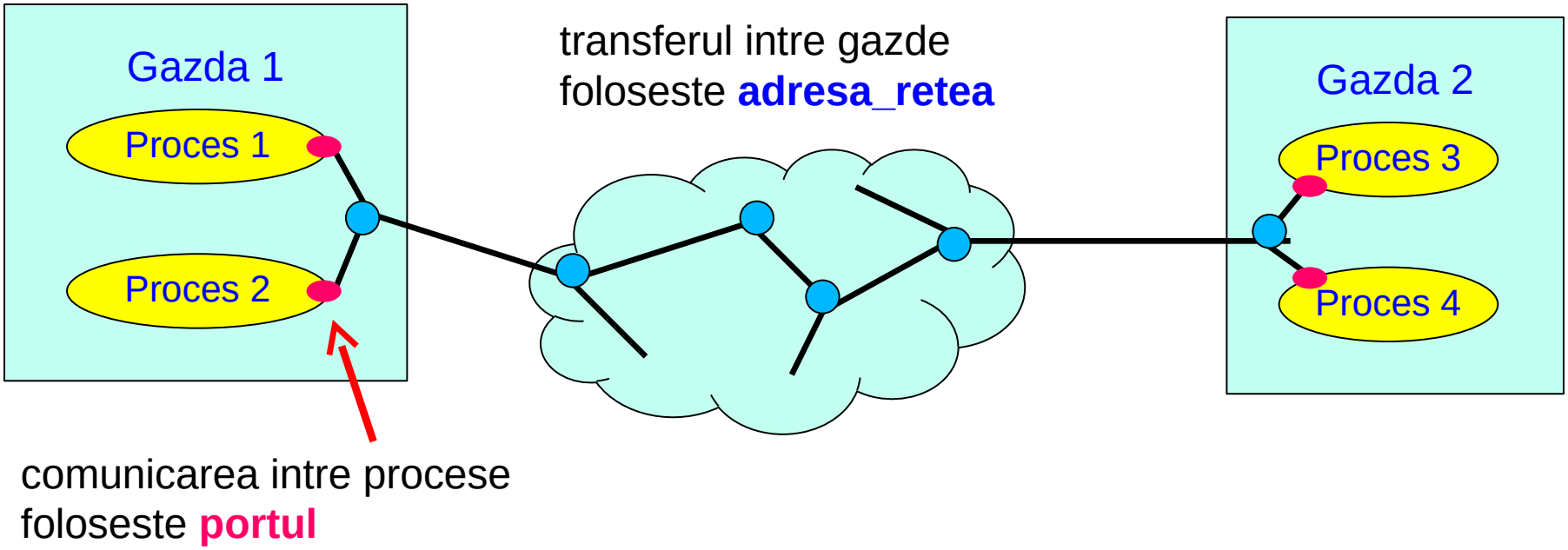


Comunicarea intre aplicatii

Reteaua asigura transmiterea pachetelor (datagrame) 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
  - interfaţa uniforma cu utilizatorii

- **Caracteristici**
  - două tipuri de servicii:
    - orientate pe conexiune (connection oriented) - TCP
    - fără conexiune (connectionless) - UDP

Protocoale de Transport

Doua protocoale majore

- UDP
- TCP

Aspecte discutate

- Formatul datelor
- Adresare
- Functionare

UDP - User Datagram Protocol

- UDP livreaza datagrame utilizator - *user datagrams*
  - *Nu se realizeaza o conexiune*
  - Livrare “Best effort” – datagramele pot fi pierdute, primite in alta ordine etc.
  - Sume de control pentru integritate

TCP - Transmission Control Protocol

Orientat pe conexiune

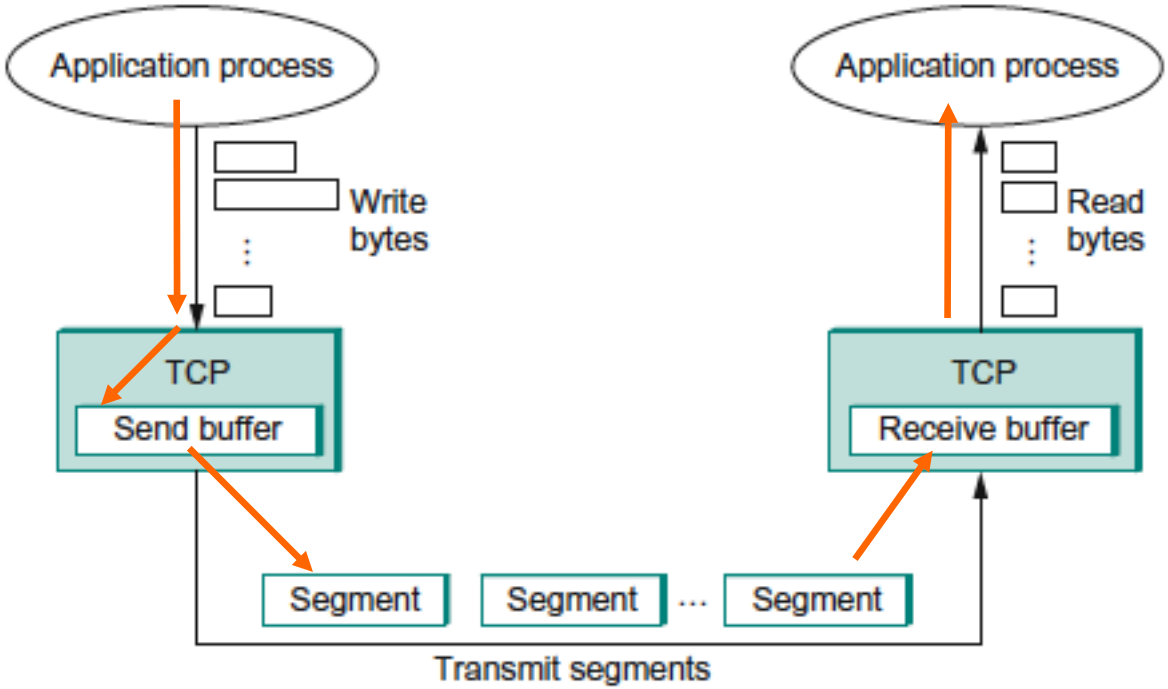
Livrare sigura pe retea nesigura

- ***Cel mai folosit protocol de transport***
  - Web
  - Email
  - SSH
  - Chat
  - Video streaming
  - Peer-to-peer

Cateva porturi standard 0 - 65535

Port	Protocol	Use
20, 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 (câți vrea!) din conexiunea TCP

Caracteristici

- Orientat pe conexiune
- Interfață flux (Stream)
  - transmisie si 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

TCP

UDP

Socket interface

- Serviciile nivelului transport sunt accesibile ca API – Application Programming Interface

- Este un standard de facto pentru comunicarea intre procese pe internet

- Oferită ca bibliotecă de functii folosite pentru

dezvoltarea aplicatiilor Internet, pentru diverse

sisteme de operare

Socket API

- Originară din Berkeley Software Distribution (BSD) UNIX, apoi adoptata ca standard POSIX.
- Disponibilă si pe Windows, Solaris, etc.
- socket = punctul în care procesul de aplicație se atașază la rețea. Este identificat prin descriptor (număr), in acelasi mod ca un fisier

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 in apelurile urmatoare

protocol_family selecteaza familia de protocoale

PF_INET - protocoale Internet

PF_APPLETALK - protocoale AppleTalk etc.

comm_type selecteaza tipul de comunicare

SOCK_DGRAM - fara conexiune - datagrama

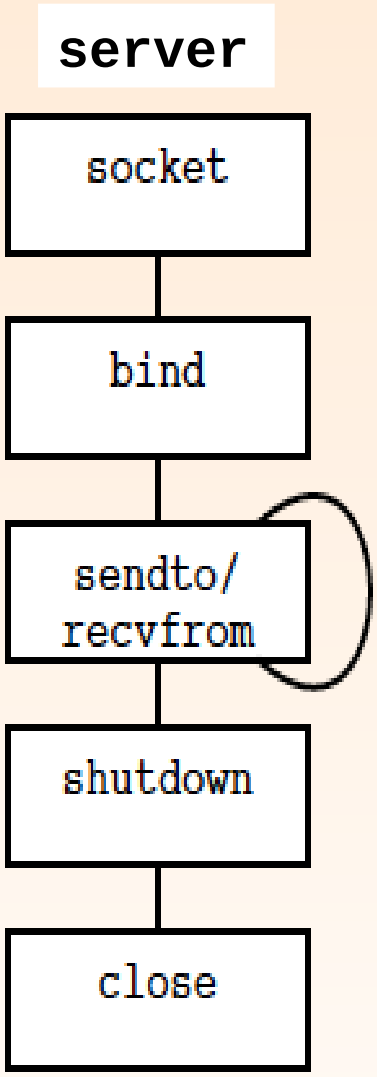
SOCK_STREAM - orientat pe conexiune – flux de octeti

protocol specifica protocolul

IPPROTO_TCP - TCP

IPPROTO_UDP - UDP

Serviciu fara conexiune - server



Creeaza socket si aloca-i resurse de sistem:

int s = ***socket*** *(PF_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

Primeste mesaje de la un socket aflat la distanță:

int recvfrom (int socket_descriptor, char* buffer_address, int buffer_length, int flags, struct sockaddr* sender_address, unsigned int * sendaddress_length)

recvfrom(s, buf, BUFLEN, 0, NULL, NULL);

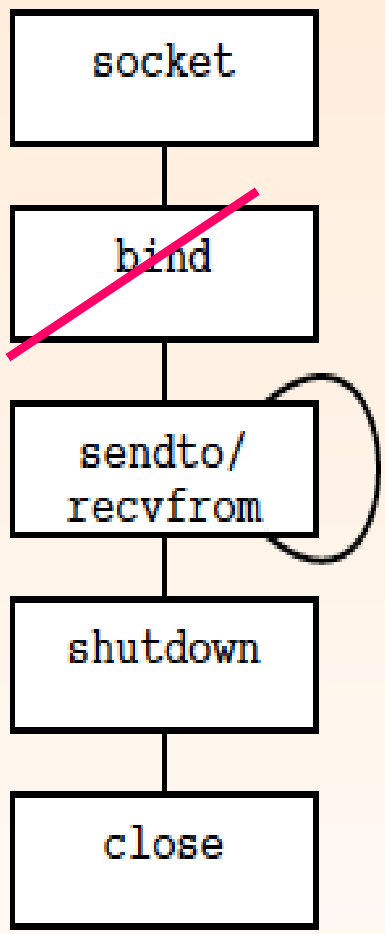
Oprește trimitere sau/și recepție de date

shutdown (s, SHUT_RD / SHUT_RDWR / SHUT_WR)

Inchide socket - termina utilizarea socket si elibereaza resurse alocate

close (s)

Serviciu fara conexiune - client



Creeaza socket și aloca-i resurse de sistem:

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

Trimite mesaj la un socket aflat la distanță:

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

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

Opreste trimitere sau/și receptie de date

shutdown (s, SHUT_RD / SHUT_RDWR / SHUT_WR)

Termina utilizarea socket si elibereaza resursele alocate

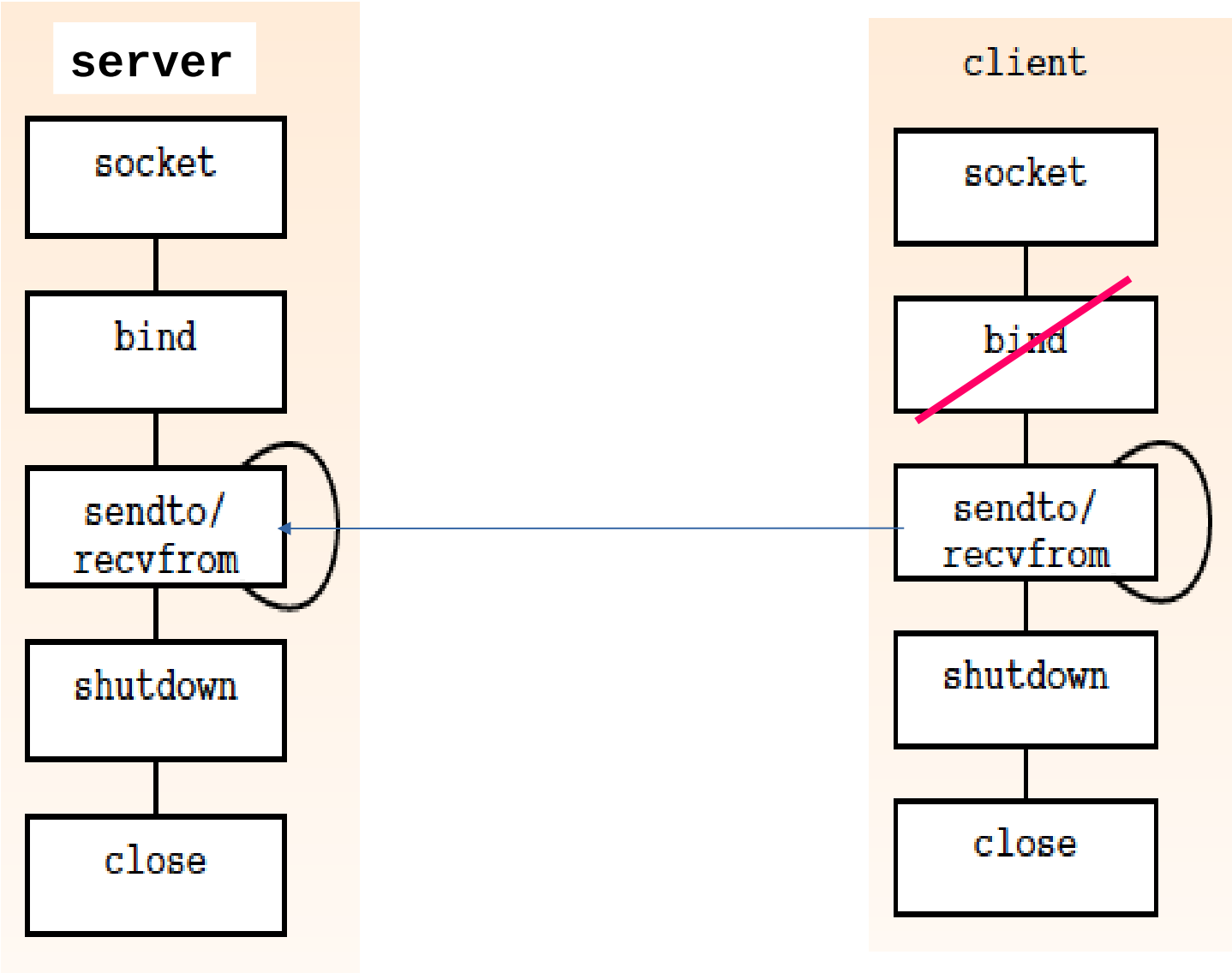
close (s)

Setare adresa

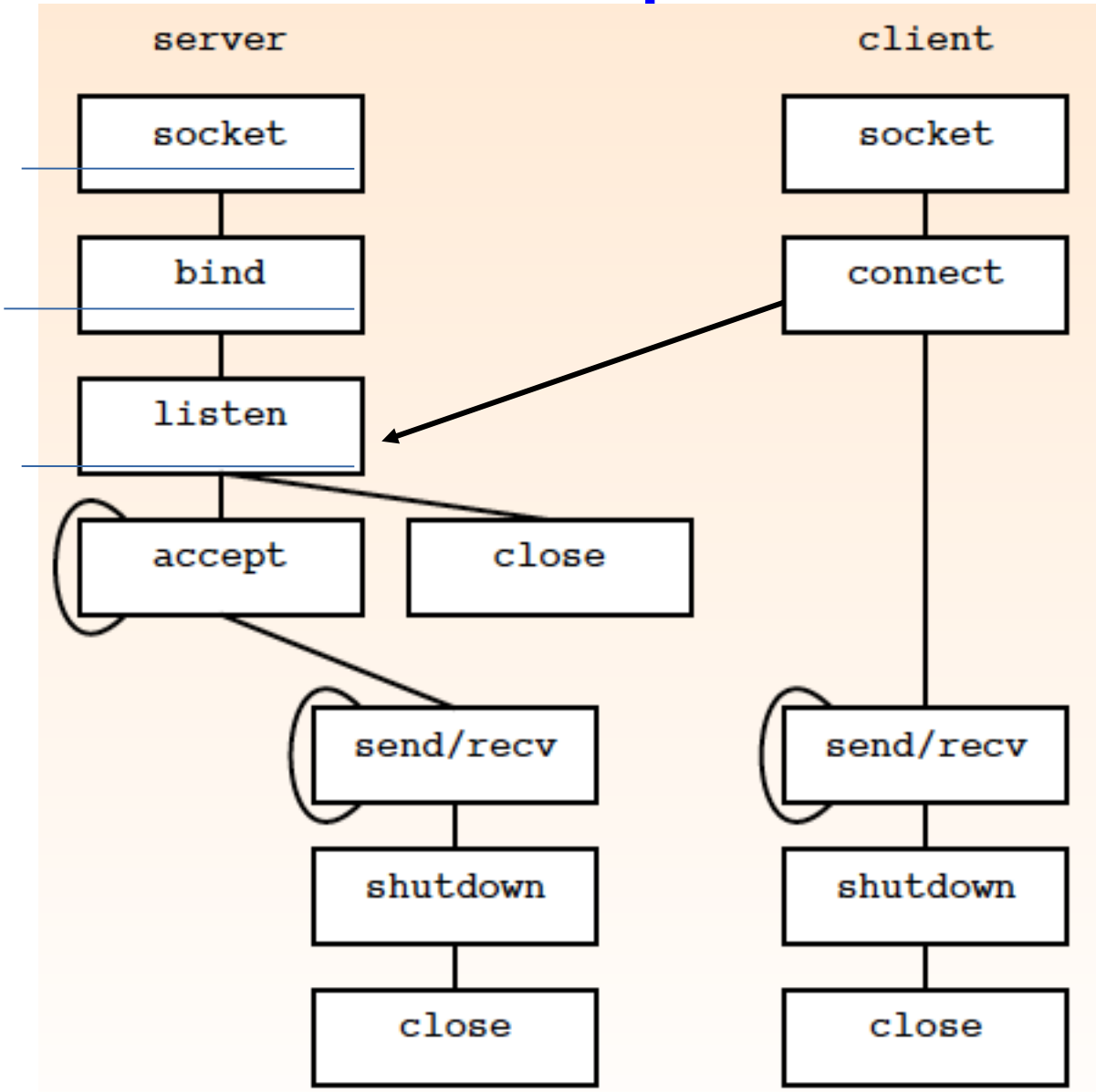
Format TCP/IP:

```
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
```

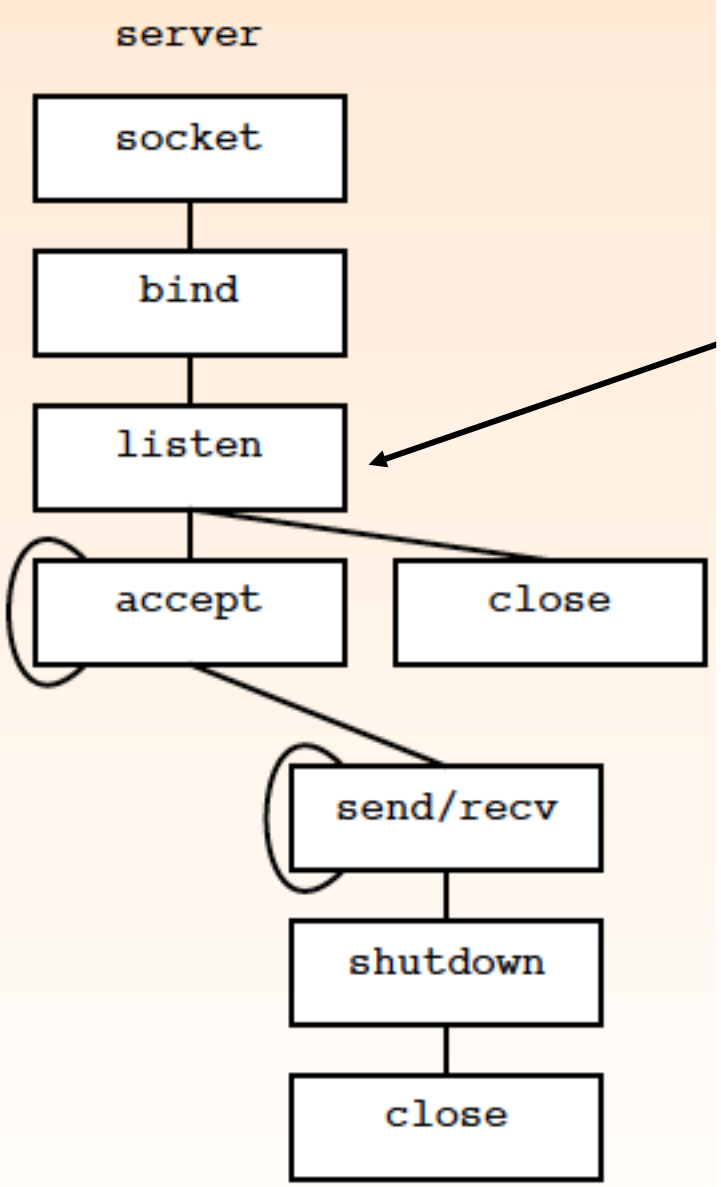
UDP


server
socket
bind
sendto/
recvfrom
shutdown
close
client
socket
bind
sendto/
recvfrom
shutdown
close

Serviciu orientat pe conexiune



Serviciu orientat pe conexiune - server



1. Creeaza socket:

*int ls = **socket** (PF_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 in asteptare):

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 - Trimite / primeste pe s etc.

Serviciu orientat pe conexiune - client

1.Creaza socket:

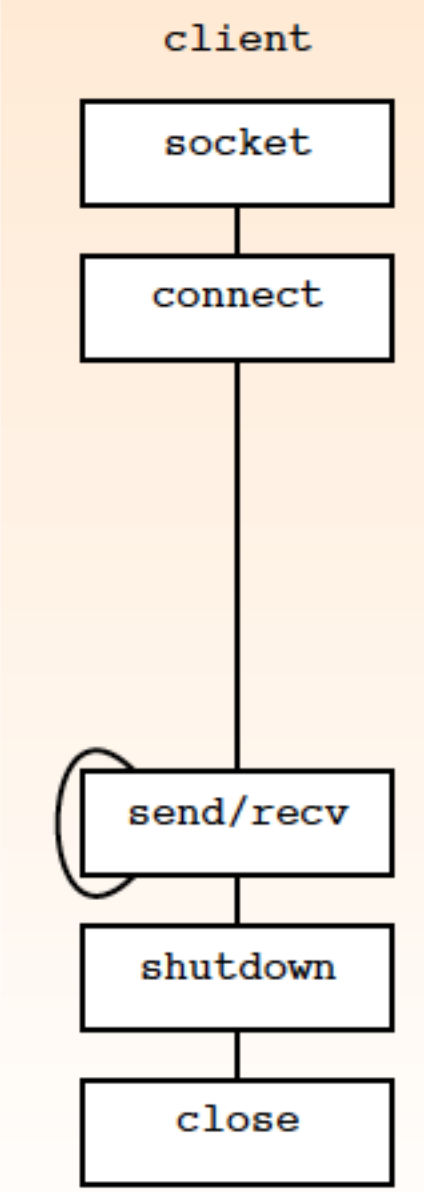
int s = ***socket*** *(PF_INET, SOCK_STREAM, IPPROTO_TCP);*

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

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

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

3. repetat - Trimite / primeste etc.



Transmisia de date cu TCP

send (s, buf, len)

int send(int socket, const char* buf, int len, int flags);

- Intoarce numarul de octeti trimisi
  - **Poate fi mai mic decat len !**

recv (s, buf, max_len)

int recv(int socket, char* buf, int len, int flags);

- Intoarce numarul de octeti primiti
  - **Poate fi mai mic decat max_len!**
- flags indica optiuni speciale

  MSG_OOB – trimite/primeste date out-of-band

  MSG_PEEK – livreaza date primite, dar trateaza ca necitite

Inchiderea conexiunii TCP

- Elibereaza resursele asociate conexiunii
- Informeaza capatul celalalt de inchiderea conexiunii
- API
  - shutdown (s,SHUT_RD/SHUT_RDWR/SHUT_WR)

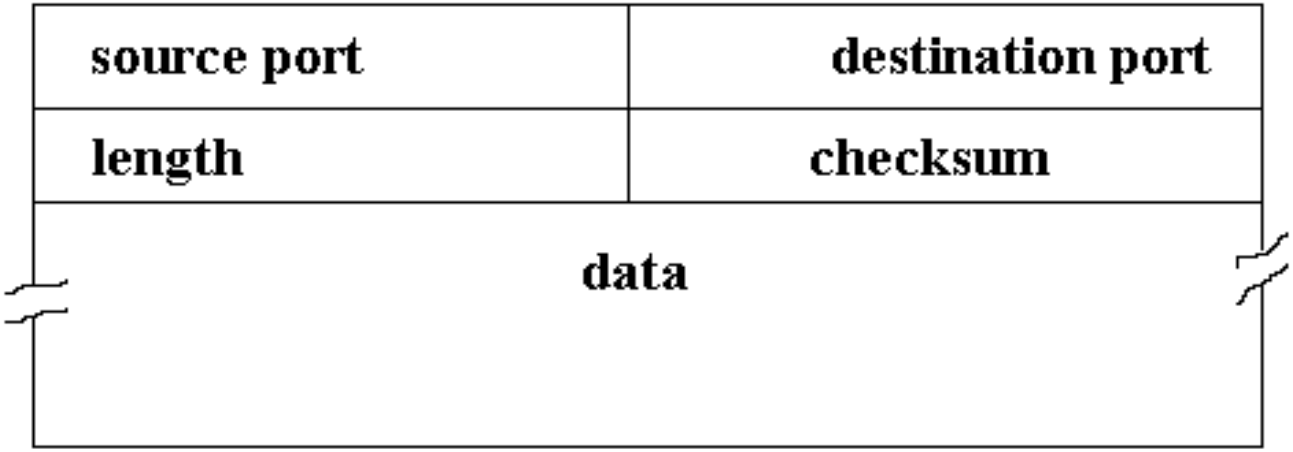
    int shutdown (int socket, int how)

    - opreste primirea – rejecteaza datele care sosesc
    - opreste transmiterea – ignora datele inca netrimise
    - ambele
  - close (s)

    int close (int socket);

    - inchide socket (elibereaza structurile de date din kernel)

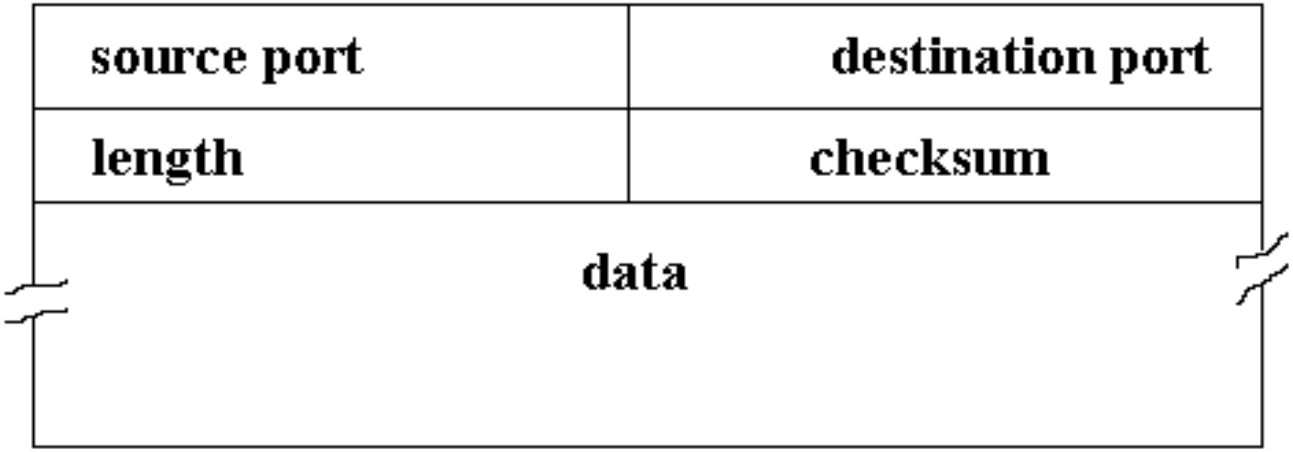
Antet UDP



NU ASIGURA: control flux, control erori, retransmisie

- UDP identifica adresa Internet si numar port pentru sursa si destinatie
- *Destination port* si *source port* pot diferi.

Antet UDP



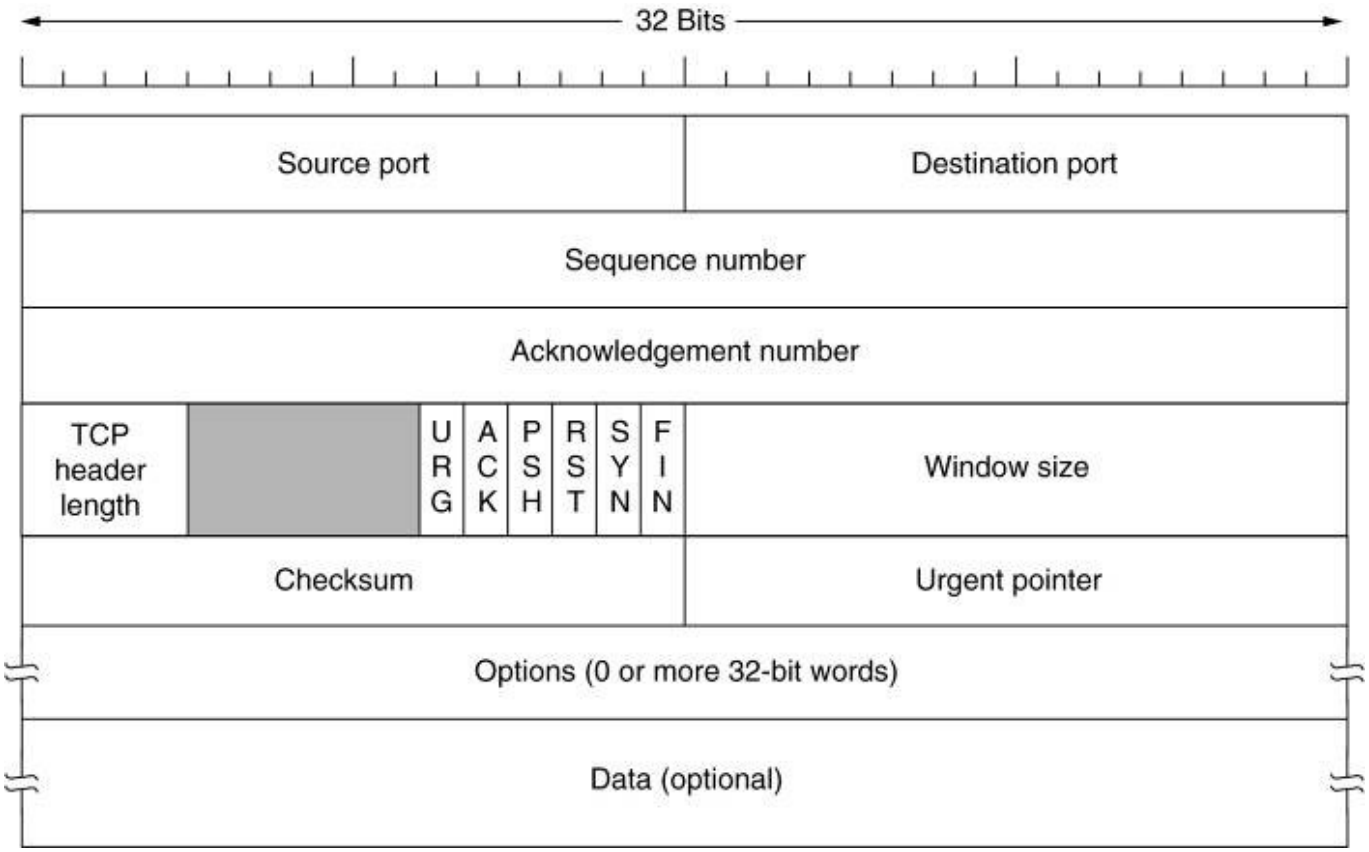
Checksum:

- Include antet, data si un pseudoantet IP care contine:
    - Adresele IP pentru sursa si destinatie
    - Numar protocol UDP (17) si lungimea segment
- Algoritmul aduna in complement fata de 1 toate cuvintele de 16 biti si retine complementul fata de 1 al sumei
- Receptorul face calculul pe intregul segment, inclusiv checksum, iar rezultatul trebuie sa fie 0

Aplicatii 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 aplicatii cand e nevoie (DNS)

Antet segment TCP



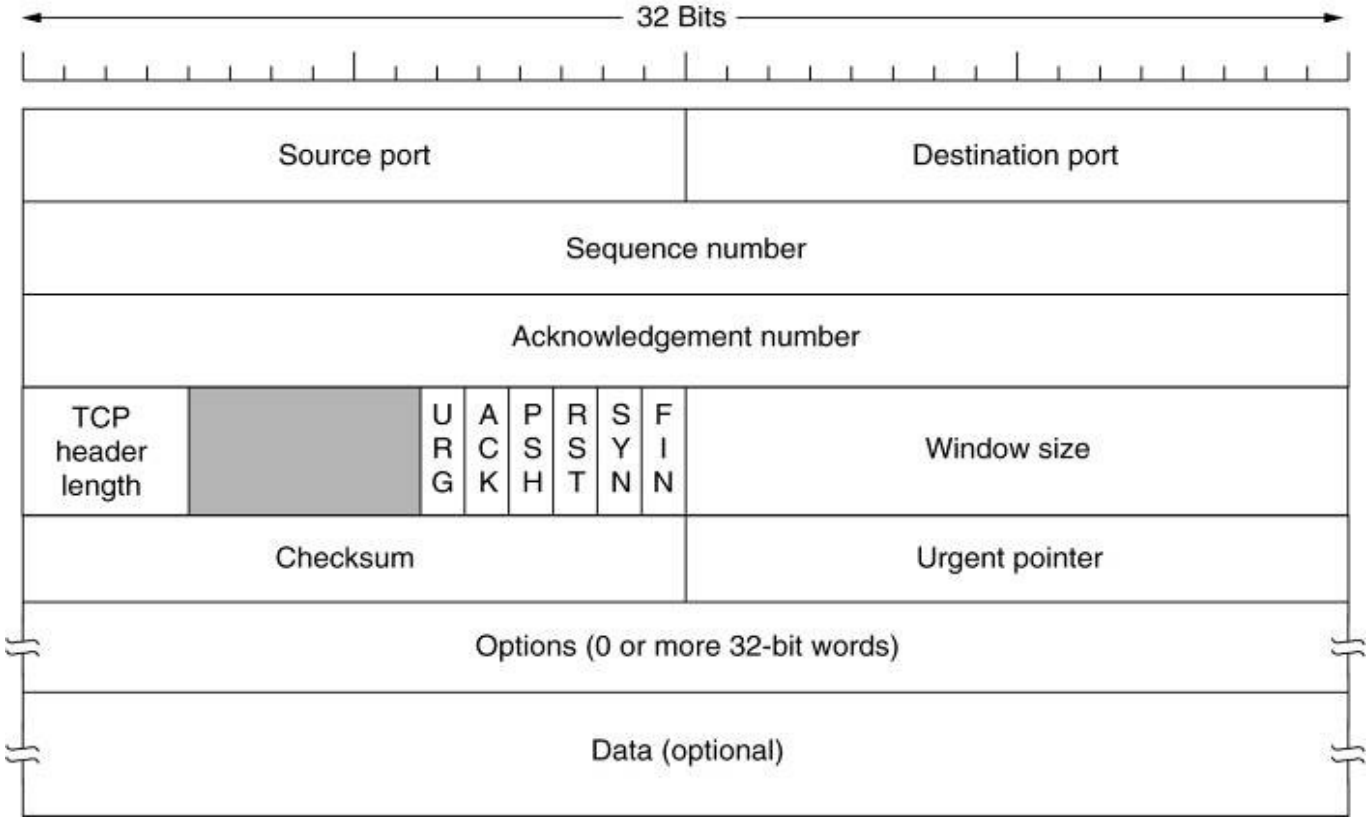
Sequence number – numarul primului octet din segment

Acknowledgement number – numarul urmatorului octet asteptat

Window size - numărul de octeţi care pot fi trimişi, începând cu ultimul octet

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

Antet segment TCP



URG Urgent pointer valid

ACK Acknowledge Number valid

PSH - push information to user

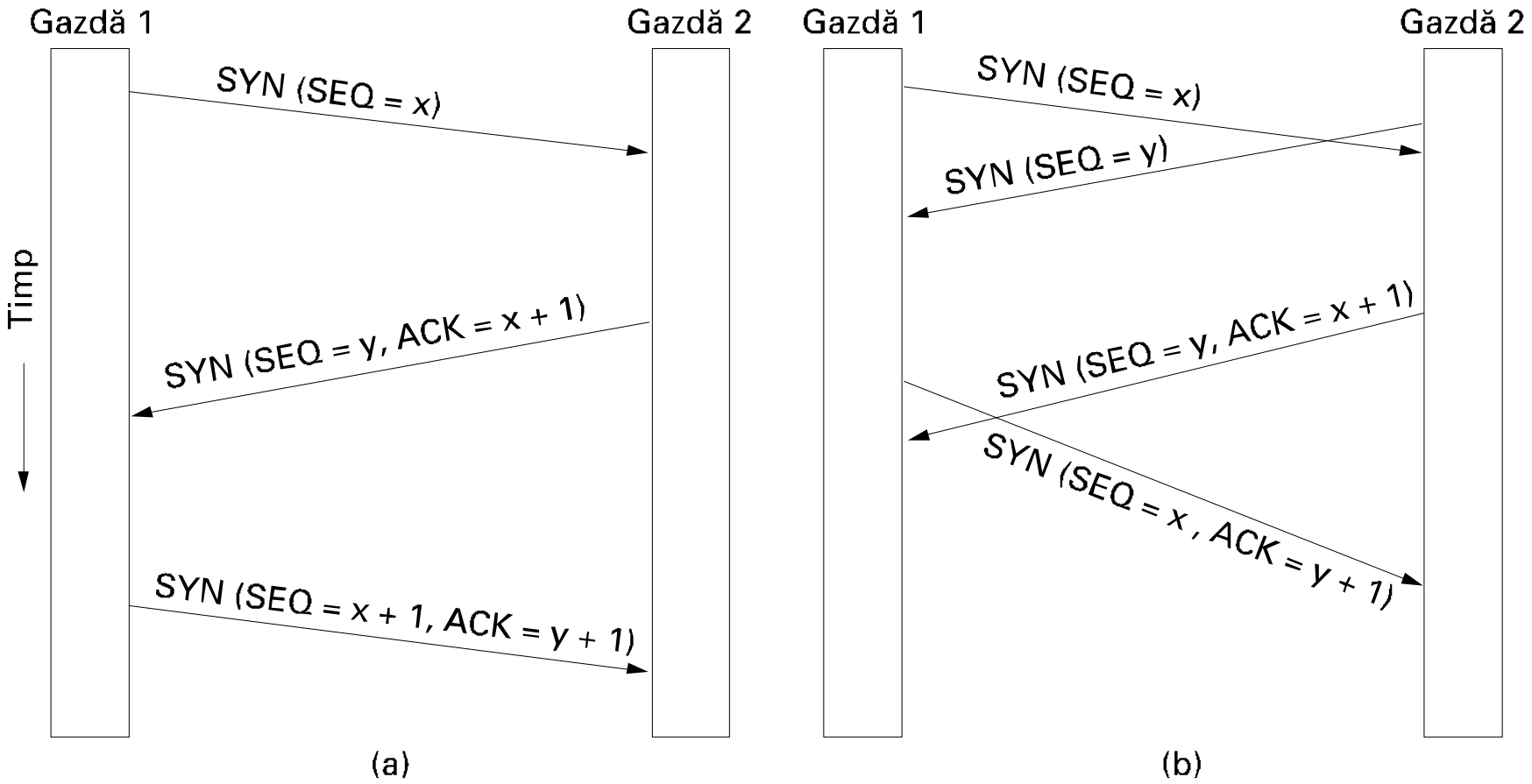
RST - close a connection due to an error

SYN - open connection

FIN - close a connection

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

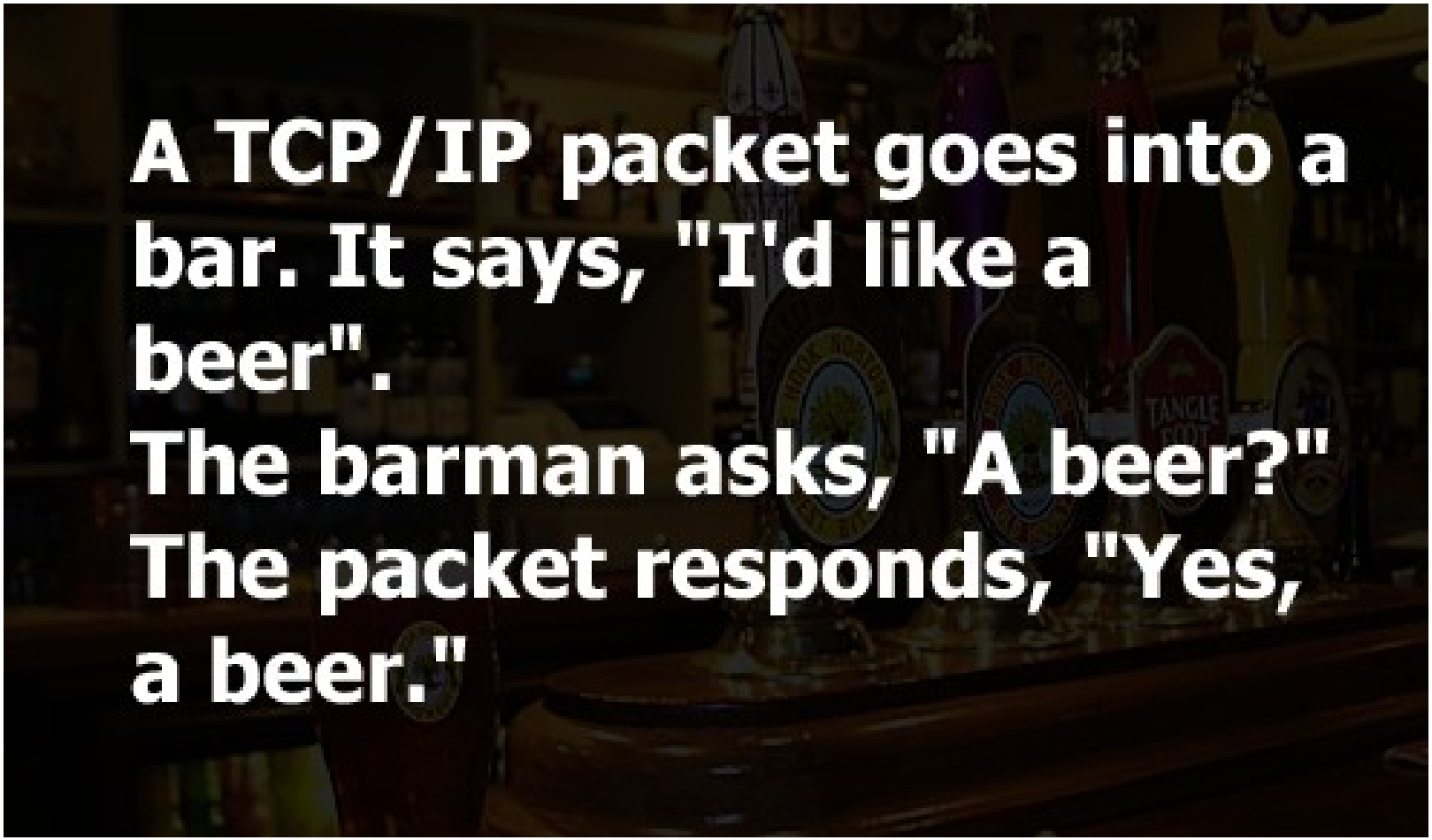
Stabilirea conexiunii - Three way handshaking



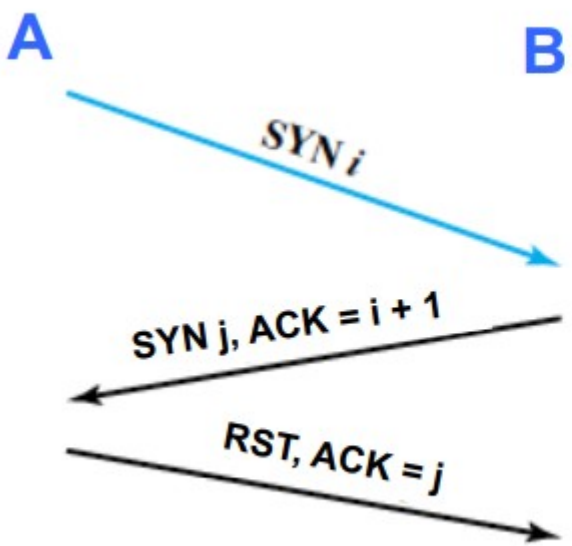
(a) Cazul normal.

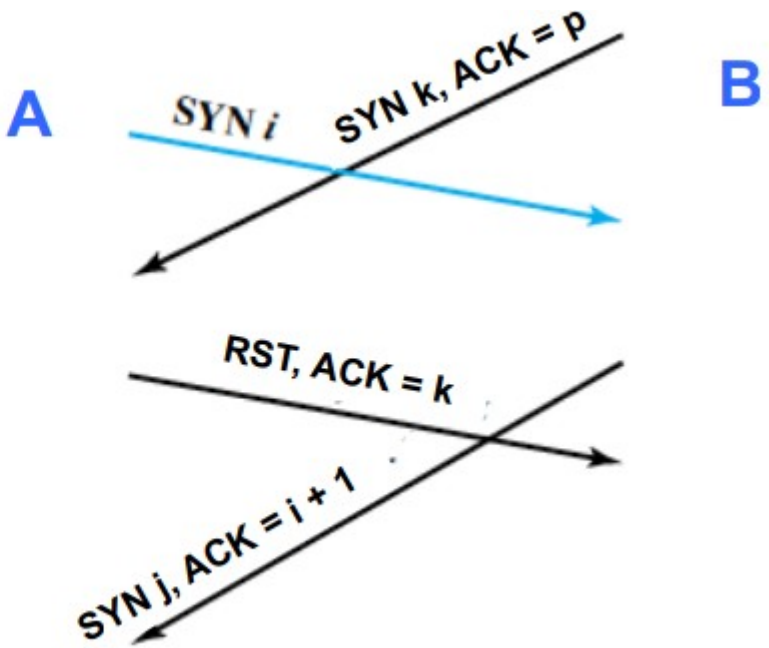
(b) Coliziune.

Stabilirea conexiunii - Three way handshaking



Rejectarea conexiunii





SYN intarziat

B accepta

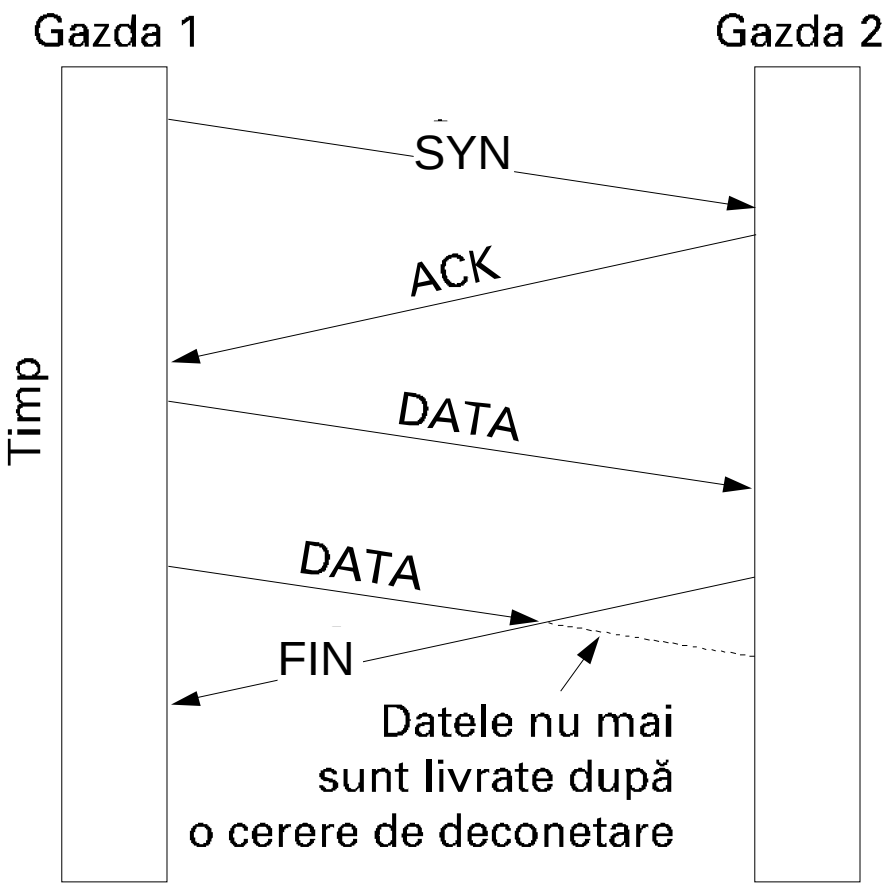
A rejecteaza

A initiaza conexiune (SYN i) + SYN k intarziat

A refuza (RST)

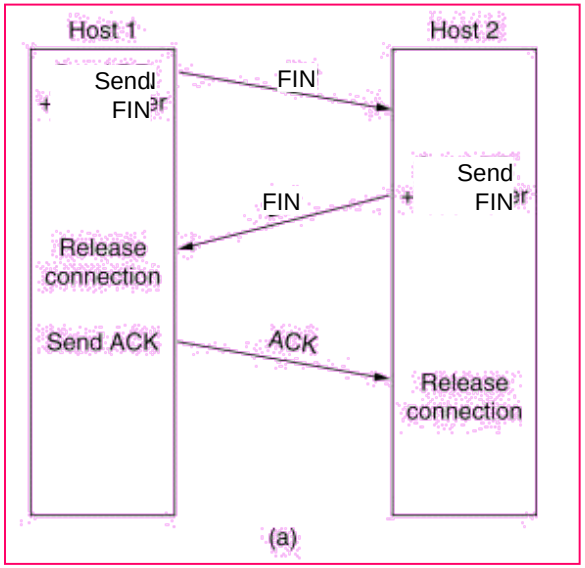
B accepta SYN i – raspunde cu SYN j

Deconectare abrupta cu pierdere de date

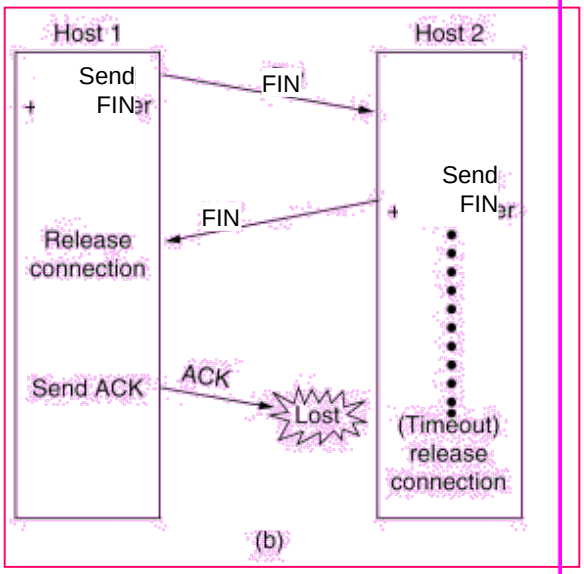


Patru scenarii de eliberarea conexiunii

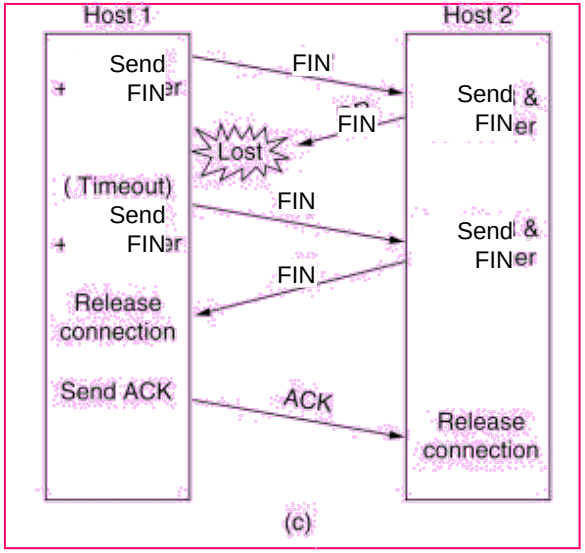
normal



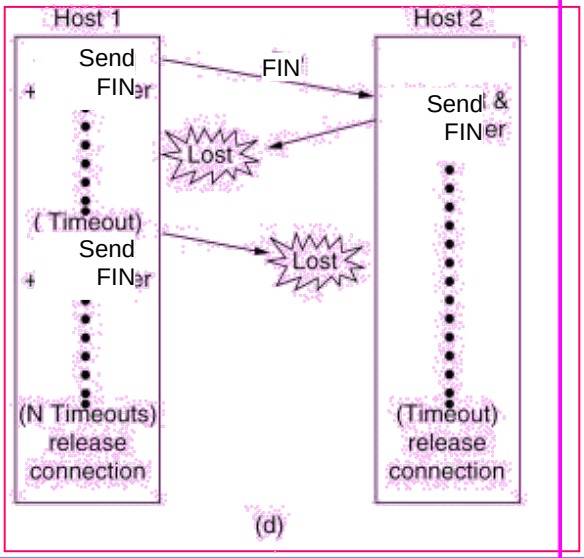
ACK final pierdut



raspuns pierdut



Raspuns si FIN pierdute



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

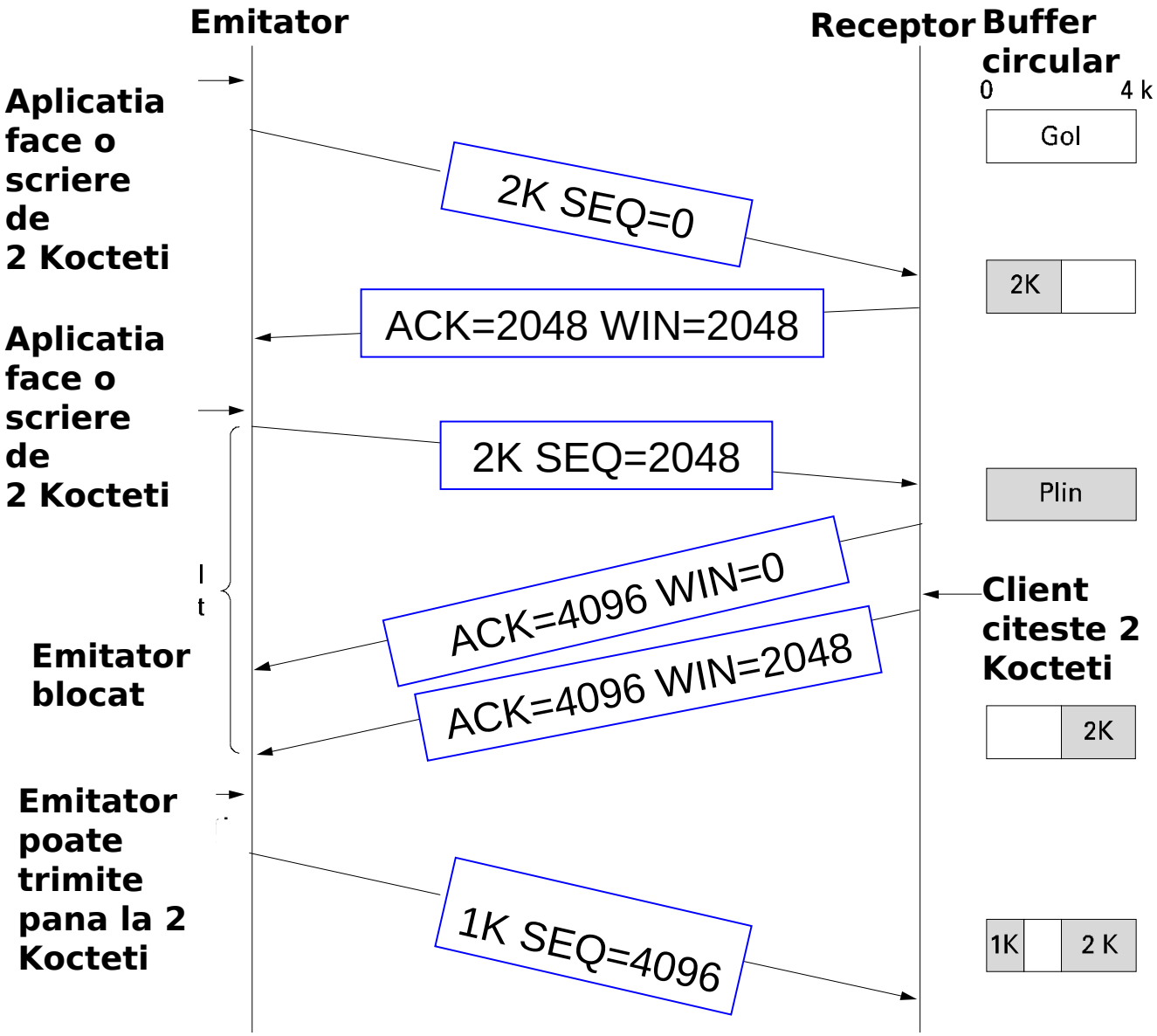
la **receptie**

aduna cuvinte de 16 biti – rezultatul trebuie sa fie zero

2 - Acknowledgement number

Corectia se face prin retransmisie

Controlul fluxului de date



Receptorul specifica fereastra de receptie disponibila

Un emitator blocat poate trimite:

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

Probleme dimensiuni câmpuri antet

Numere secvență de 32-biti

- 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 sa apara 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 μsec

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

Ocupare completă in ambele direcții: produs bandwidth*delay =

= 1 Gbps * 40 ms = 40 milioane biti

Condiții - fereastra receptor ar trebui sa fie >= 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

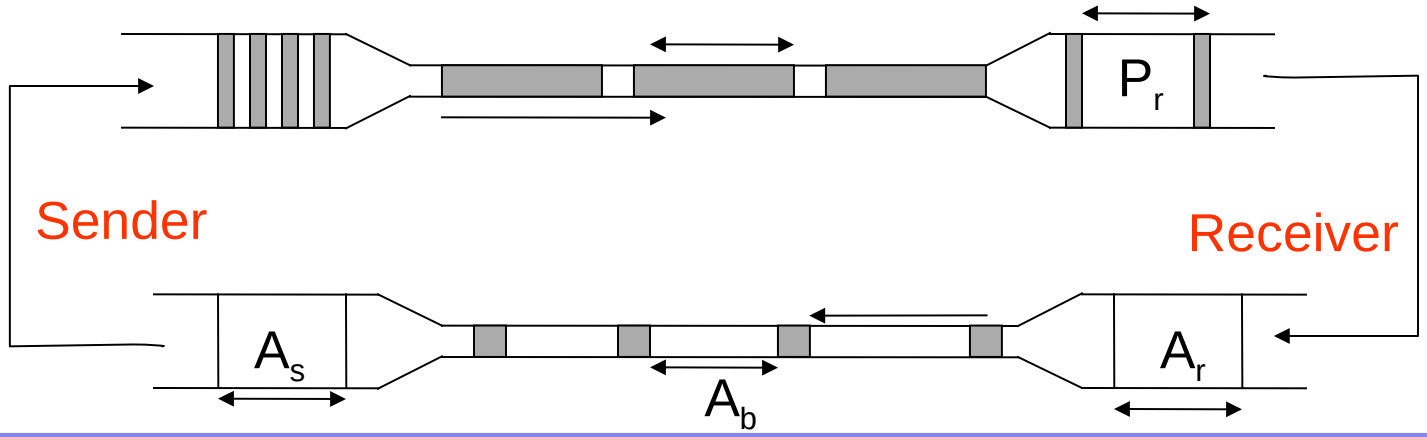
Controlul congestiei

Controlul congestiei

- Fluxul de date transmis pe o conexiune TCP limitat de minimul dintre:
  - dimensiunea fereastrei receptorului - *rwnd*
  - capacitatea retelei (**fereastra de congestie**) - *cwnd*
- Dimensiunea ferestrei glisante ar trebui sa fie *min(rwnd,cwnd)*
- Algoritm de stabilire fereastra de congestie
  - transmite un segment de dimensiune maximă pe conexiunea stabilita
  - 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)

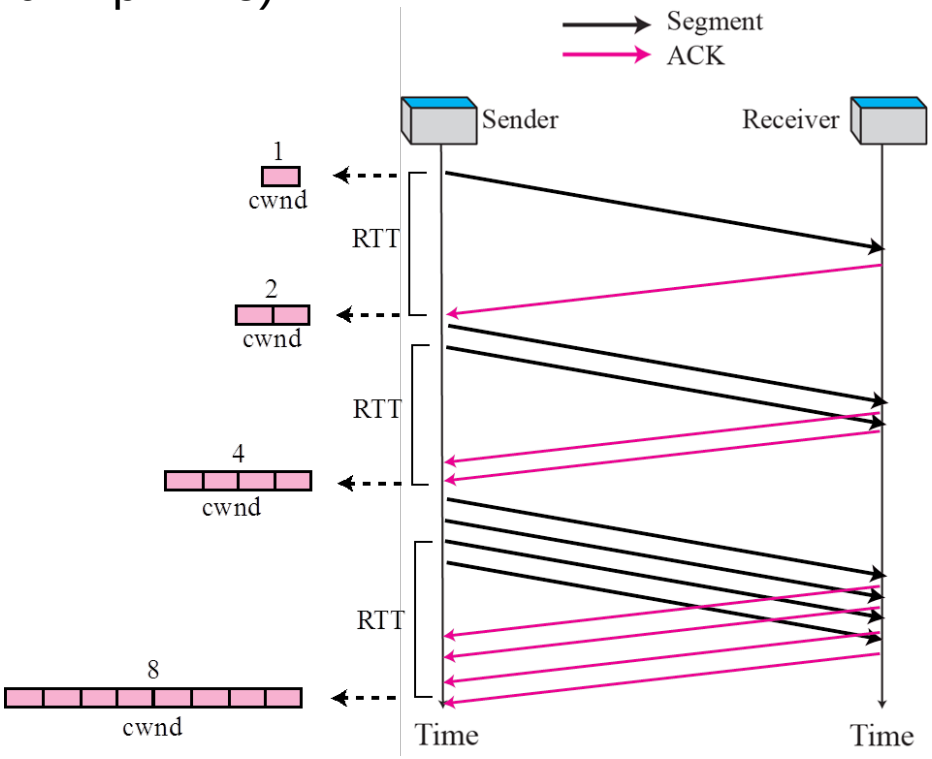
Controlul congestiei

- Algoritmul de control al congestiei
  - foloseste un **prag** (threshold)
  - la un timeout pragul este setat la jumatate 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)



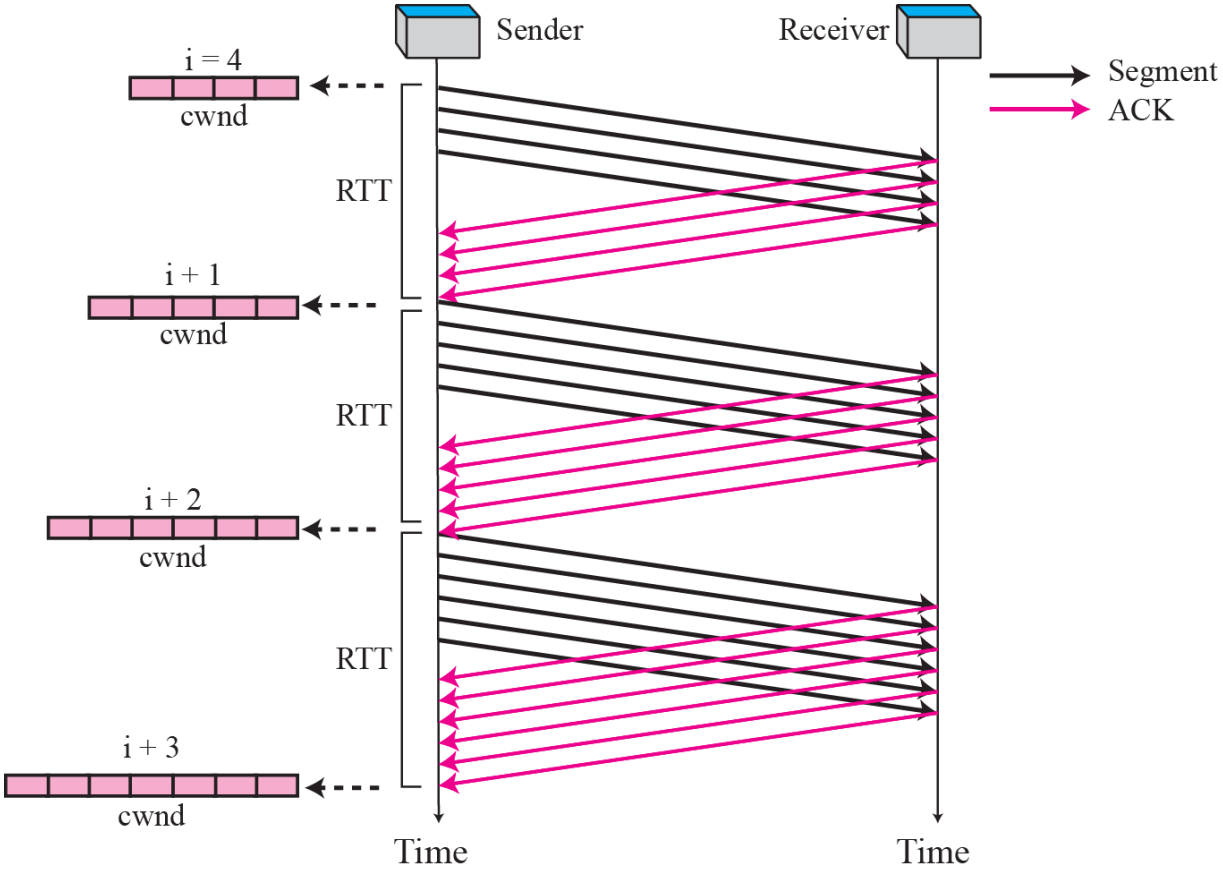
TCP Slow Start

- Se doreste cresterea rapida a ratei de transmisie
- Se incepe cu trimiterea unui singur pachet si fereastra de congestie creste exponential pana la primul pachet pierdut
- Rata de transfer este foarte lenta la inceput dar creste exponential la fiecare RTT (Round Trip Time)



Evitarea congestiei (crestere liniara)

- Cand dimensiunea ferestrei de congestie a ajuns la nivelul pragului pentru “slow start”, dimensiunea ferestrei de congestie va mai creste incremental pana se declanseaza un timeout.

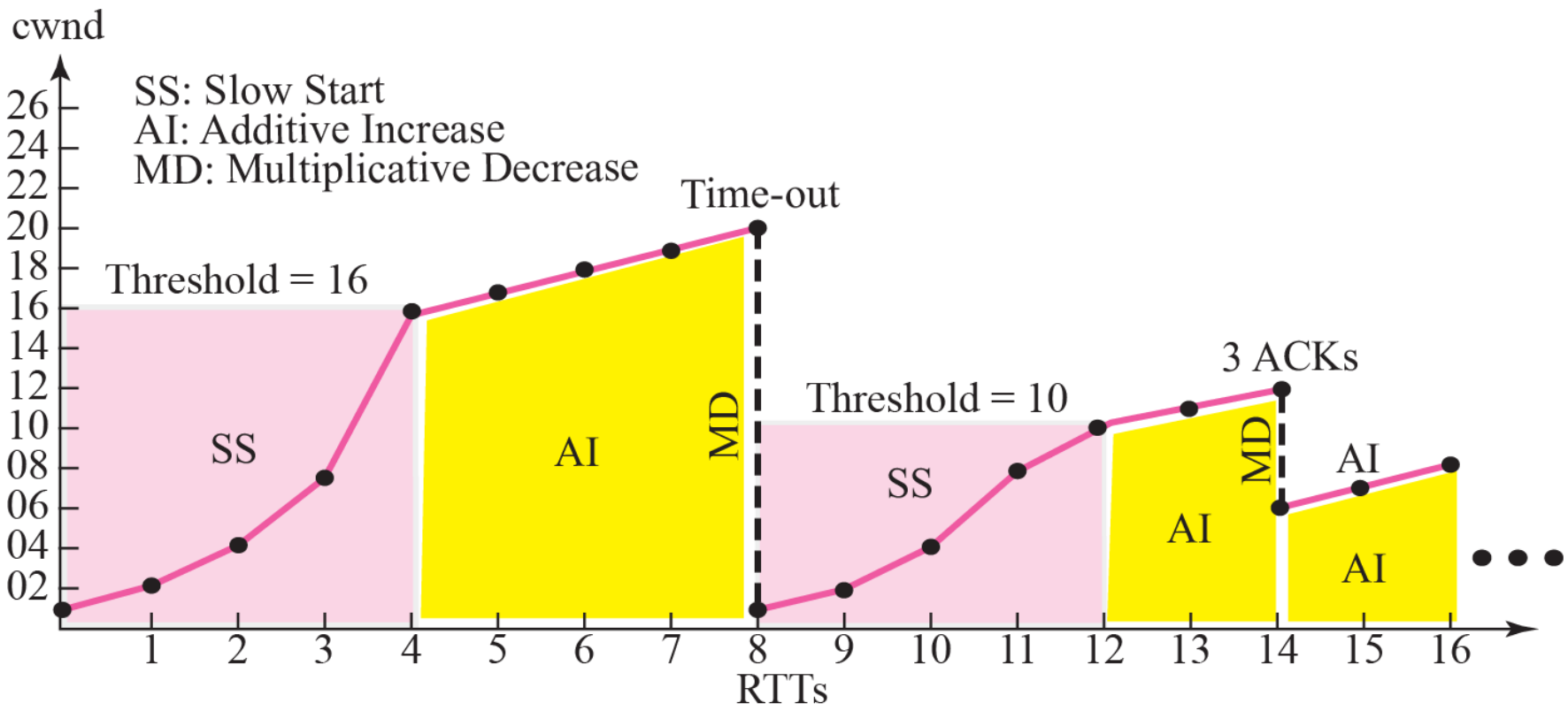


Evitarea congestiei (scadere multiplicativa)

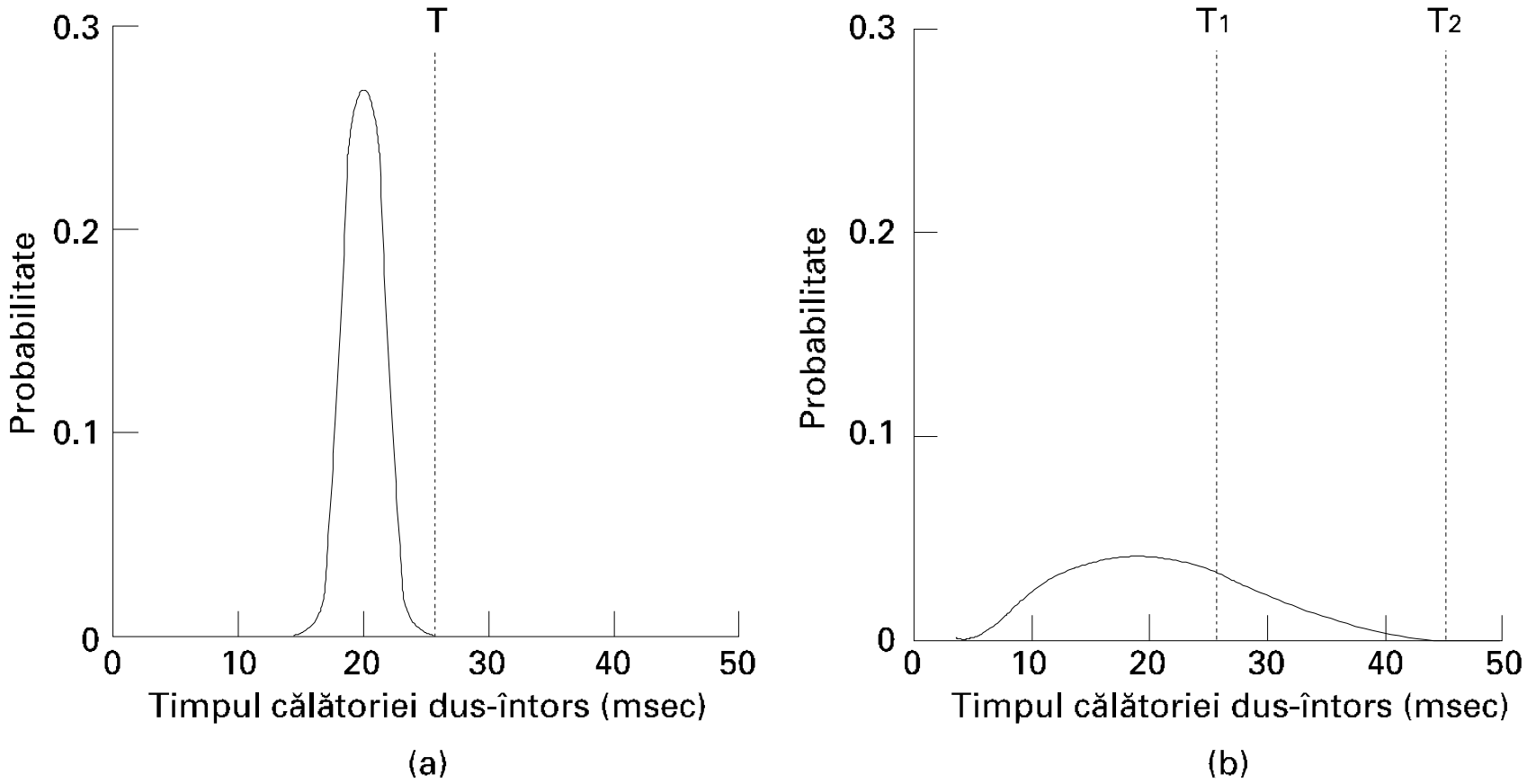
- In cazul in care se detecteaza o congestie pot fi luate una din urmatoarele doua actiuni:
  - Daca a avut loc un **timeout** se reia fereastra de congestie de la valoarea 1 si se reporneste etapa de Slow Start
  - Daca au fost primite 3 **ACK-uri duplicate**, este posibil sa nu fie atat de mare congestia, deci se reduce valoarea ferestrei de congestie la jumatate din valoarea curenta a ferestrei si se porneste etapa de crestere liniara.

Comportament tipic TCP

(The TCP Sawtooth)



Gestiunea ceasurilor in TCP



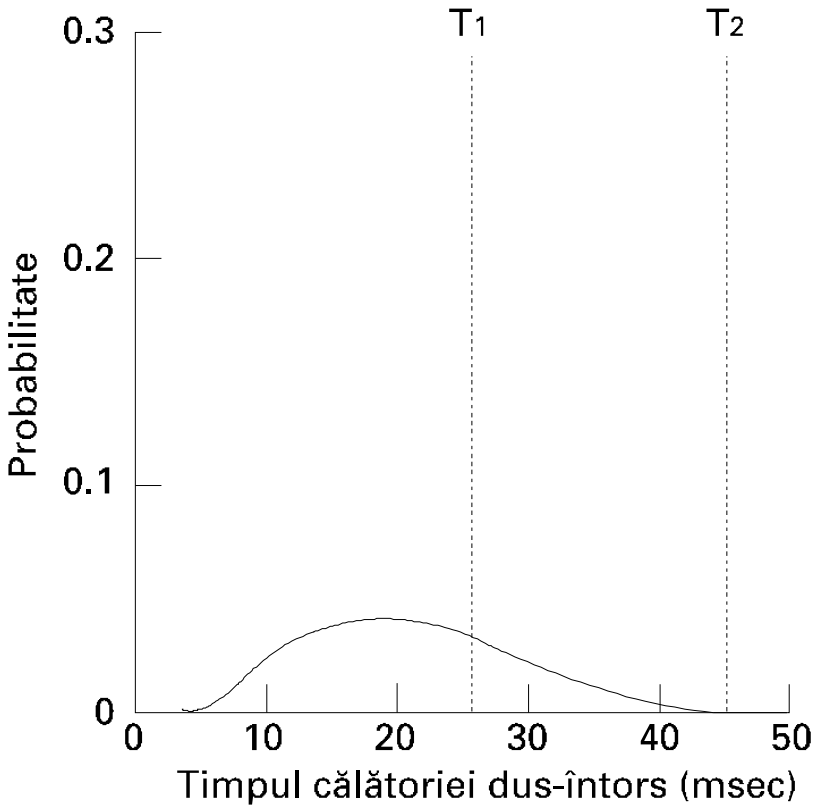
(a) Densitatea de probabilitate a timpilor de sosire ACK in nivelul legatura de date.

(b) Densitatea de probabilitate a timpilor de sosire ACK pentru TCP.

Gestiunea ceasurilor in TCP (2)

Setare proasta – performante slabe:

- Prea lung (T2) – transmitatorul asteapta mult ptr retransmisie
- Prea scurt (T1) – 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 = α*RTT + (1- α)*M cu α = 7/8

  RTO = β*RTT cu β = 2
- Alegere dupa *deviatia standard (DS);*

  D aproximeaza DS

  D = α*D + (1- α)*|RTT-M|

  RTO = RTT + 4*D

  Valoarea lui **α** este dependenta de implementare , dar de obicei este ales 3/4

The Real-Time Transport Protocol

Folosit pentru aplicatii multimedia de timp real

- muzica sau video la cerere, videoconferinte etc.

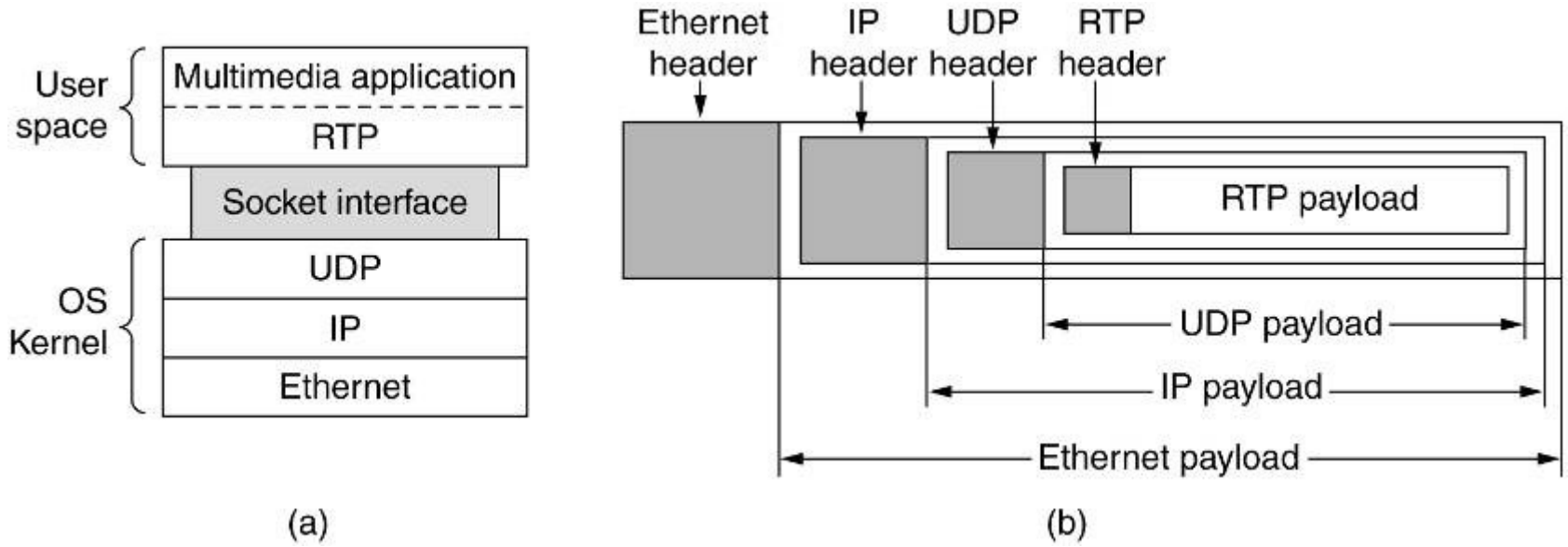
Functie principala: multiplexare fluxuri RTP si transmiterea lor ca un singur șir de pachete UDP

La receptie, RTP livreaza aplicatiei datele multimedia

Fara retransmiterea pachetelor

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

The Real-Time Transport Protocol (2)

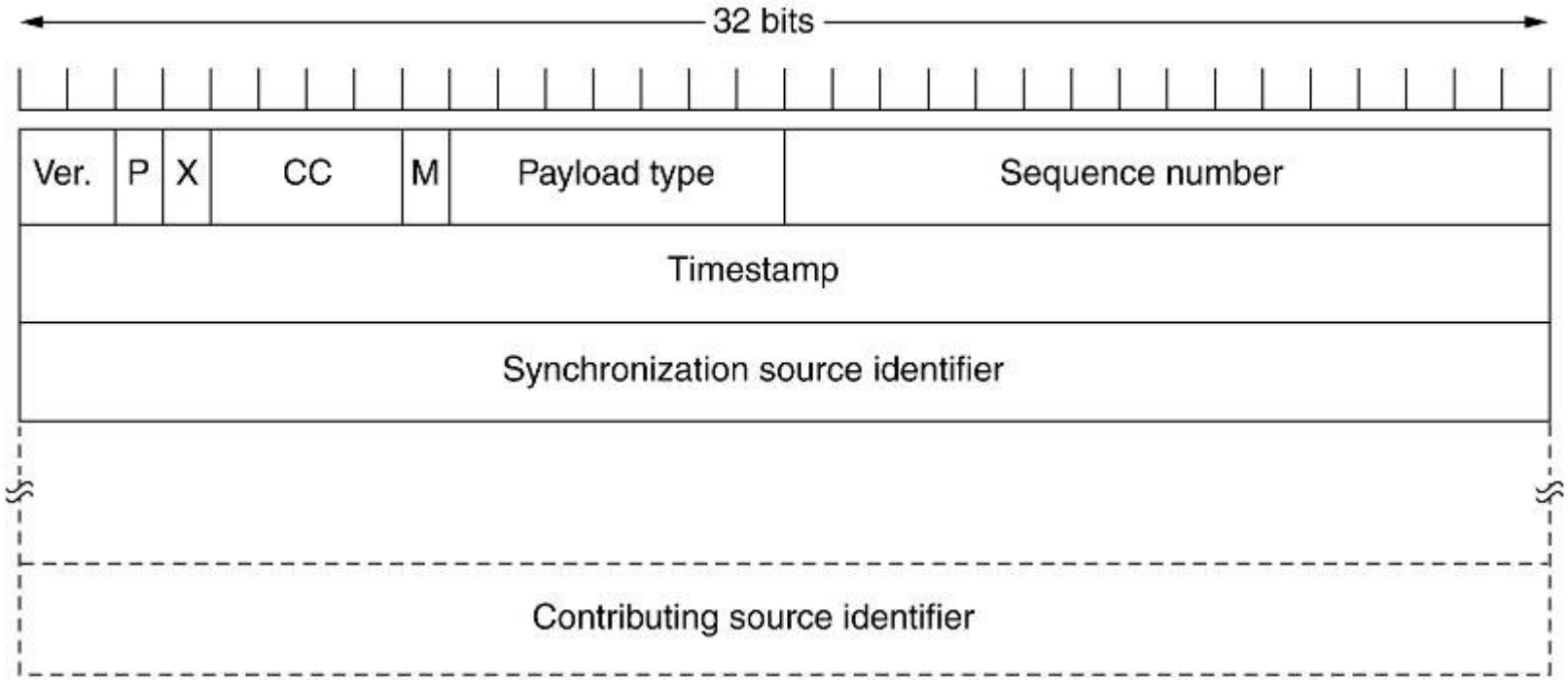


(a) Pozitia RTP in stiva de protocoale. (b) Format pachet.

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

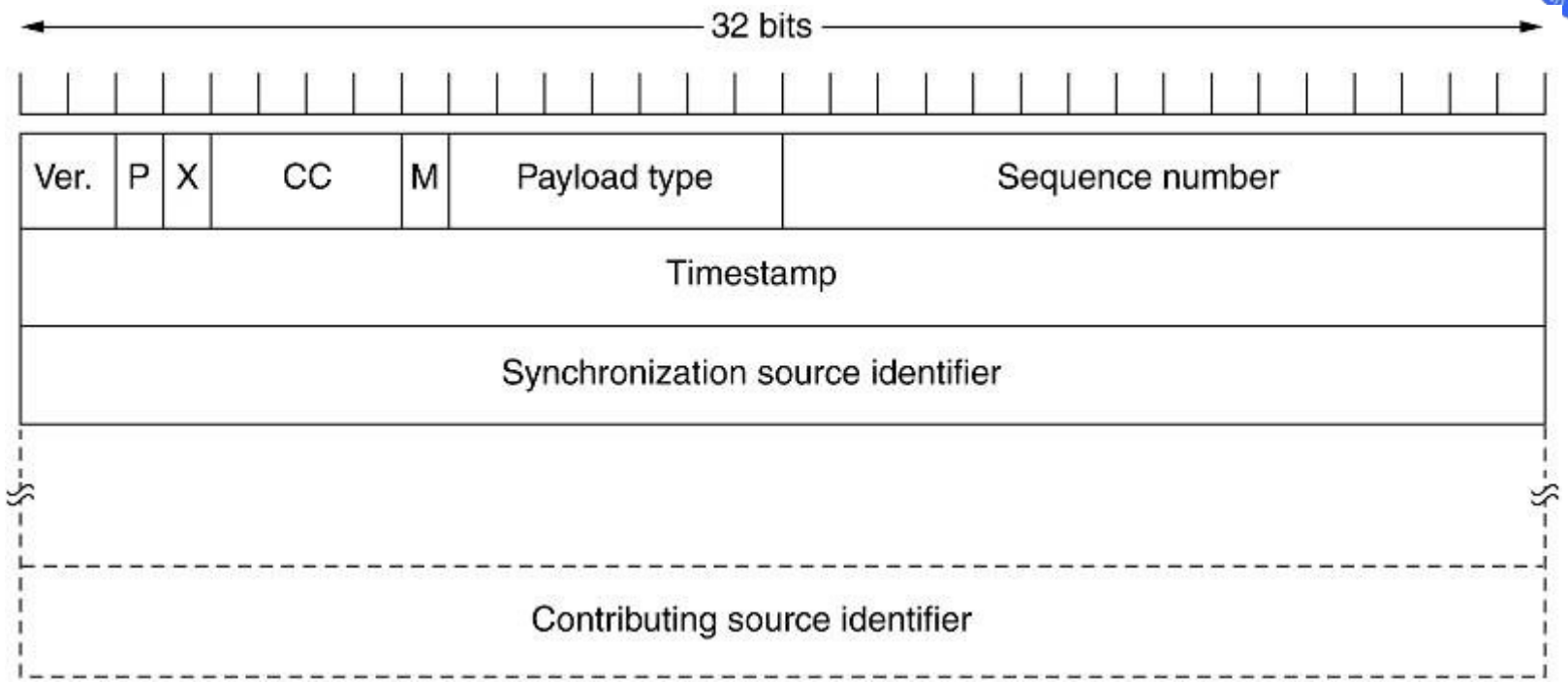
Synchro – identifica sursa de sincronizare (ex. microfon, camera video)

- 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 – nr. antete surse contributoare

M – mark (specific aplicatiei. Ex. start video frame)

Payload type – e.g. MP3

Studiu individual

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

6.1 THE TRANSPORT SERVICE
6.2.1 Addressing
6.2.2 Connection Establishment
6.2.3 Connection Release6.2.4 Error Control and Flow Control
6.3.2 Regulating the Sending Rate
6.4.1 Introduction to UDP
6.4.3 Real-Time Transport Protocols
6.5.3 The TCP Protocol
6.5.4 The TCP Segment Header
6.5.5 TCP Connection Establishment
6.5.6 TCP Connection Release
6.5.7 TCP Connection Management Modeling
6.5.8 TCP Sliding Window
6.5.9 TCP Timer Management
6.5.10 TCP Congestion Control
6.6.4 Fast Segment Processing

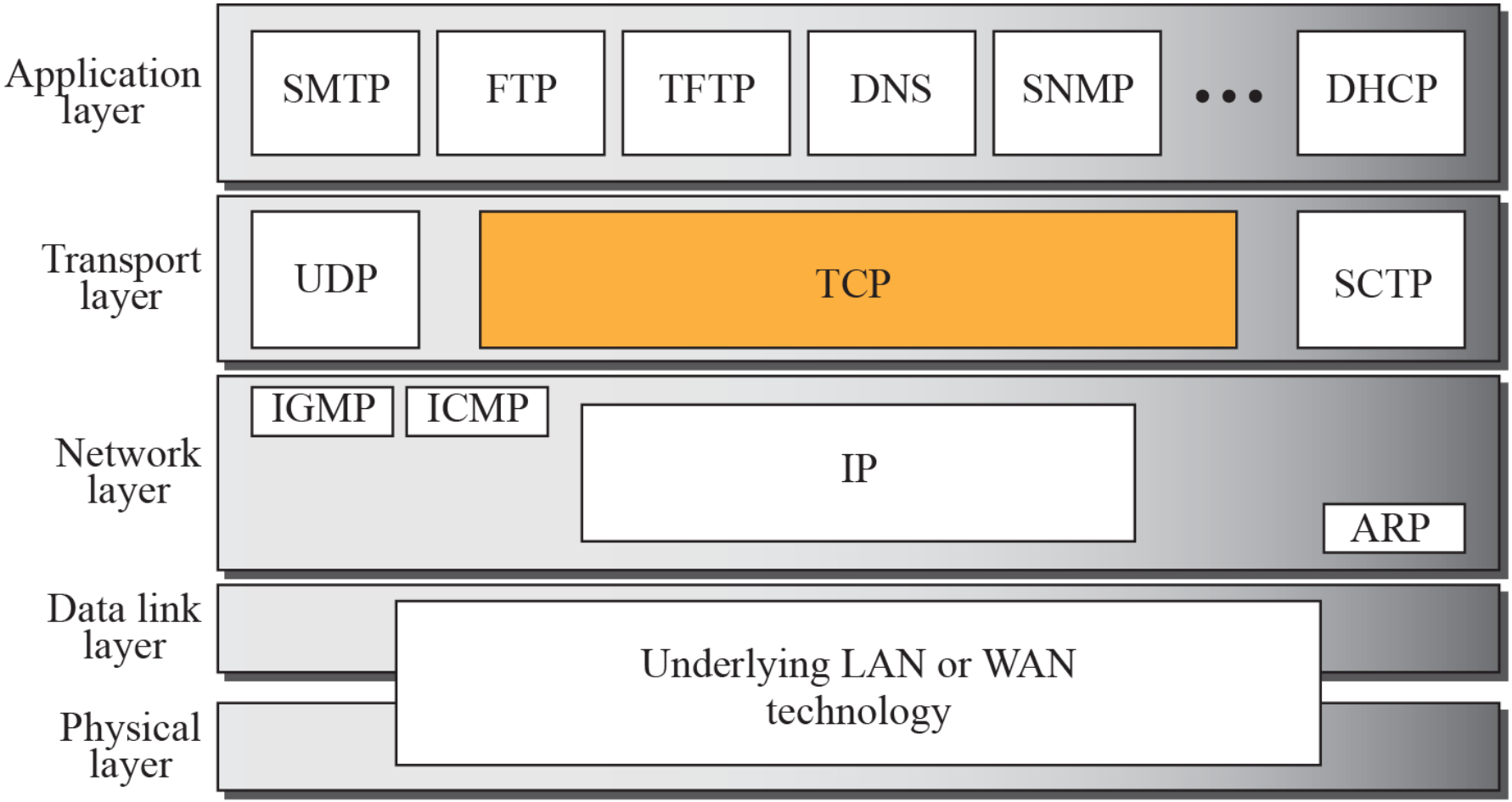
Nivelul Aplicaţie

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
- Rezorvarea recursiva si iterativa a numelor
- Cereri inverse
- Replicarea serverelor DNS

Protocoale de aplicații



De ce este nevoie de DNS ?

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

- Un nume de domeniu este mult mai usor de retinut decat o adresa IP.

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 - trimite o comanda GET indicand adresa IP, portul si calea la fisierul care contine pagina TheProject.html
- Serverul trimite fisierul TheProject.html
- Conexiunea TCP este inchisa
- Browser - afişează conţinutul 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 retinut !

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

Adrese simbolice: nume de domeniu

Continute in URL – Uniform Resource Locator

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

Schema	Utilizat pentru	Exemple
http	Hipertext (HTML)	http://www.cs.vu.nl/~ast
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
mailto	Trimitere de poşta electronică	mailto:JohnUser@acm.org
telnet	Conectare la distanţă	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 - trimite o comanda la server Web

```
GET TheProject.html HTTP/1.1
Host: www.w3.org
```

  - Server Web www.w3.org trimite fisierul TheProject.html
- Conexiunea TCP este inchisa
- Browser - afişează conţinutul din TheProject.html

DNS – The Domain Name System

- Inainte de existenta DNS (1985) asocierea dintre numele unui domeniu si adresa IP era realizata prin downloadarea unui singur fisier *host.txt* de pe un server central prin FTP

- Numele din hosts.txt nu erau structurate

- Fisierul hosts inca exista pe majoritatea sistemelor de operare si poate fi folosit pentru definirea locala de nume

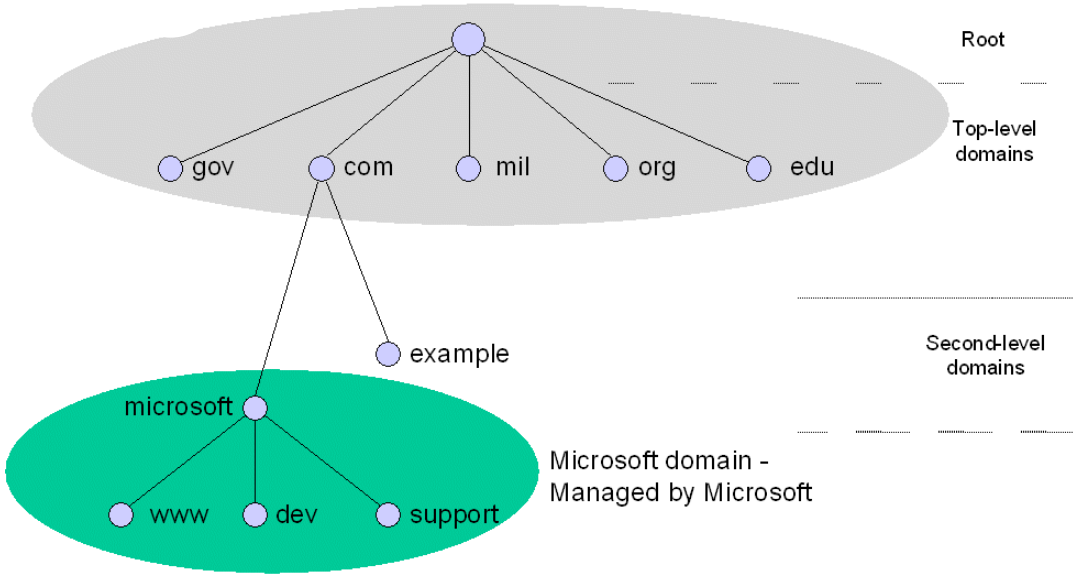
DNS – The Domain Name System

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

  https://www.icann.org/resources/pages/tlds-2012-02-25-en
- De nivel 2 (ex. microsoft.com) … etc.
- **Frunzele** corespund gazdelor



Numele unui domeniu foloseste numele nodurilor pe calea la radacina

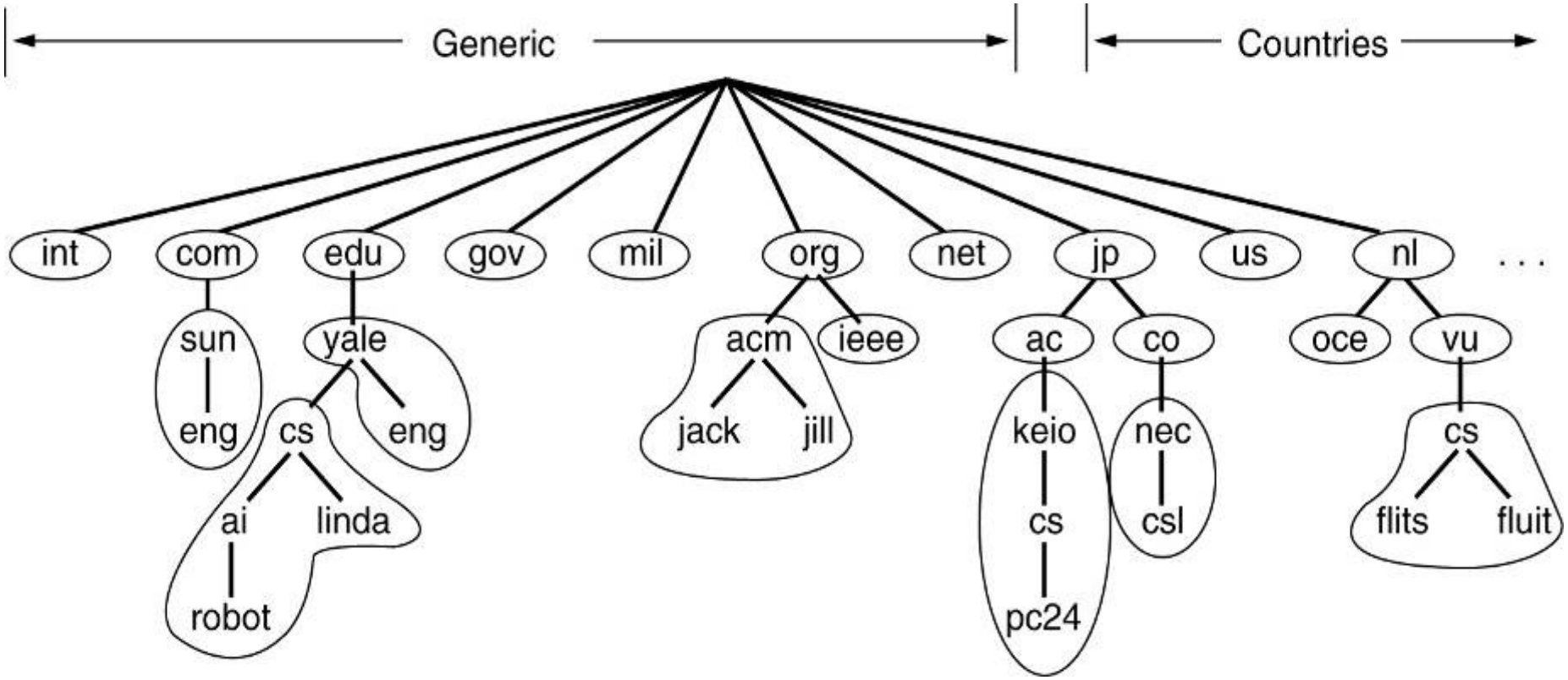
example.com

este diferit de

example.mil

Zone DNS si Servere de Nume

- Spaţiul de nume DNS este împărţit în **zone** administrate de **servere** de nume distincte (mai multe servere pot raspunde de o zona)
- Serverul de nume
  - pastreaza informatii pentru unul sau mai multe domenii
  - cunoaste adresele altor 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.	
Timp_de_viata	ex:	3600	(in secunde)
Clasa	ex:	IN	(pentru Internet)
Tip	ex:	A	(adresa)
Valoare	ex:	157.60.221.205	

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

Principalele tipuri de înregistrări DNS

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

Exemplu Resource Records

O parte a unei baze de date DNS pentru *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

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

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

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

little-sister          IN  A      130.37.62.23

laserjet               IN  A      192.31.231.216
```

info desspre domeniu

2 servere de mail si unul de nume

3 adrese IP si 2 pseudonime pentru Web si FTP

sectiune ptr server flits cu 2 adrese si 3 servere mail

RRs statie de lucru cu 2 adrese mail

Serverul de nume (DNS)

```
   Local Host                                |
+---------+       +----------+               |  +--------+
|         |       |          |responses|     |  |        |
|         |       |   Name   |---------|->|Foreign |
|  Master |------>|  Server  |               |  |Resolver|
|  files  |       |          |<--------|--|        |
|         |       |          | queries |     |  +--------+
+---------+       +----------+               |
```

Informatia primara despre zone este pastrata in fisiere master (Master files) aflate in sistemul de fisiere local serverului DNS

Un server de nume primar

- Foloseste Master files pentru a defini sau actualiza BD pentru una sau mai multe zone
- Raspunde intrebarilor resolverelor

Translatarea de la nume domeniu la adresa IP

```
           Local Host                                |
+---------+                +----------+          |  +--------+
|         | user queries   |          |queries   |  |Foreign |
|  User   |--------------->|          |---------|->|        |
|         | user responses| Resolver |responses|  |  Name  |
| Program |<---------------|          |<--------|--| Server |
+---------+                +----------+          |  +--------+
                              |    ^ references |
                             +----------+        |
                             |  cache   |        |
                             +----------+        |
```

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
  - Comenzile de consola host și dig
- La apelul unui *client*, Resolverul
  - Construieşte un mesaj *DNS request*
  - Transmite mesajul serverului DNS local
- *Serverul* DNS rezolva numele
  - Construieşte un mesaj *DNS reply*
  - Trimite mesajul Resolverului si aşteaptă următoarea cerere

Format mesaje DNS

```
+---------------------+
|        Header       |
+---------------------+
|       Question      |  the question for the name server
+---------------------+
|        Answer       |  RRs answering the question
+---------------------+
|      Authority      |  RRs pointing toward an authority
+---------------------+
|      Additional     |  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)

```
+---------------------+
|        Header       |
+---------------------+
|       Question      |  the question for the name server
+---------------------+
|        Answer       |  RRs answering the question
+---------------------+
|      Authority      |  RRs pointing toward an authority
+---------------------+
|      Additional     |  RRs holding additional information
+----------------------+
```

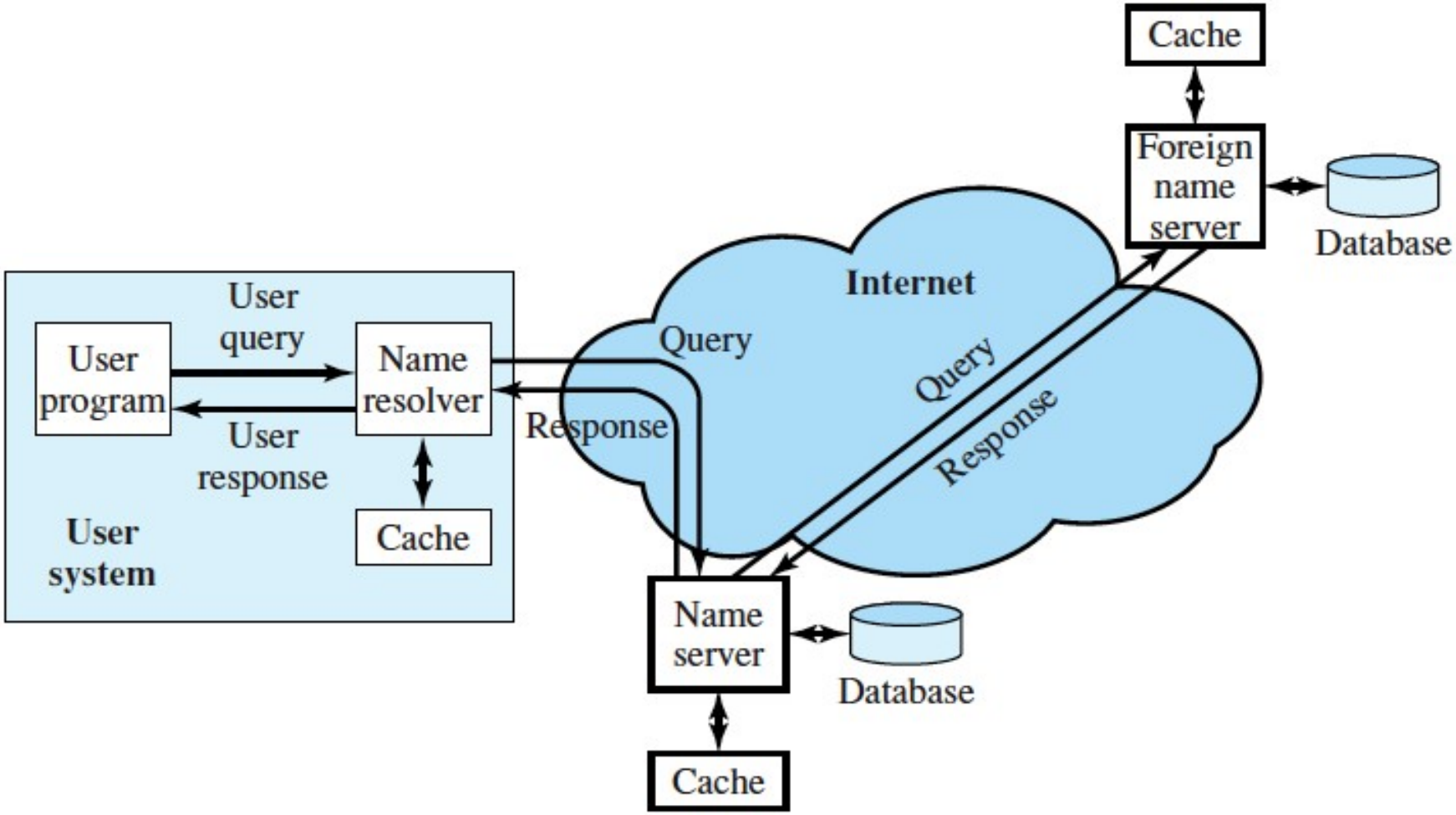
Question – intrebarea

- tuplu Nume-domeniu, tip, clasa
- este singurul camp inclus in intrebare

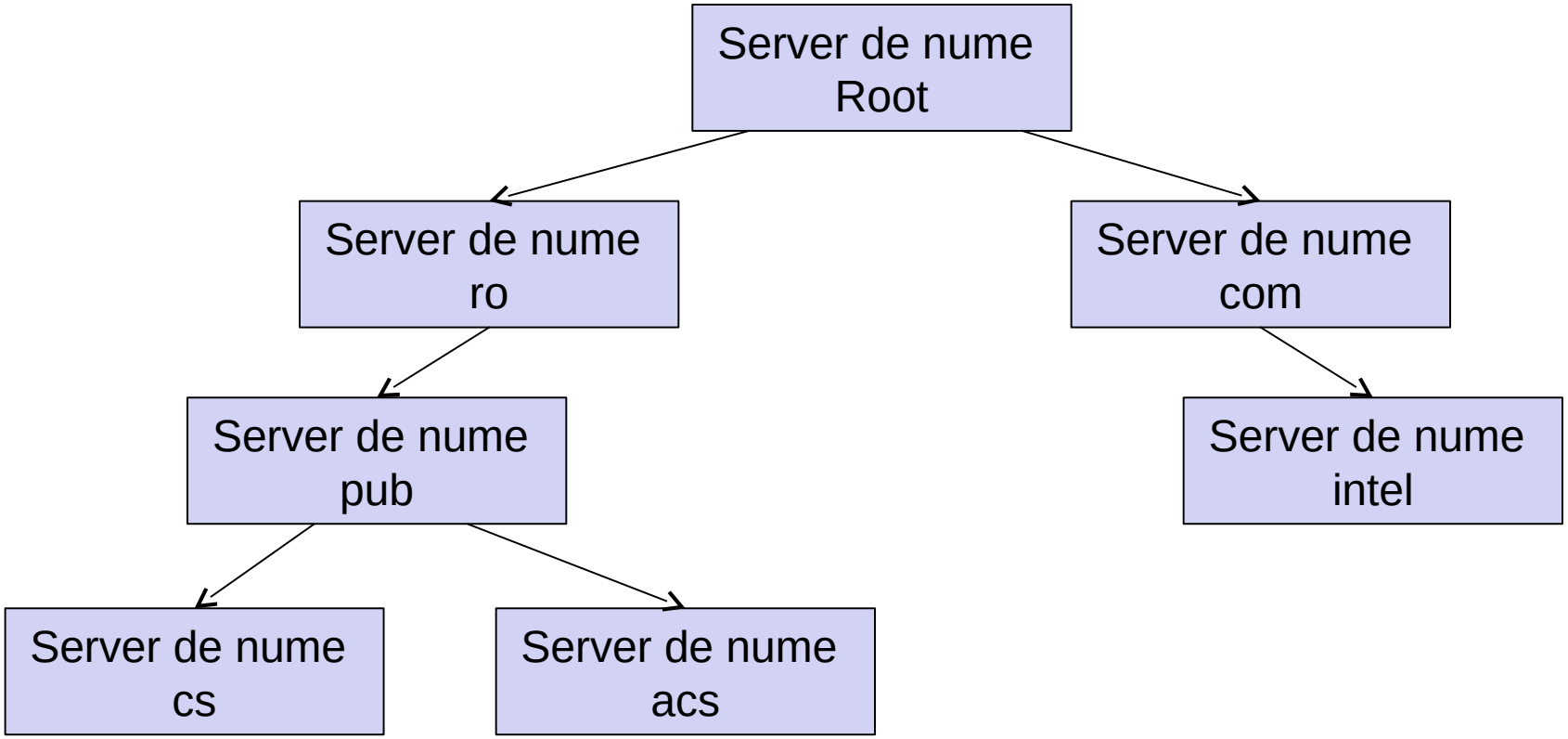
Answer include RRs care corespund intrebarii

Restul – colectie de RRs reprezentand raspunsul, autoritatea si info aditionale

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



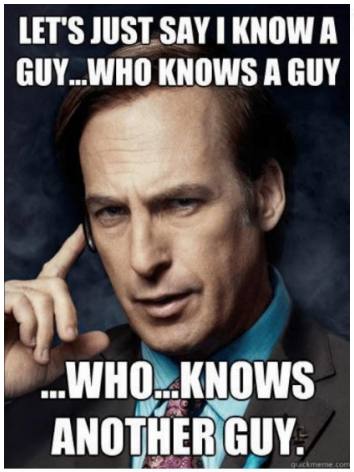
O posibila ierarhie de servere DNS



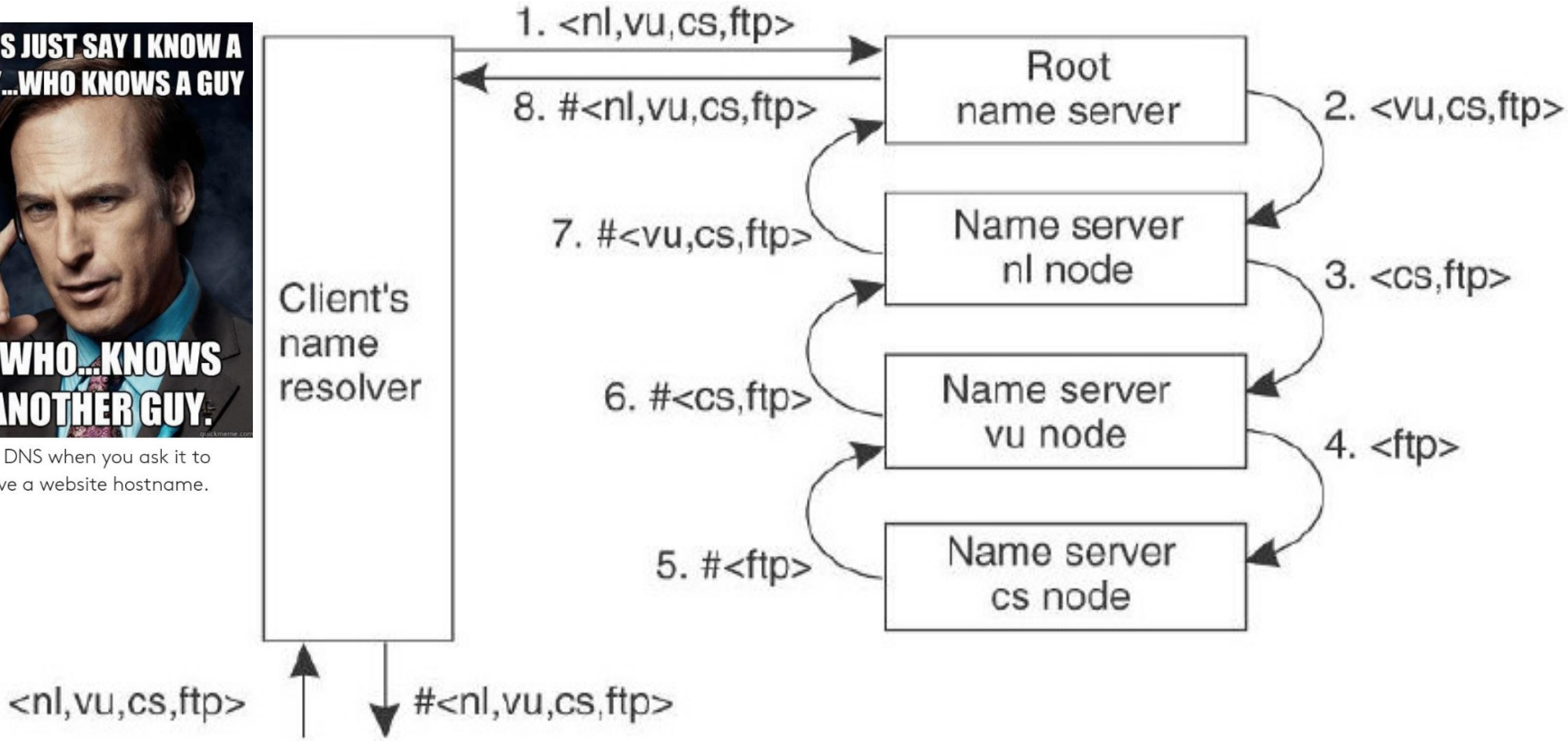
- adresele ptr nume de top (.ro, .nl, com) sunt stiute de root
- exista mai multe servere root, adresele lor IP fiind copiate, din fisiere de config in cache DNS, la pornirea serverului DNS

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



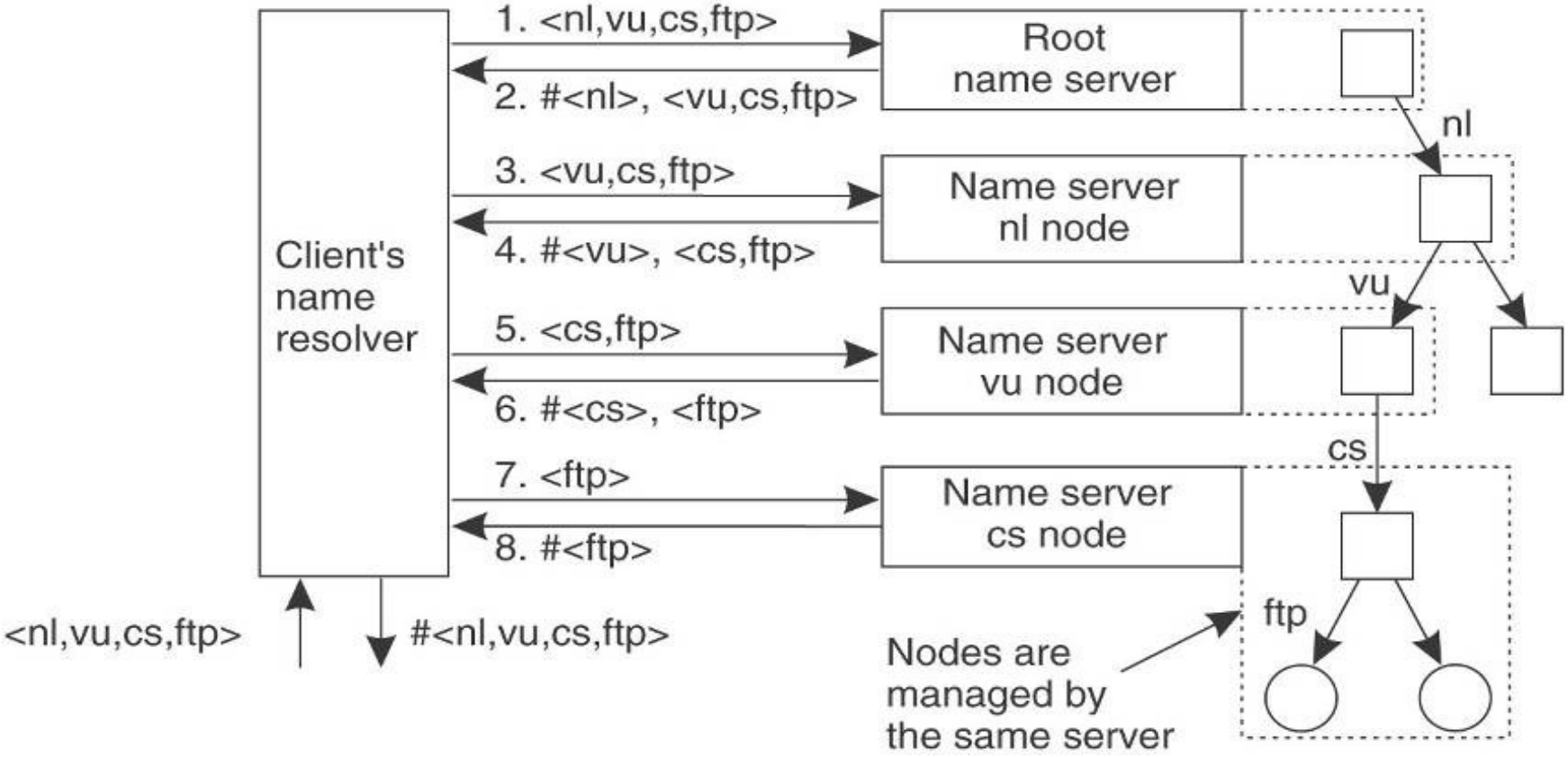
Root DNS when you ask it to resolve a website hostname.



In exemplu: #<ftp> este adresa IP a serverului ftp.cs.vu.nl

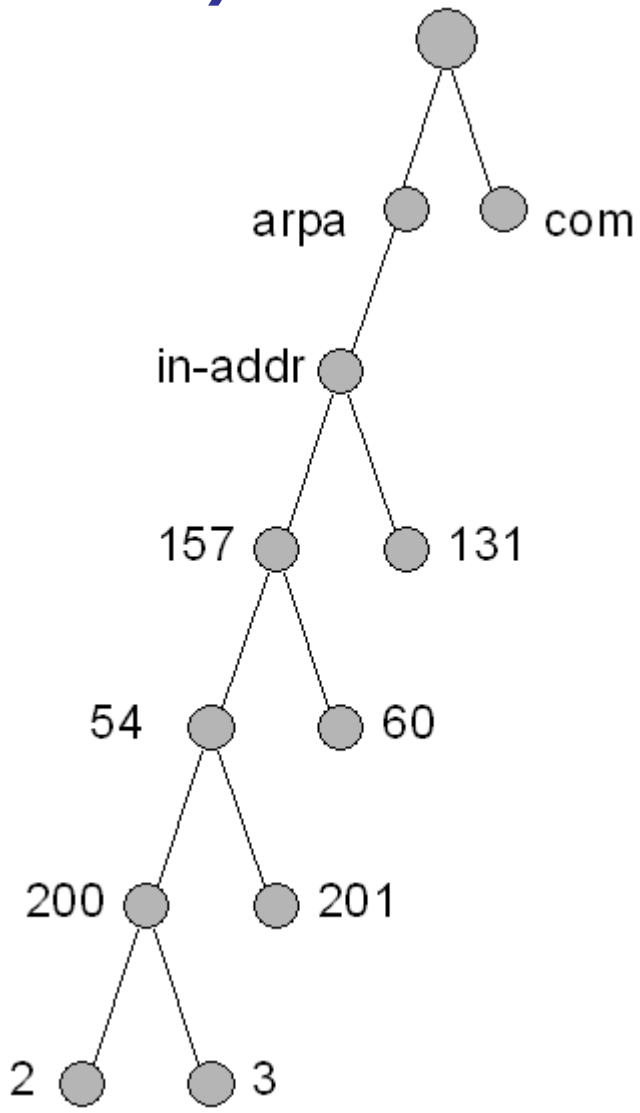
Rezolvare Iterativă

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



Cereri inverse (Reverse DNS)

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



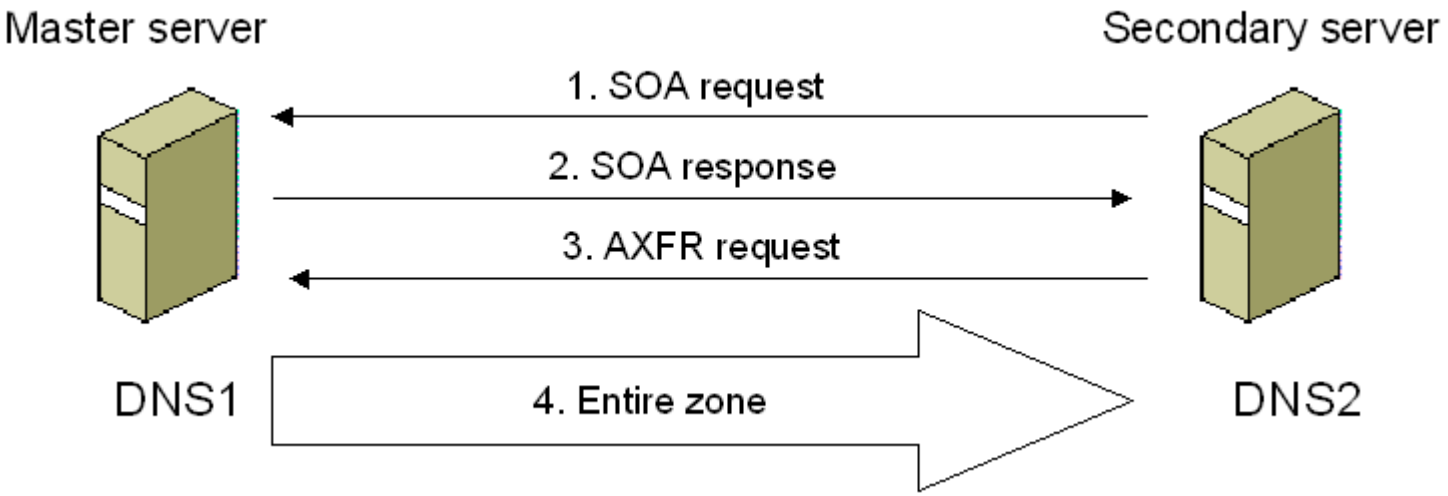
Replicarea serverelor DNS

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

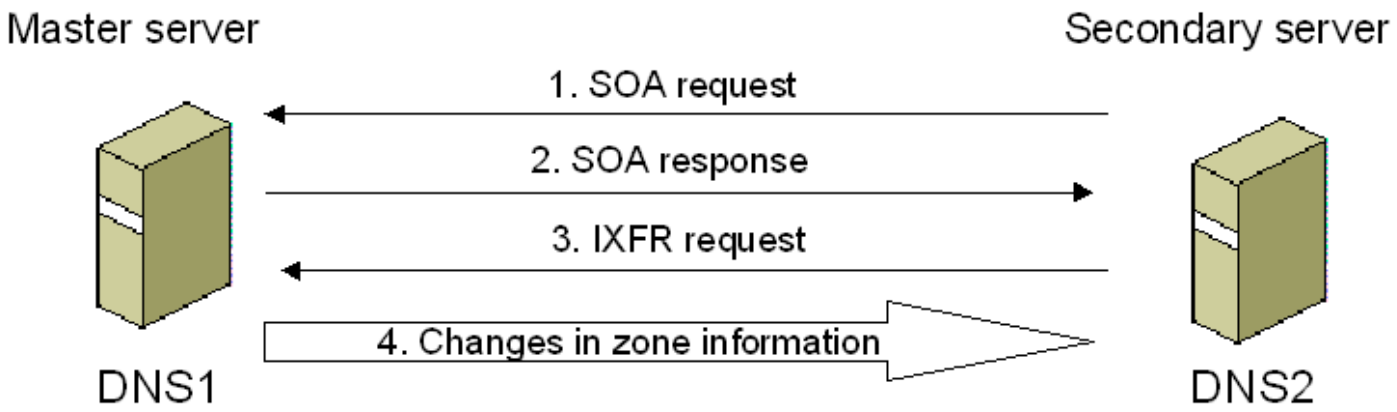
```
          Local Host                                Foreign
                          +----------+         |  +--------+
 +---------+              |          |responses|  |        |
 |         |              |   Name   |---------|->|Foreign |
 |  Master |------>|  Server  | queries |  |Resolver|
 |  files  |              |          |<--------|--|        |
 |         |              +----------+         |  +--------+
 +---------+               ^     |maintenance  |  +--------+
                           |     +-------------|->|Foreign |
                           |      queries      |  | Name   |
                           +-------------------|--| Server |
                         maintenance responses |  +--------+
```

Facilitati – transfer toata zona

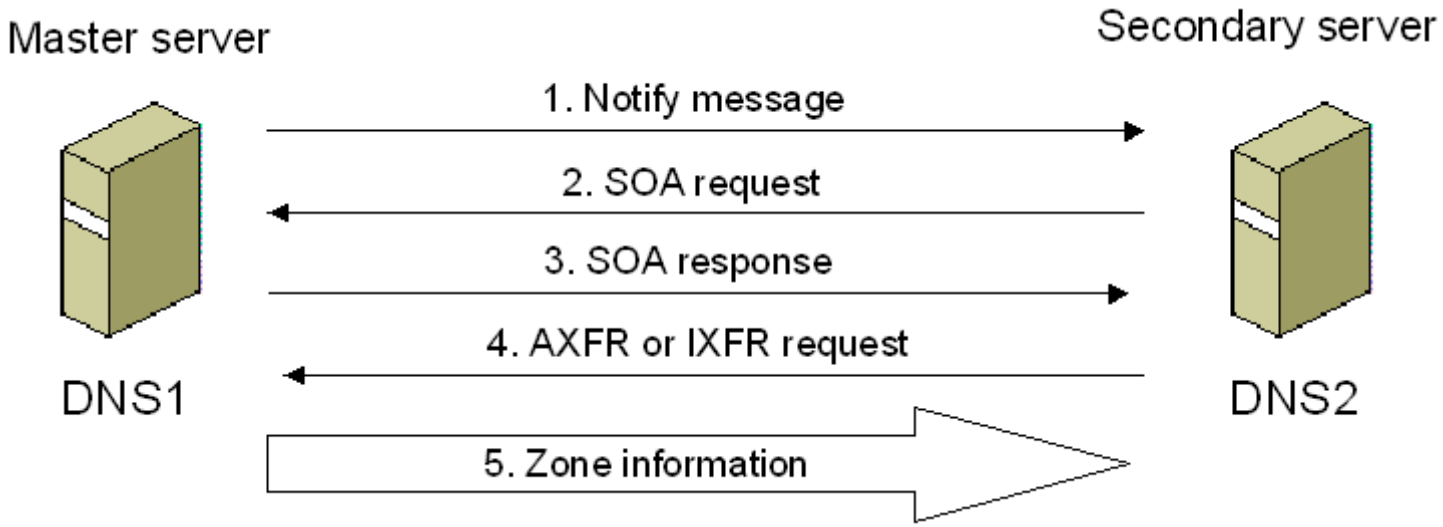
- Server secundar
  - (periodic) Cerere SOA (Start Of Authority)
  - Primeste raspuns si verifica daca "serial number" este mai mare decat cel local
  - Daca da, cere toata zona (cerere AXFR – Authoritative transfer)
  - Primeste info toata zona



Transfer incremental (Incremental Zone Transfer)



Notificari



Studiu individual

A. S. Tanenbaum Reţele 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

END

Nivelul Aplicaţie

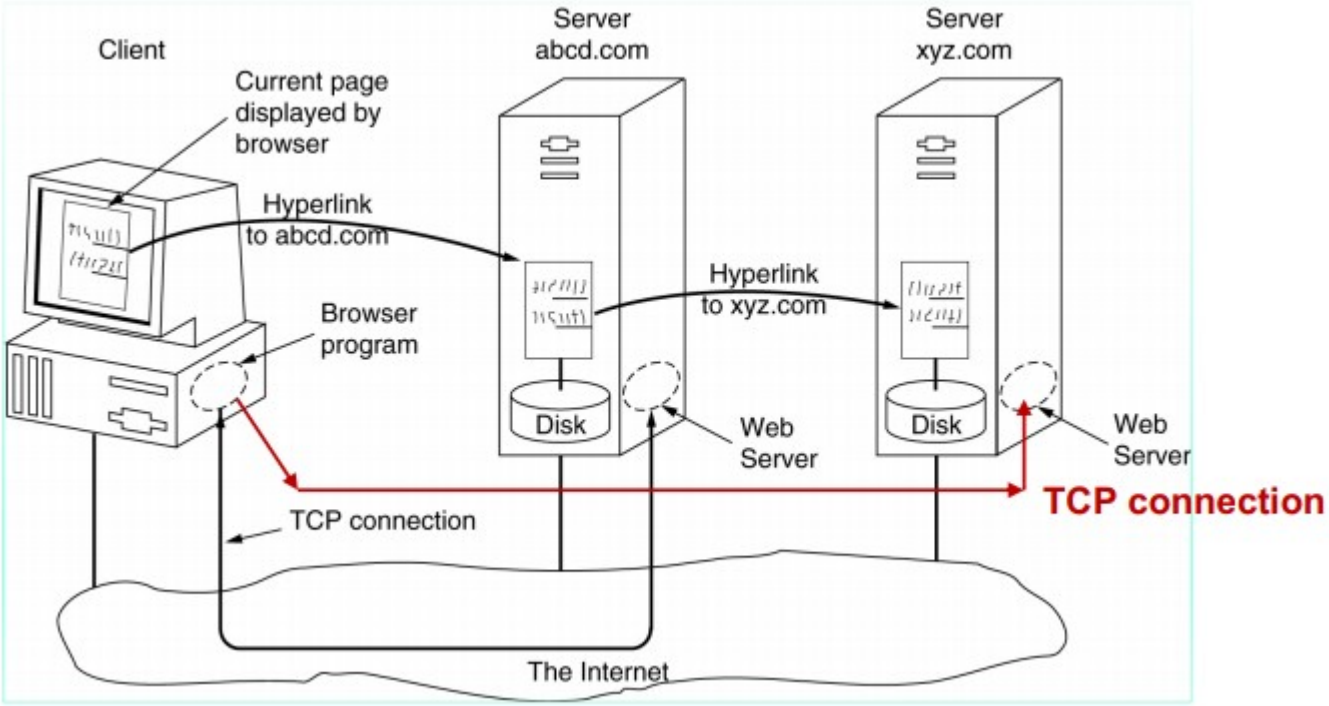
World Wide Web

Cuprins

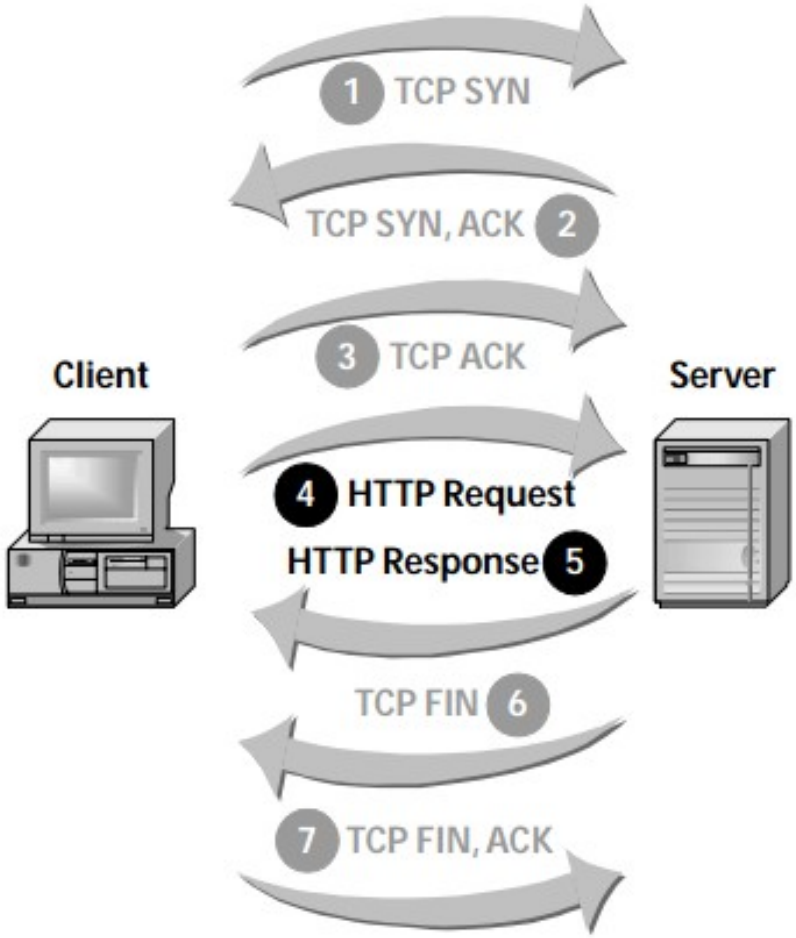
- Functionarea WWW
- URL
- HTML – marcaje, formulare
- HTTP
- Clientul (Browser)
- Serverul

World Wide Web

- Set de documente (pagini) cu legături intre ele (hyperlinks)
- Distribuite pe maşini diferite
- Include o pagina de referinţa (home page)
  - pagina initiala a unui site Web
  - pagina afisata la pornirea unui browser



World Wide Web



◀ **Figure 2.2**
Before systems can exchange HTTP messages, they must establish a TCP connection. Steps 1, 2, and 3 in this example show the connection establishment. Once the TCP connection is available, the client sends the server an HTTP request. The final two steps, 6 and 7, show the closing of the TCP connection.

Interacţiunea client – server

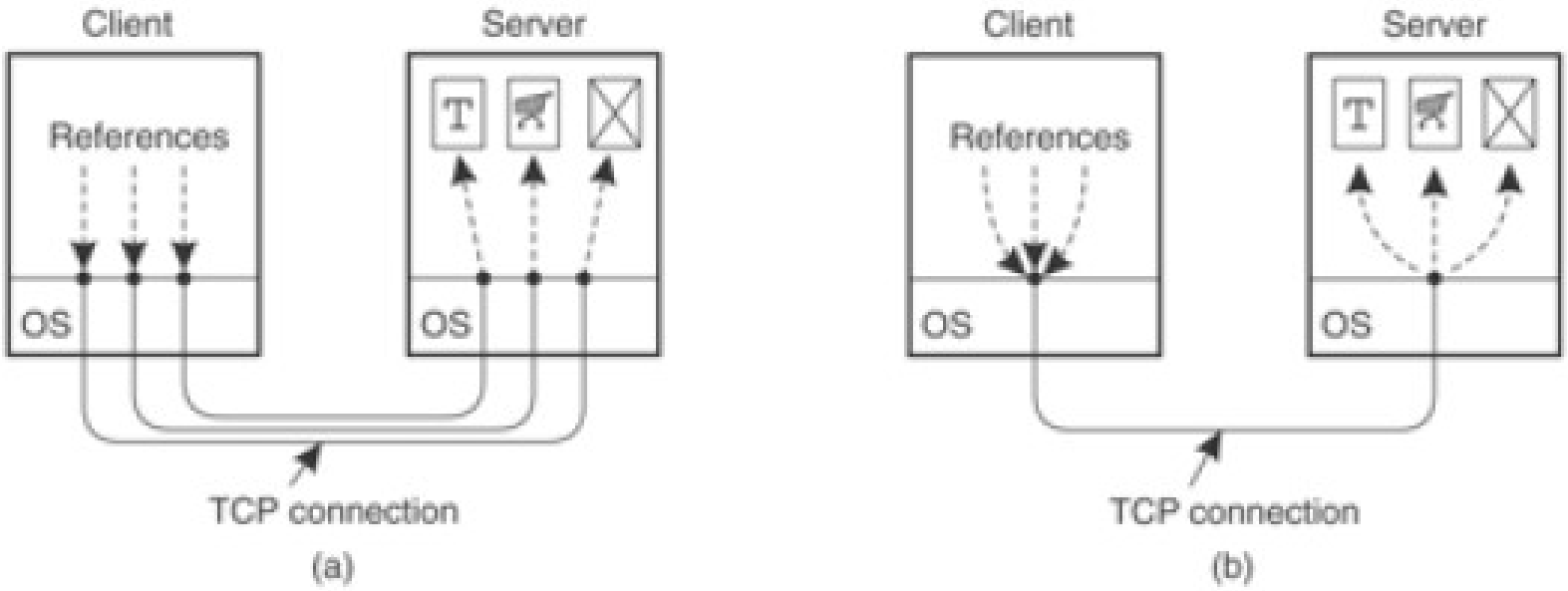
- Browser - determina URL
- Browser - cere DNS-ului adresa IP pentru www.w3.org
  - DNS - raspunde cu 18.23.0.23
- Browser - deschide o conexiune TCP la port 80 pe 18.23.0.23
- Browser - trimite o comanda

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

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

Conexiuni persistente

- Introduse in HTTP 1.1
- O singura conexiune persistenta poate fi folosita pentru mai multe cereri-raspunsuri
- Cererile pot fi transmise si in “pipeline” (fara a astepta 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 specializate prin retea (HTTP – HyperText Transfer Protocol)

URL – Uniform Resource Locator

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

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

exemplu

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

Câteva URL-uri obişnuite

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

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 selecţie de marcaje uzuale

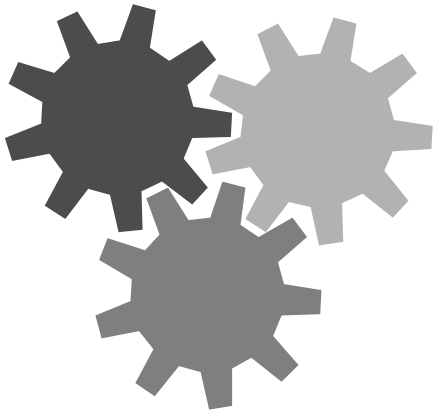
Marcaj	Descriere
<html> ... </html>	Delimitează textul scris în HTML
<head> ... </head>	Delimitează zona de antet
<title> ... </title>	Defineşte titlul (nu este afişat de programul de navigare)
<body> ... </body>	Delimitează zona de corp
<h*n*> ... </h*n*>	Delimitează un titlu de nivel *n*
...	Text îngroşat
...	Text cursiv
...	Centrat pe orizontală
 	Trecere la linie nouă
<p>	Început de paragraf
<ul> ... </ul>	Delimitează o listă neordonată
<ol> ... </ol>	Delimitează o listă ordonată (numerotată)
<li> ... </li>	Delimitează un element într-o listă ordonată sau neordonată
<hr>	Linie orizontală
<img src="URL" ALT="text">	Afişează 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	Semnificaţie
<INPUT>, TYPE=text	NAME, SIZE, MAXLENGTH	câmp de intrare (tipul implicit)
<INPUT>, TYPE=radio	NAME, VALUE	buton radio
<INPUT>, TYPE=checkbox	NAME, CHECKED	casetã de selecţie
<INPUT>, TYPE=password	NAME, SIZE, MAXLENGTH	câmp de parolã
<INPUT>, TYPE=reset sau submit		buton de acţiune
<INPUT>, TYPE=image	NAME, ALIGN, SRC	hartã (imagine) activã
<INPUT>, TYPE=hidden	NAME,	element ascuns
<SELECT>	NAME, OPTION, MULTIPLE	listã de selecţie
<TEXTAREA>	NAME, COLS, ROWS, WRAP	zonã 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

Submit order

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

Formulare

Un text cu informaţiile completate de utilizator

Widget Order Form

Name John Doe

Street address 100 Main St.

City White Plain State NY Country USA

Credit card # 1234567890 Expires 6/98 M/C ◉ Visa ○

Widget size Big ○ Little ◉ Ship by express courier ◉

Submit order

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

Datele introduse de utilizator → lista de elemente “nume=valoare”

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

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

HTTP

- Protocol "stateless"
- Foloseste paradigma request/response
  - clientul si serverul comunica direct sau prin proxy-uri
  - structura mesajelor:
    - linia de comanda / raspuns
    - 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	Ştergerea unei pagini de Web
TRACE	Transmite in ecou cererea care a sosit
OPTIONS	Interogarea anumitor opţiuni
CONNECT	Folosit ptr 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: **YHOO** **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! financiar - YHOO</TITLE></HEAD>
<BODY>
…
</BODY>
</HTML>
```

Exemplu POST

- Aceeasi cerere, formulata cu metoda POST

```
POST /q  HTTP/1.1
Host: financiar.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

Cod	Semnificaţie	Exemple
1xx	Informaţie	100 = serverul acceptă continuarea tratarii cererii de la client (asociat cu un antet Expect din cerere)
2xx	Succes	200 = cerere reuşită; 204 = nu există conţinut
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	Informaţie asupra programului de navigare şi 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	Informaţie despre server (ex. Server: Apache/1.2.5)

Antete Mesaje HTTP (2)

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

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 dacat 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
    - Exista expirata - adauga la cerere antet If-Modified-Since
      - daca server raspunde cu 304 Not Modified foloseste intrarea din cache
    - Exista 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
- **Secventa 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/**

depinde de **http://cs.pub.ro/~popescu/**

care este un prefix al primului

Suport sesiune

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

- ex. info stare - identificatorul unei sesiuni

- Intelegerea este initiata 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 inapoi de browser

path, domain identifica 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 sa transmita info pe legatura securizata

Suport sesiune (2)

- Intelegerea 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

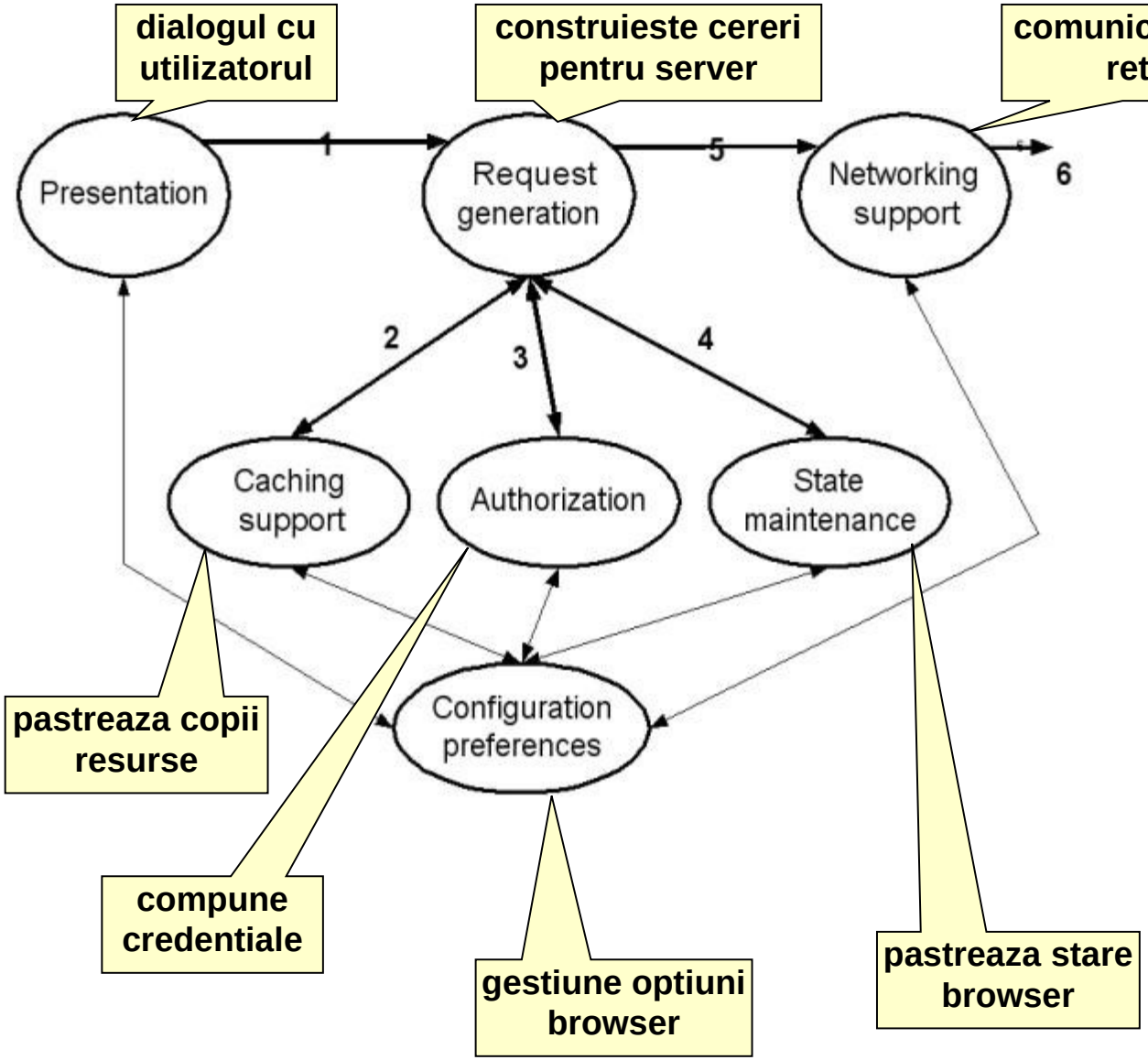
Set-Cookie: client=Ion; path=/carte/capitol3/; domain=.pub.edu

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

Interfata utilizator

- Afiseaza pe ecran rezultatul primit de la modulul Interpretare continut (din grupul de procesare a raspunsului)
- Executa actiunile 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 optiuni configurare
    - **Nu descarca imagini referite in pagina HTML**
    - **Rejecteaza cookies**
- Paseaza informatia 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 doua 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 in 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**

– Construieste linia de cerere, care are 3 componente

METODA

Implicit (la activare hyperlink) GET

In formular (specificat explicit) GET sau POST

/cale-resursa

Numai calea in HTTP/1.1

Tot URL in HTTP/1.0

HTTP/versiune

– Construieste antetele de baza

Host: 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 in care se afla 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

- Construieste corp cerere
  - se aplica pentru POST, PUT
  - POST
    - parametrii din formulare in corp comanda

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 6

s=YHOO
```

- PUT sau POST
  - folosind MIME

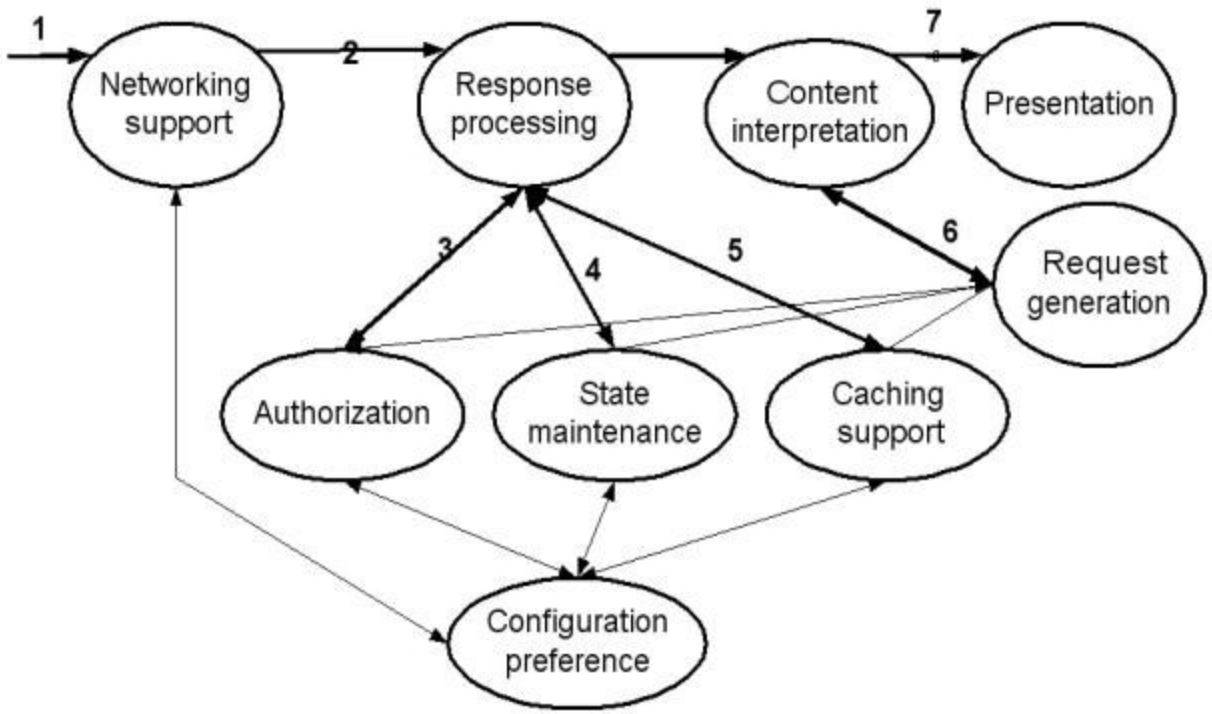
```
Content-Type: multipart/multipart_subtype;
  boundary="ThisRandomString”

--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

- Transmite 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 transmite cereri din coada
    - transmite 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
```

  - 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

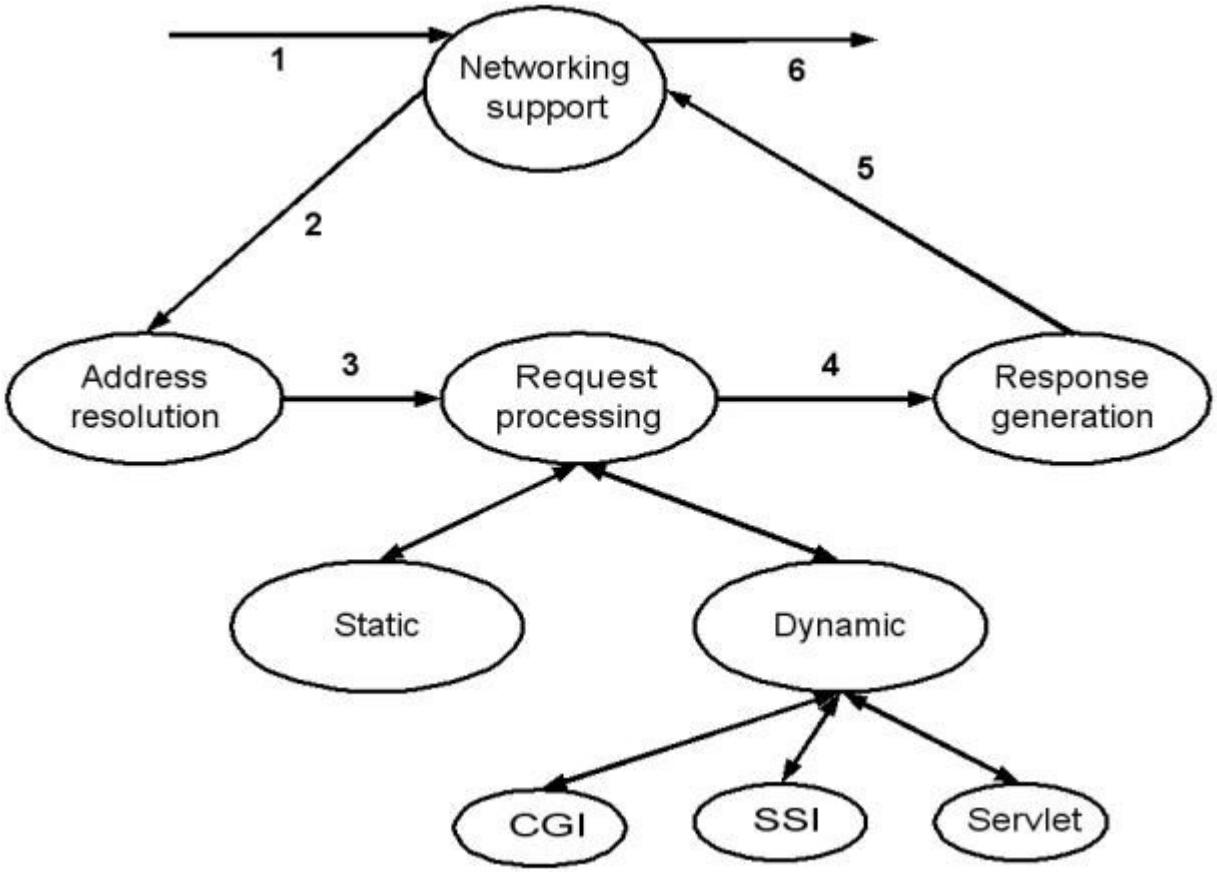
```
HTTP/1.1 200 OK
.  .  .
Cache-Control: private
Expires: Tue, 06 Feb 2005 04:33:20 GMT
Last-Modified: Mon, 05 Feb 2005 04:33:18 GMT
…
```

- 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

Operaţii Server



1. serverul primeste 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

4. Rezultat pasat generatorului de raspuns
5. Pasat modulului suport retea
6. Transmite clientului

Rezolvarea adresei

- selecteaza virtual host
  - nu exista antet Host: -> eroare 400 Bad request
  - exista -> determina domeniul
    -> determina parametrii de config. logica (proprii 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 statica** – 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
        /cgi-bin/ target = CGI script
      - **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

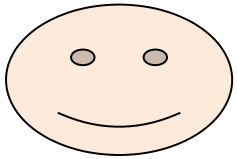
A. S. Tanenbaum Reţele de calculatoare, ed 4-a, BYBLOS 2003

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

END



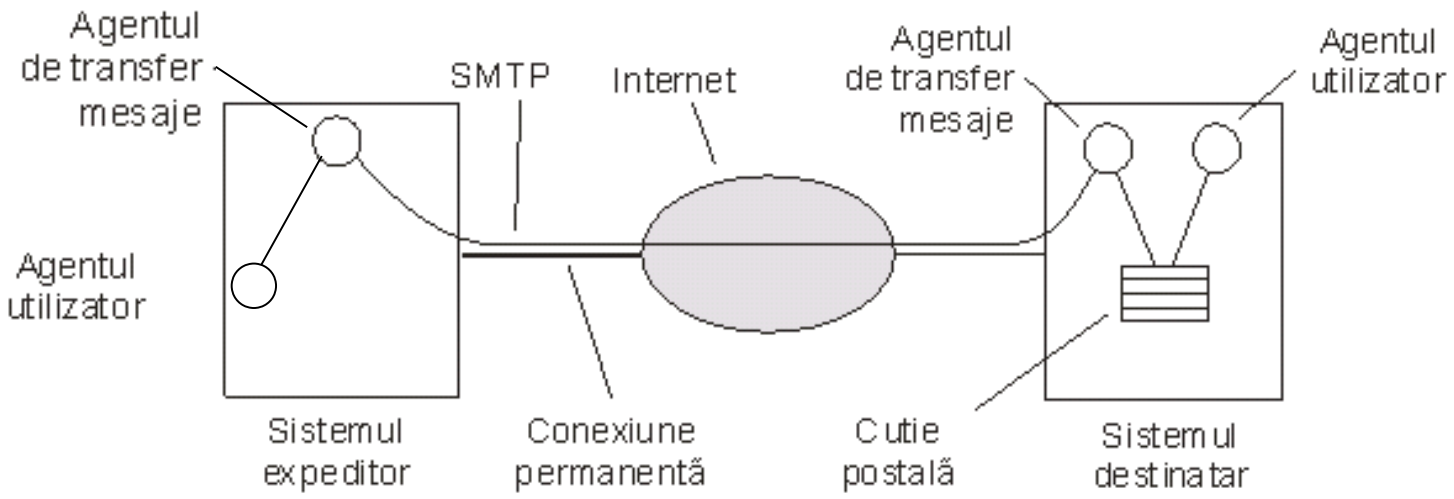
Nivelul Aplicaţie

Poşta electronică (SMTP, POP3, IMAP)

Cuprins

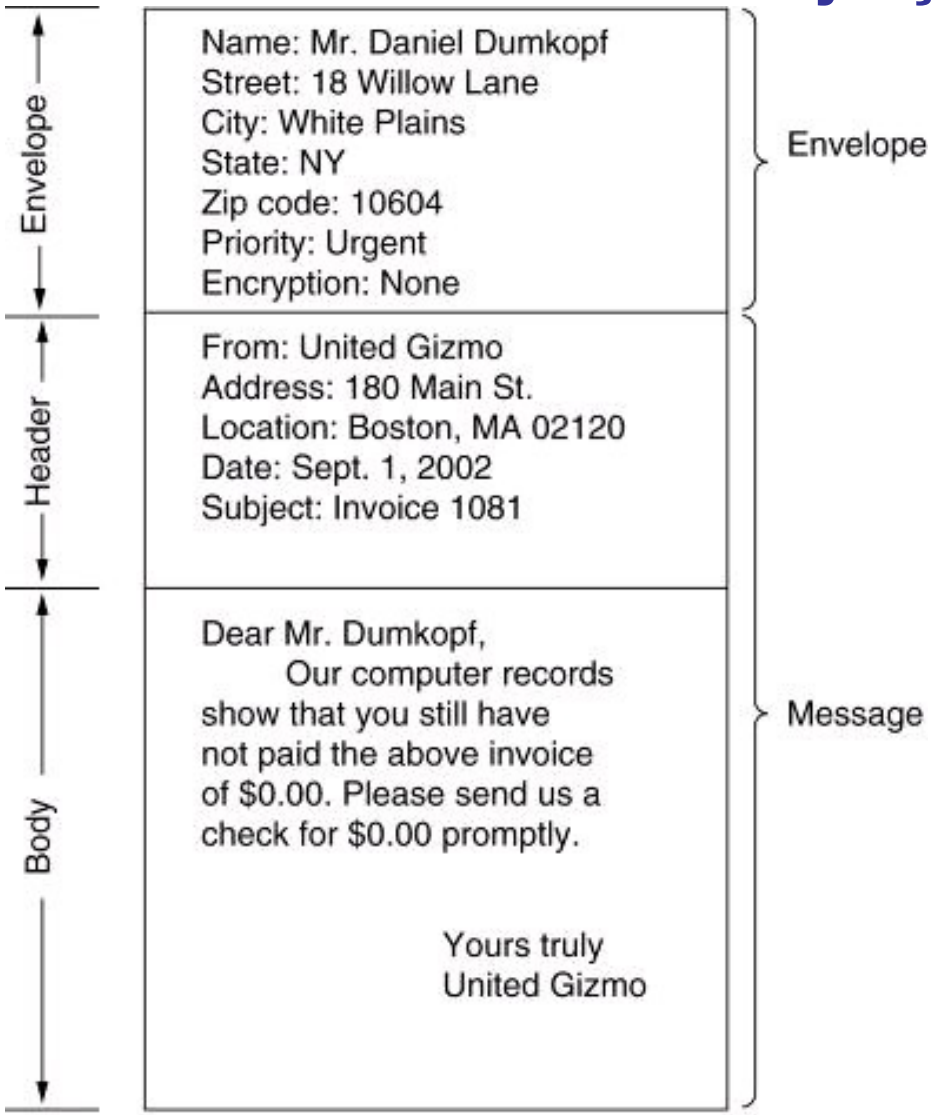
- Arhitectura sistemului de e-mail
- Adrese de e-mail
- Formatul mesajelor
- SMTP
- Livrarea finala a mesajelor

Arhitectura sistemului de e-mail



- Agent utilizator (Mail User Agent)
  - permite citirea si scrierea mesajelor
  - interfata utilizatorului cu sistemul de e-mail
- Agentul de transfer mesaje (Mail Transfer Agent)
  - 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 şi plicuri



Envelope (ro. Plic) = informatia necesara pentru transportul mesajului, folosita in protocolul SMTP

- adresa destinatar
- prioritatea
- nivel securitate

Header (ro. antet) = informatia de control pentru agentul utilizator

- perechi nume-valoare referitoare la utilizatori si la continutul mesajului

Body (ro. corp) = informatia destinata utilizatorului

- text sau multimedia

Posta electronica: Adrese e-mail

- Adresa e-mail

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

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

Campuri din antet care se refera la transportul mesajului.

Unele campuri sunt folosite de agentul de transfer

Mesaje auto-continute

Câmpuri folosite de agentul utilizator sau de utilizator

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

Antete adaugate de MIME – Multi-Purpose Internet Mail Extensions

Antet	Conţinut
MIME-Version:	Identifică versiunea de MIME
Content-Description:	Descrierea a ce este în mesaj (similar subiectului)
Content-Id:	Identificator unic al continutului
Content-Transfer-Encoding:	Cum este codificat continutul pentru transmisie
Content-Type:	Tipul datelor continute 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	Secvenţă neinterpretată de octeţi
	Postscript	Un document afişabil în PostScript
Message	Rfc822	Un mesaj MIME RFC 822
	Partial	Mesajul a fost fragmentat pentru transmisie
	External-body	Mesajul în sine trebuie adus din reţea
Multipart	Mixed	Părţi independente în ordine specificată
	Alternative	Acelaşi mesaj în formate diferite
	Parallel	Părţile 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 <bold> Carolyn </bold>
Happy birthday to you

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

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

SMTP

- *Simple Mail Transfer Protocol* (SMTP) este protocolul standard de aplicaţie pentru livrarea mesajelor de posta electronica de la sursa la destinaţie
- Foloseşte TCP si un schimb de mesaje-text intre client si server
  - comenzi (MAIL, RCPT, DATA, QUIT,…)
    - nume comanda urmat de parametri
  - raspunsuri
    - numar din 3 cifre (caractere) urmat de text
- Oferă o livrare sigura a mesajelor
- Alte funcţii:
  - Verificarea numelui unui utilizator

    S: VRFY Smith

    R: 250 Fred Smith <Smith@USC-ISIF.ARPA>

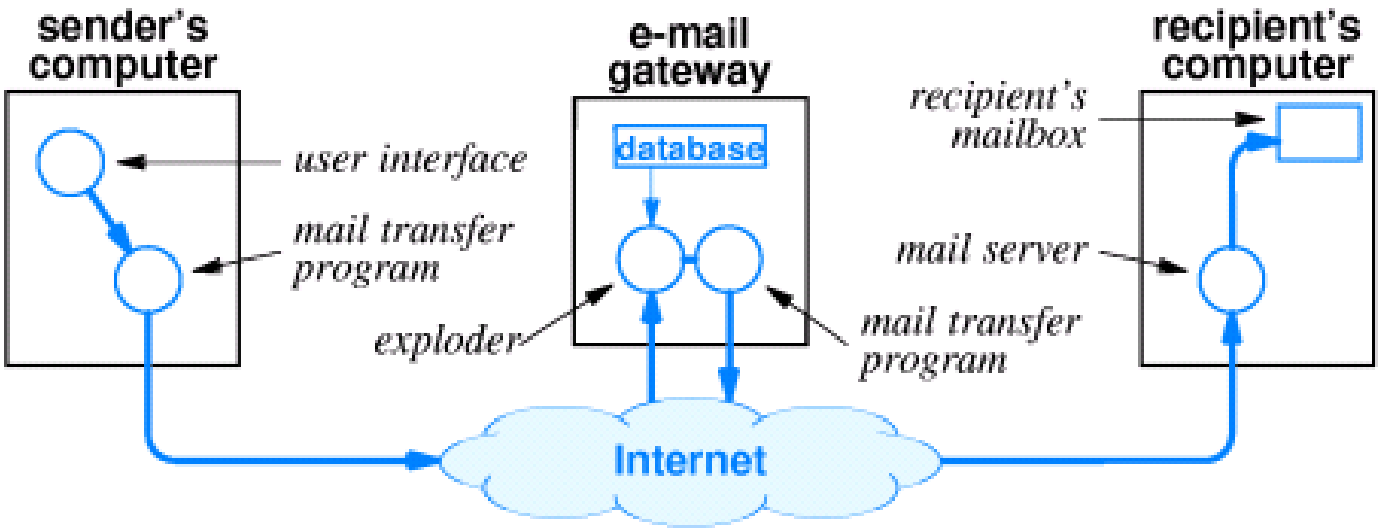
Transfer e-mail

Secventa incepe dupa stabilire conex TCP

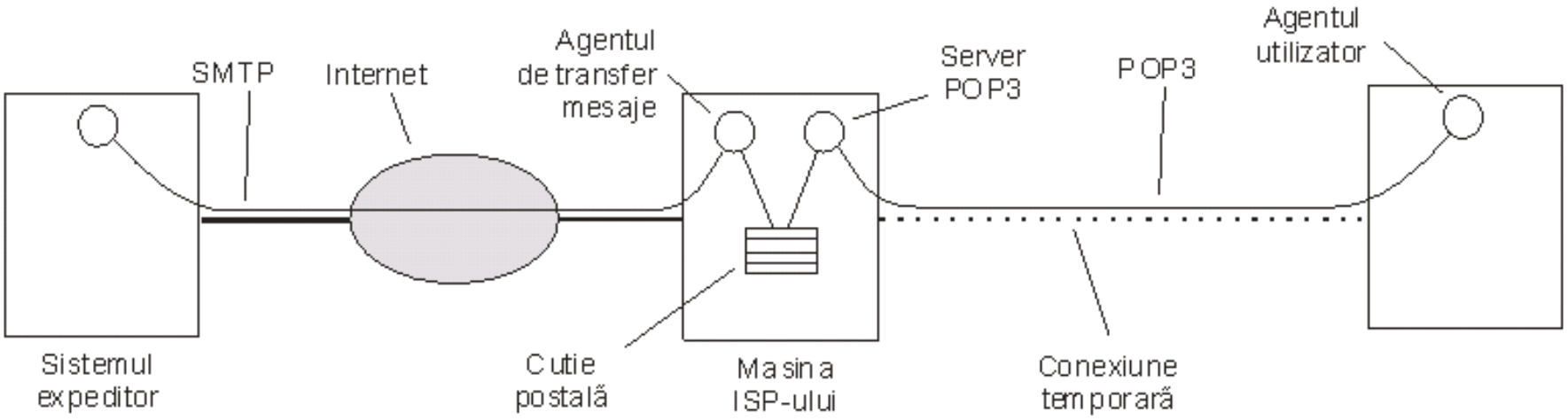
```
                S: 220 xyz.com SMTP service ready
C: HELO abcd.com
                S: 250 xyz.com says hello to abcd.com
C: MAIL FROM: <elinor@abcd.com>
                S: 250 sender ok
C: RCPT TO: <carolyn@xyz.com>
                S: 250 recipient ok
C: DATA
                S: 354 Send mail; end 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
```

E-Mail gateways

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



Livrarea finală



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

Protocoale

- 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
```

TRANSACTIONS

```
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
```

UPDATE

```
C: QUIT
                    S: +OK POP3 server disconnecting
```

IMAP

- **Internet Message Access Protocol (IMAP)**
  - Stil de lucru similar cu POP3
  - Definit in RFC diferit si foloseste port diferit de POP3
  - Permite accesul la mail de pe diferite staţii de lucru
    - IMAP pastreaza mail-uri la server
    - POP3 descarca mail-uri pe calculatorul client si (de regula) le sterge de la server

Studiu individual

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

7.2 POŞTA ELECTRONICĂ

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

7.2 ELECTRONIC MAIL

END

Nivelul Aplicaţie

Transferul de fişiere (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

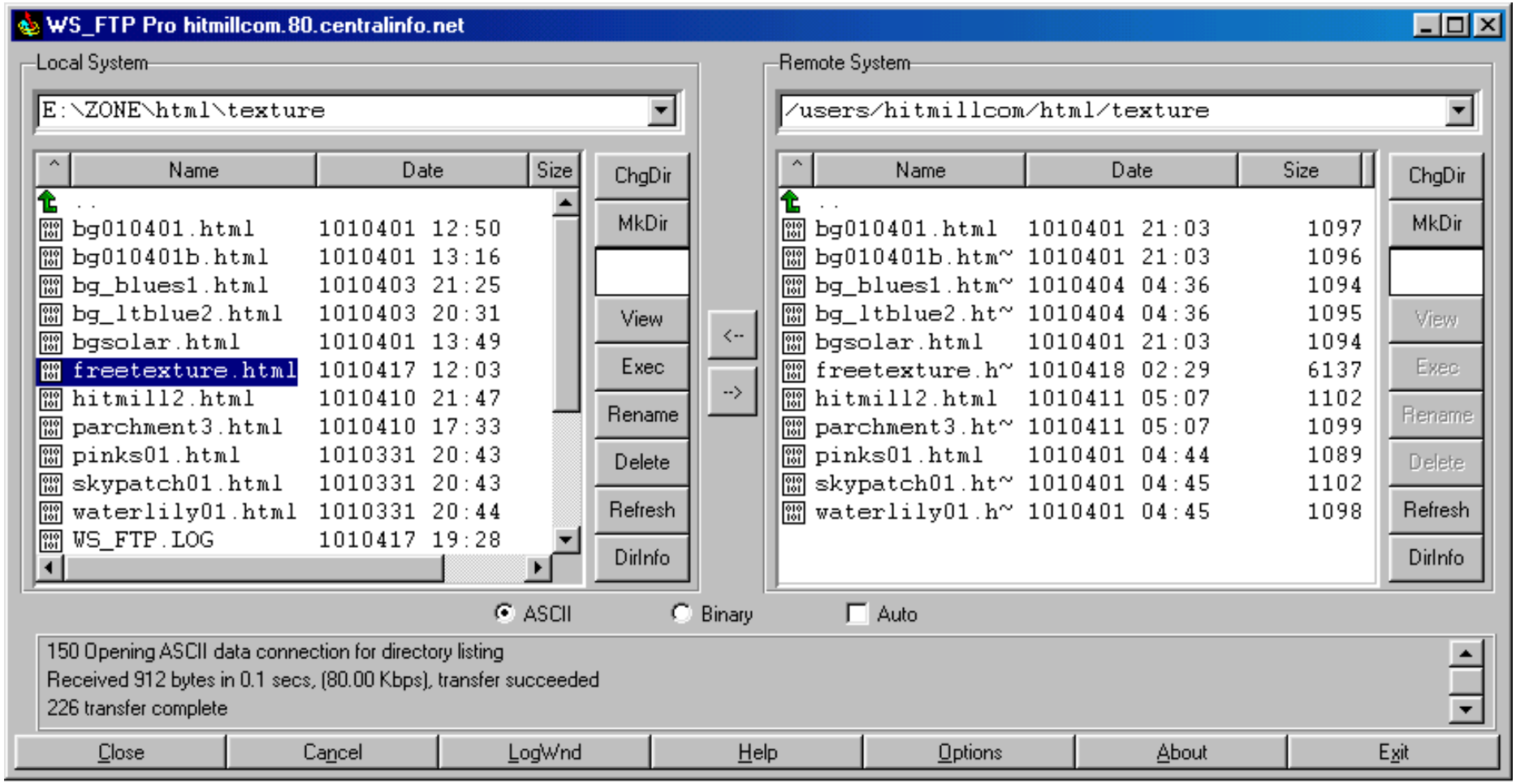
- Standard pentru transfer de fisiere (RFC959)
- Protocol independent de SO si de hardware
- Protocolul permite:
  - Transferul unor fisiere oarecare
    - intre utilizator si server
    - intre doua servere
  - Listarea continutului unui director
  - Modificarea directorului curent
  - Gestionarea restrictiilor 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



Modelul FTP

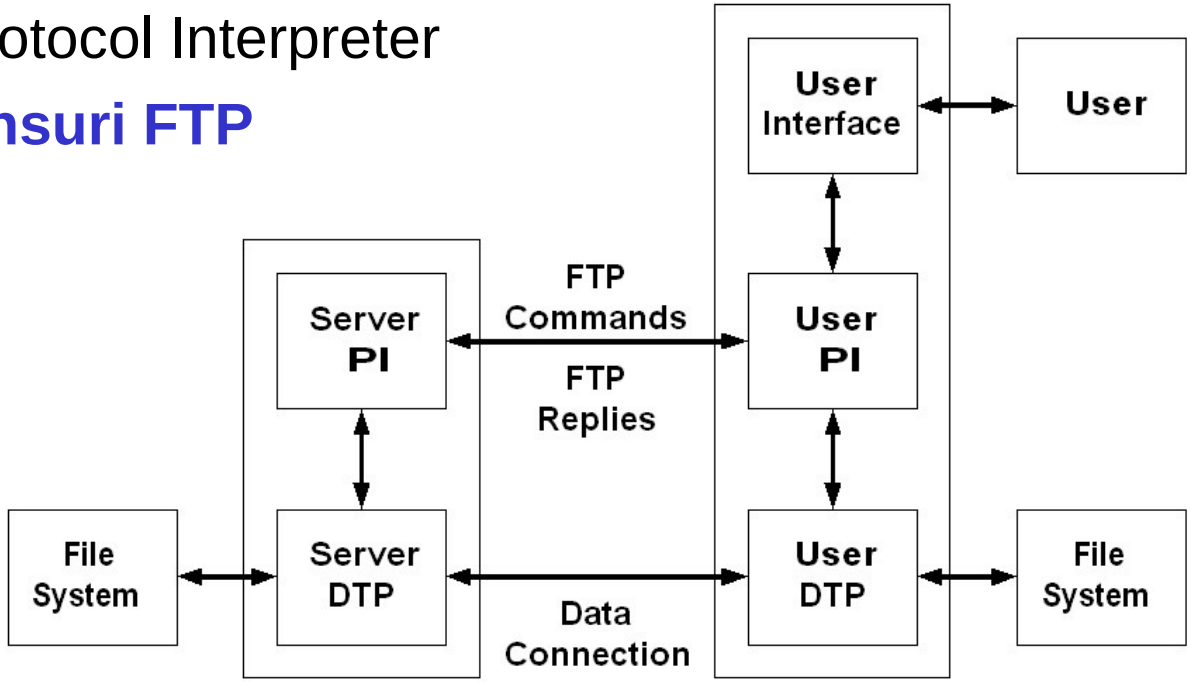
Are **doua conexiuni**

- control - pentru comenzi si raspunsuri

- User PI – User Protocol Interpreter
  - initiaza conexiunea de control (lucreaza dupa protocolul telnet)
  - genereaza **comenzi FTP**
- Server PI – Server Protocol Interpreter
  - genereaza **raspunsuri FTP**

- date - pentru transmisie / receptie simultana

- User DTP – User Data Transfer Protocol
- Server DTP – Server Data Transfer Protocol



Cateva Comenzi FTP

Comanda	Descriere
ABOR	Abort proces de conexiune de date
ALLO <bytes>	Aloca spatiu fisier pe server
CWD <dir path>	Schimba director de lucru pe server
DELE <filename>	Sterge fisier specificat, pe server
MODE <mode>	Mod de transfer (S=stream, B=block, C=compressed).
MKD <directory>	Creaza director specificat, pe server
QUIT	De-logheaza de pe server
TYPE <data type>	Tip date (A-ASCII, E-EBCDIC, B-binar)
USER <username>	username pentru login
PASS <password>	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 distanta accepta *conexiune de control* de la client

Se stabileste apoi *conexiunea de date* – metode:

Activa – serverul (activ) se conecteaza la client

- clientul specifica pe conexiunea de control o adresa si un port;
- serverul initiaza 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: initiaza 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

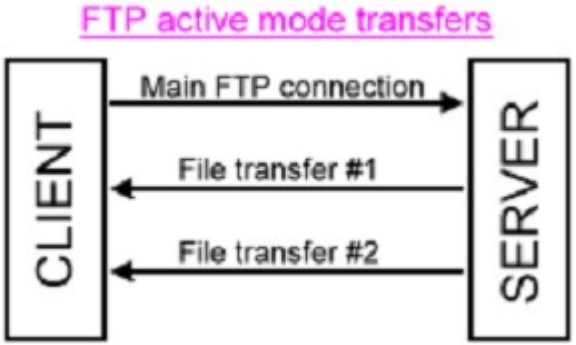
Server: Entering Passive Mode (172,16,62,36,13,111)

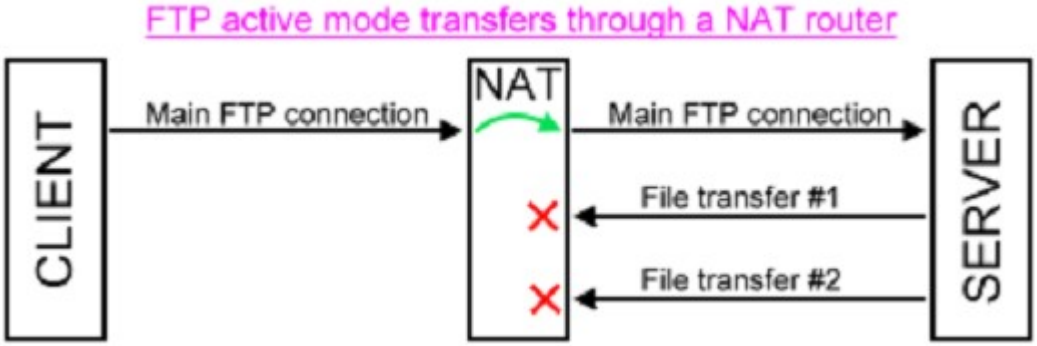
Client: initiaza conexiunea de date la portul specificat de server

Server: confirma

FTP prin NAT

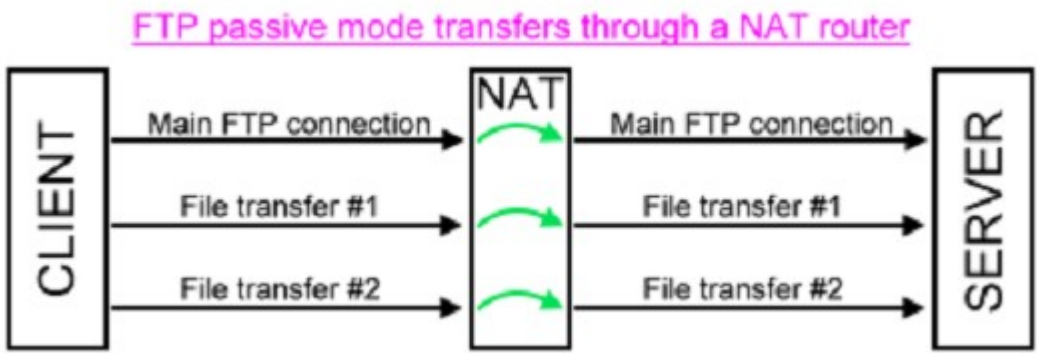
- Apare o problema in cazul in care clientul FTP este intr-o retea locala si se conecteaza la server trecand prin NAT.
- In modul activ serverul nu se poate conecta la client si va fi blocat de NAT.





FTP prin NAT

- Este posibila folosirea modului pasiv deoarece se realizeaza conexiunea de la client, care are o adresa IP locala, spre server.



Transfer FTP intre doua servere

```
      Control              +---------------+          Control
          ------------->|   User-FTP    |<------------
          |             |   User-PI     |             |
          |             |     "C"       |             |
          V             +---------------+             V
 +----------------+                          +----------------+
 |   Server-FTP   |   Data Connection        |   Server-FTP   |
 |    "A"         |<------------------------>|    "B"         |
 +----------------+ Port (A)      Port (B)   +----------------+
```

User-PI - Server A	User-PI - Server B
C->A : Connect	C->B : Connect
C->A : PASV	
A->C : 227 Entering Passive Mode A1,A2,A3,A4,a1,a2	
	C->B : PORT A1,A2,A3,A4,a1,a2
	B->C : 200 Okay
C->A : STOR file	C->B : RETR file
	B->A : Connect to HOST-A, PORT-a

Securitate

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

Protocoale de Securitate
-
Criptologie

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 securitatii

- confidentialitatea
  - informaţia este disponibilă doar utilizatorilor autorizaţi
- integritatea
  - informaţia poate fi modificată doar de utilizatorii autorizaţi sau în modalitatea autorizată (mesajul primit nu a fost modificat în tranzit sau măsluit)
- disponibilitatea
  - accesul la informaţie al utilizatorilor autorizaţi nu este îngrădit (opusul este denial of service)

Probleme derivate

- autentificarea
  - determinarea identităţii persoanei cu care schimbi mesaje înainte de a dezvălui informaţii importante
- autorizarea (controlul accesului)
  - protectia impotriva accesului ne-autorizat
- non-repudierea
  - transmitatorul nu poate nega transmiterea unui mesaj pe care un receptor l-a primit

Metode de rezolvare

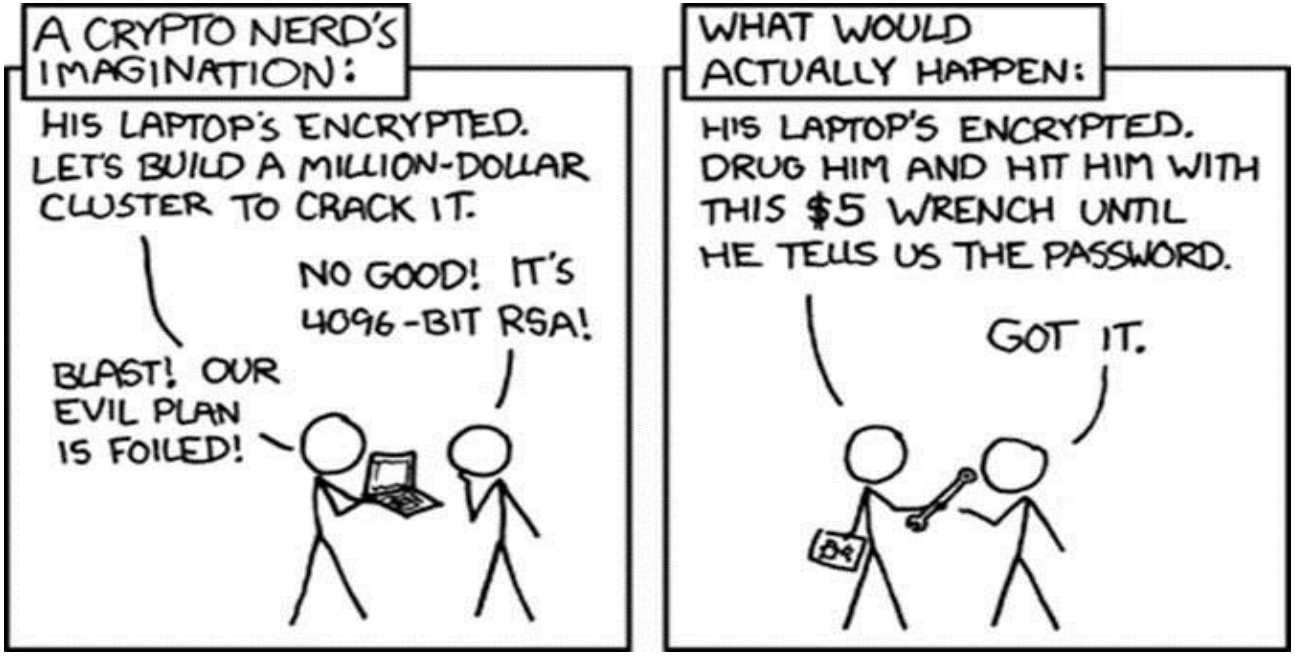
- Organizare
  - Algoritmi de criptare si hash
  - Mecanisme de securitate
    - **criptare, rezumare (hash), semnatura digitala**
  - Servicii si **protocoale** de securitate

- 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

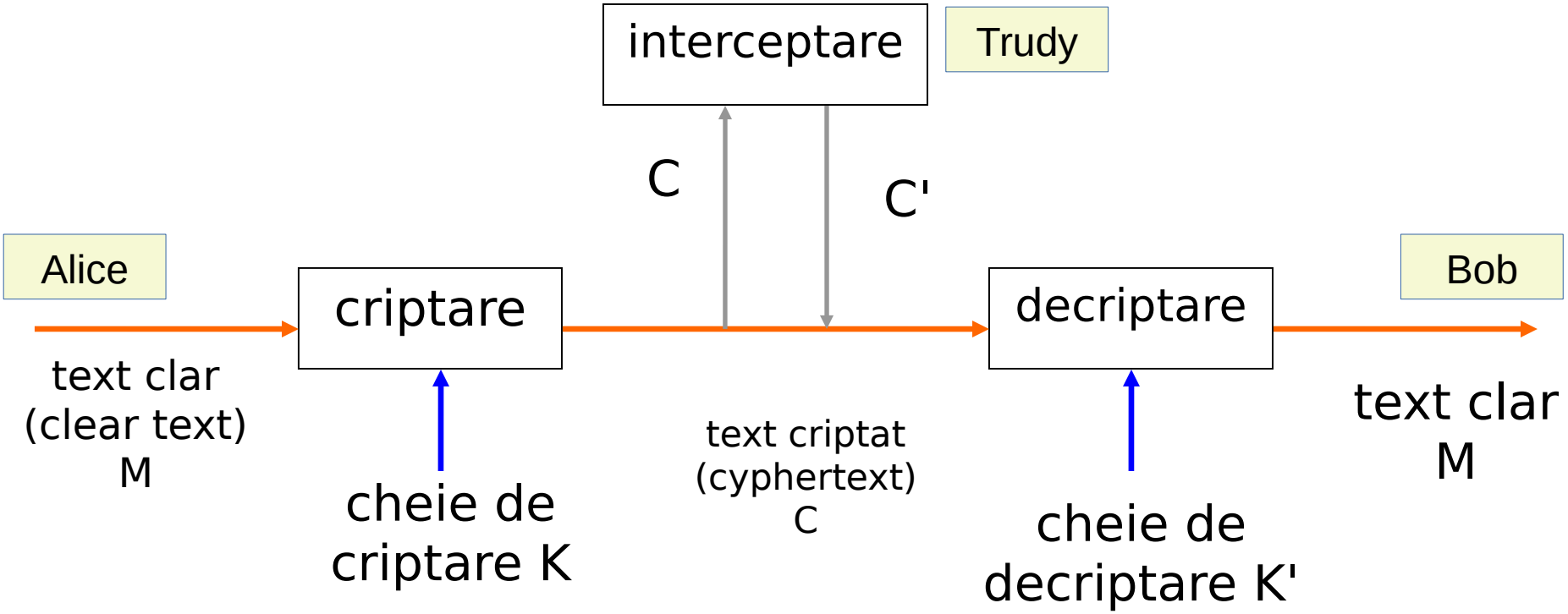
Alte aspecte

- Politici de securitate.
- Control software (antivirus).
- Control hardware:
  - Cartele inteligente;
  - Biometrie.
- Control fizic (protecţie).
- Educaţie.
- Măsuri legale.

Elemente de criptografie



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).

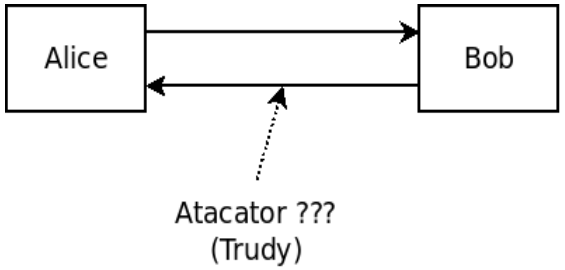
Criptografie – concepte de baza

Mecanism de baza pentru asigurarea securitatii pentru transferul datelor sau stocarea lor.

Criptologie = Criptografie + Criptanaliza

Utilizare:

- Criptarea datelor, protectie impotriva interceptarii datelor
- Comunicatie anonima
- Asigurarea integritatii datelor (functii Hash)
- Verificarea originii datelor (semnaturi digitale)
- Verificarea identitatii (certificate X.509)
- ….



Definiţii

- Spargerea cifrurilor = criptanaliză.
- Proiectarea cifrurilor = criptografie.
- Ambele sunt subdomenii ale criptologiei.
- Transformarea F realizată la cifrarea unui mesaj:

F : {M} x {K} -> {C}, unde:

  - {M} este mulţimea mesajelor;
  - {K} este mulţimea cheilor;
  - {C} este mulţimea criptogramelor.

- Operaţii:
  - Criptarea (Encryption): C = E(k,M).
  - Decriptarea (Decryption): M = D(k’,C).
- D(k’,E(k,M))=M

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 porţiuni de text;
  - Exemplu pentru o bază de date - modificare / efect.

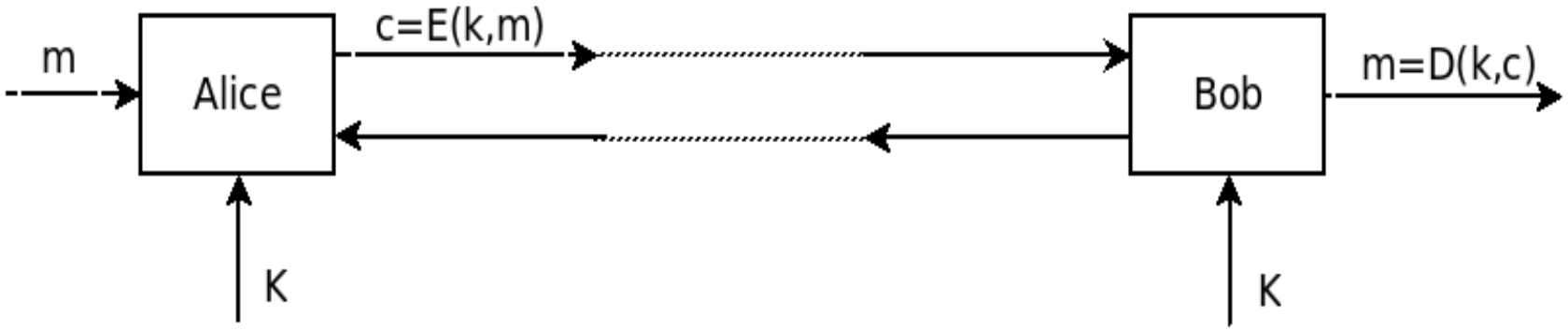
Caracteristicile sistemelor secrete

- 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 probabilităti apropriate

Cerinţe criptosisteme cu chei secrete

- Cerinţe generale:
  - Cifrare şi descifrare eficiente pentru toate cheile.
  - Sistem uşor de folosit (gasire chei de transformare).
  - Securitatea să depindă de chei, nu de algoritm.
- Cerinţe specifice pentru confidenţialitate: să fie imposibil computaţional ca un criptanalist să determine sistematic:
  - Cheia de criptare din C, chiar dacă ar cunoaşte M.
  - M din C (fără a cunoaste cheia k).
- Cerinţe specifice pentru integritate: să fie imposibil computaţional ca un criptanalist să determine sistematic:
  - Cheia de criptare k, din C, chiar dacă ar cunoaşte M.
  - Cifrul C' astfel ca D(k,C') să fie un mesaj valid (fără a cunoaşte cheia de criptare).

Criptare simetrică



Foloseste aceeasi cheie k pentru criptare si pentru decriptare

- m – mesajul în clar (clear text)
- c – mesajul criptat (cyphertext)
- Criptare: c=E(k,m)
- Decriptare: m=D(k,c)

m=D(k,E(k,m))

Vulnerabilitate: Cum ajung Alice si Bob sa cunoască valoarea K?

• Cerinţe criptosisteme cu chei secrete

- Cerinţe generale:
  - Cifrare şi descifrare eficiente pentru toate cheile.
  - Sistem uşor de folosit (chei de transformare).
  - Securitatea să depindă de chei, nu de algoritm.
- Cerinţe specifice pentru confidenţialitate: să fie imposibil computaţional ca un criptanalist să determine sistematic:
  - Transformarea D_k din C, chiar dacă ar cunoaşte M.
  - M din C (fără a cunoaste D_k).
- Cerinţe specifice pentru autentificare: să fie imposibil computaţional ca un criptanalist să determine sistematic:
  - Transformarea E_k, din C, chiar dacă ar cunoaşte M.
  - Cifrul C' astfel ca $D_k(C')$ să fie un mesaj valid (fără a cunoaşte E_k).

Criptare simetrică - exemple

Algoritmi istorici	Algoritmi reali/eficienti
• Cifrul lui Cezar • Cifruri de substitutie • Substituție polialfabetică • ROT13	• Data Encryption Standard(DES) • 3DES • Advanced Encryption Standard(AES) • Foarte multi alti algoritmi: RC4, Twofish, Blowfish, IDEA, etc.

Criptare simetrică - exemple

DES:

- Dezvoltat în 1977, cheie k de 56b, blocuri de 64b, 16 runde
- Considerat nesigur din anii '90

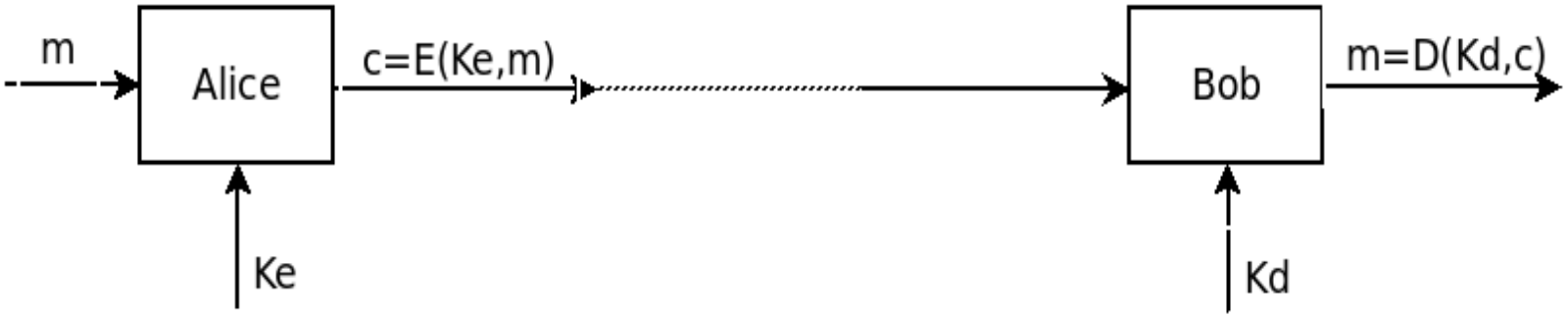
3DES:

- Rulează algoritmul DES de 3 ori, cu 3 chei K1, K2, K3 (max 168b)\
- Mare îmbunătățire a securității față de DES

AES:

- Dezvoltat în 2001
- Chei de 128b (10 runde), 196b (12 runde) sau 256b (14 runde)
- Considerat sigur, poate fi folosit pentru doc guvernamentale

Criptare asimetrică



Folosește o cheie publică penru criptare (Ke) și o cheie privată pentru decriptare (Kd), cunoscută doar de Bob.

- Ke – cheie de criptare (publică)
- Kd – cheie de decriptare (privată)
- Criptare: c=E(ke,m)
- Decriptare: m=D(kd,c)

m=D(kd,E(ke,m))

Criptare asimetrică - exemple

Exemple de algoritmi: ElGamal, RSA, etc.

RSA:

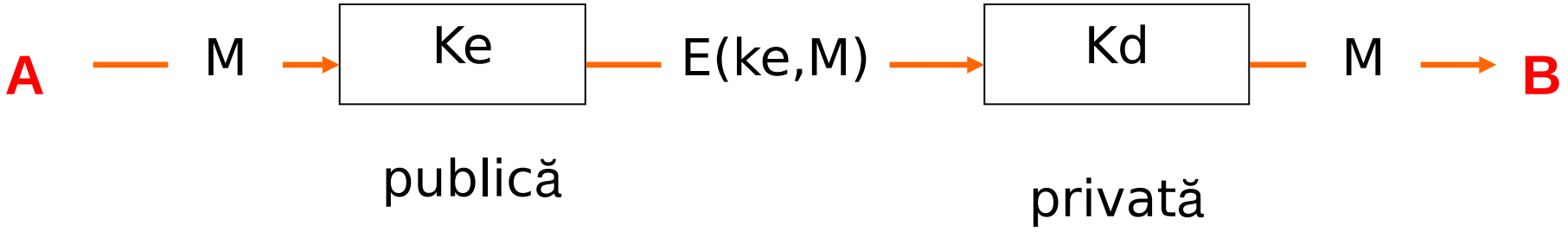
- Dezvoltat în 1977 de R. Rivest, A. Shamir, L. Adleman
- Dimensiunea cheilor: 1024b – 4096b
- Este considerat sigur pentru chei peste 2048b

Modelul criptografic cu chei publice

- Sistemele criptografice:
  - Simetrice.
  - Asimetrice:
    - Propuse de Diffie şi Hellman în 1976.
    - Chei diferite de cifrare Ke şi descifrare Kd cu proprietatea

      D(kd,E(ke,m)) =m
    - Nu se pot deduce (uşor) una din alta, mai precis:
      - Este extrem de greu să se deducă Kd din Ke;
      - D nu poate fi "spart" prin criptanaliză cu text clar ales.

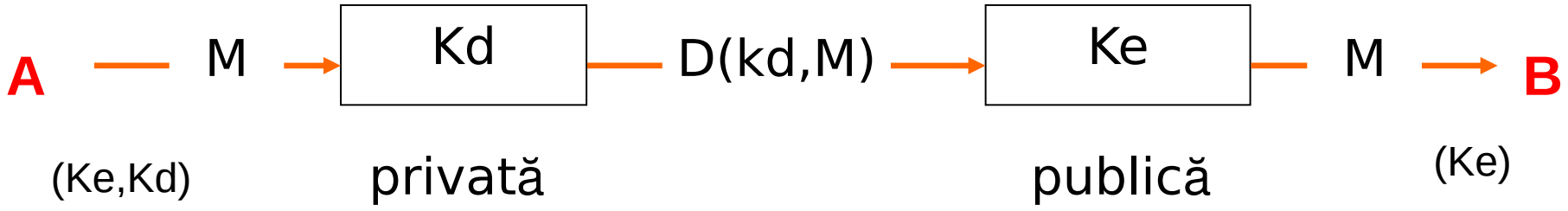
Schema de confidențialitate



- 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 Db poate intelege mesajul M

Schema de integritate (semnatura digitala)



- Pentru integritate / autentificare:

condiţia necesară este ca transformările E şi D să comute, adică

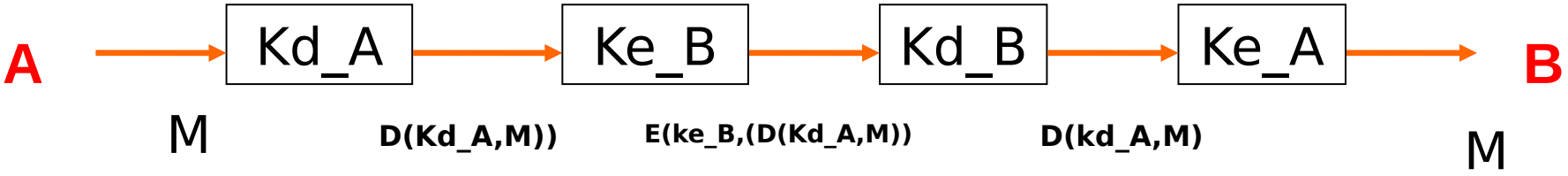
$$E(ke,D(kd,M)) = D(kd,E(ke,M)) = M.$$

Cheia Kd se foloseste pentru criptarea mesajului M

Se asigură integritatea

- Oricine poate folosi cheia publica Ke pentru a decripta mesajul
- Nimeni nu poate modifica mesajul M deoarece nu cunoaste cheia privata kd

Schema de autentificare



autentificare M este criptat mai intai cu cheia privata a lui A

- D(Kd_A,M) este un fel de “semnatura” a mesajului

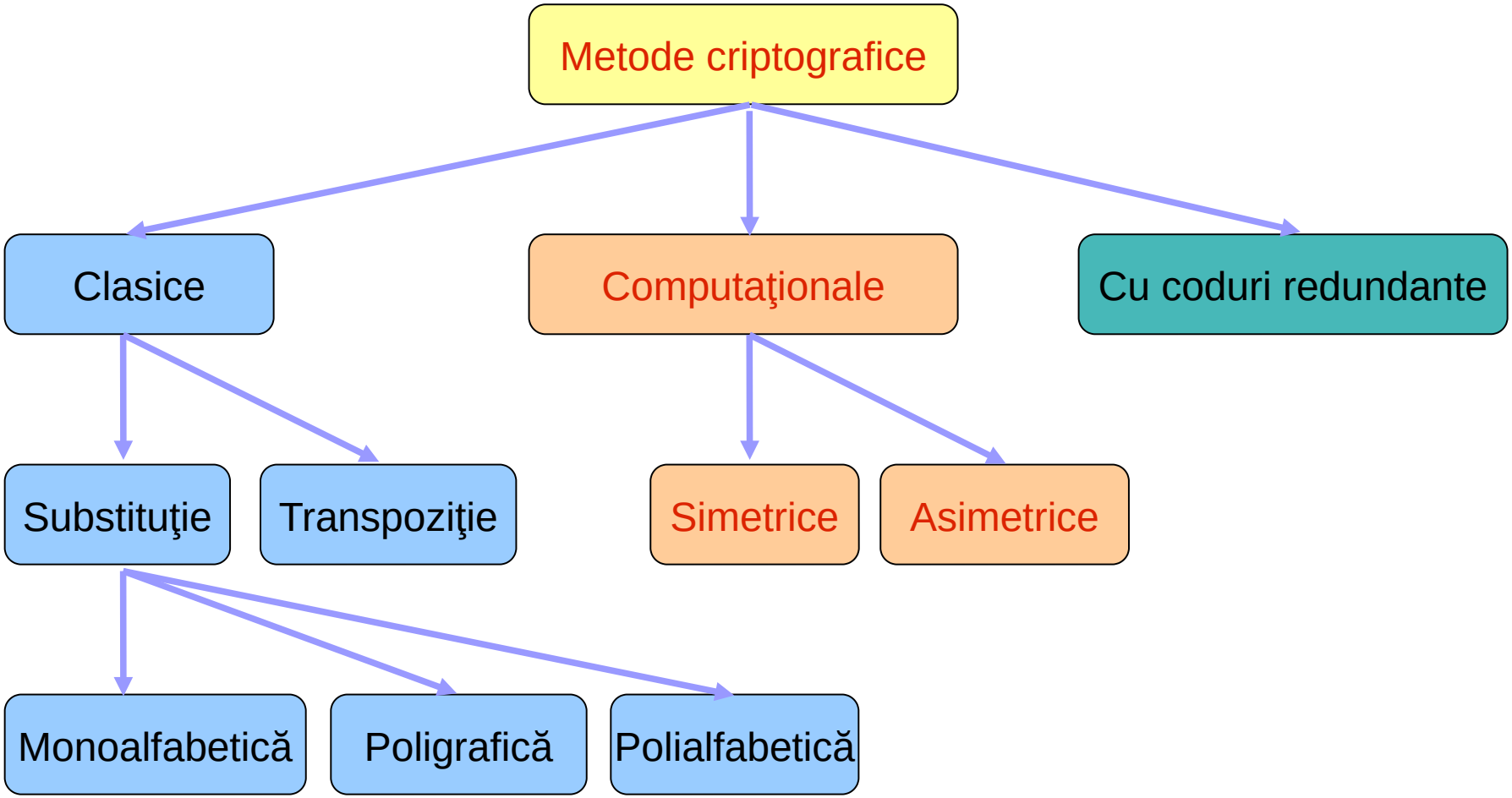
rezultatul este apoi criptat cu cheia publica a lui B

Se asigura că A este sursa mesajului

și că mesajul este confidențial

non-repudiere folosind perechea Da(M) si M, B poate demonstra ca a primit mesajul de la A

Clasificare generală



Algoritmi de criptare simetrica

Cifrarea prin substitutie

Cifrul lui Cezar (substitutie 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
| | | | | | | | | | | | | | | | | | | | | | | | |
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: **CRIPTOGRAFIE**

text cifrat: **FULSWRJUDILH**

fiecare litera este inlocuita de litera aflata la distanta 3 de ea

Relatia de calcul

$$c[i] = (m[i] + 3) \bmod 26$$

In general

$$c[i] = (a.m[i] + b) \bmod n$$

- **ROT-13** este o alta varianta de substitutie monoalfabetica in care pentru criptare se realizeaza o rotatie cu 13 pozitii.

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

POLIGRAFPOLIGRAGPOLIGRAFPOLIGRAFPOLI	*cheie*
AFOSTODATACANPOVESTIAFOSTCANICIODATA	*clar*
PTZAZFDFIONITGOATGEQGWOXIQLVOTITSOEI	*cifrat*

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 succesive ale unei matrice si
- 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

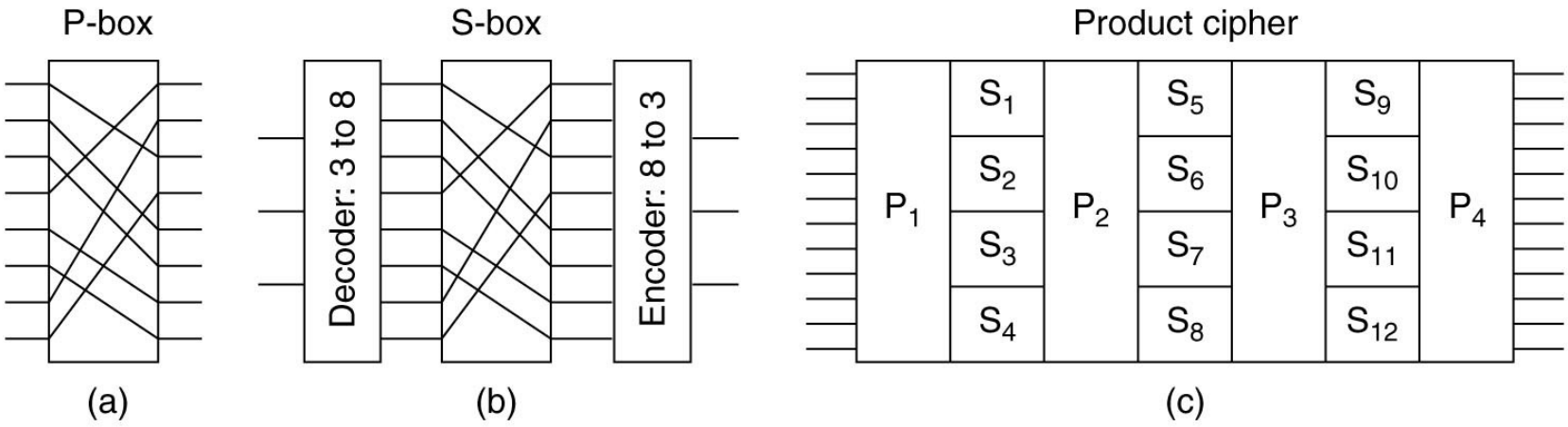
ordine: 76543812

text clar: AFOSTODATACANPOVESTIAFOSTCANICIO

POLIGRAF
AFOSTODA
TACANPOV
ESTIAFOS
TCANICIO

text cifrat: DOOIAVSOTNAISAINOCTAFASCATETOPFC

Cifruri produs

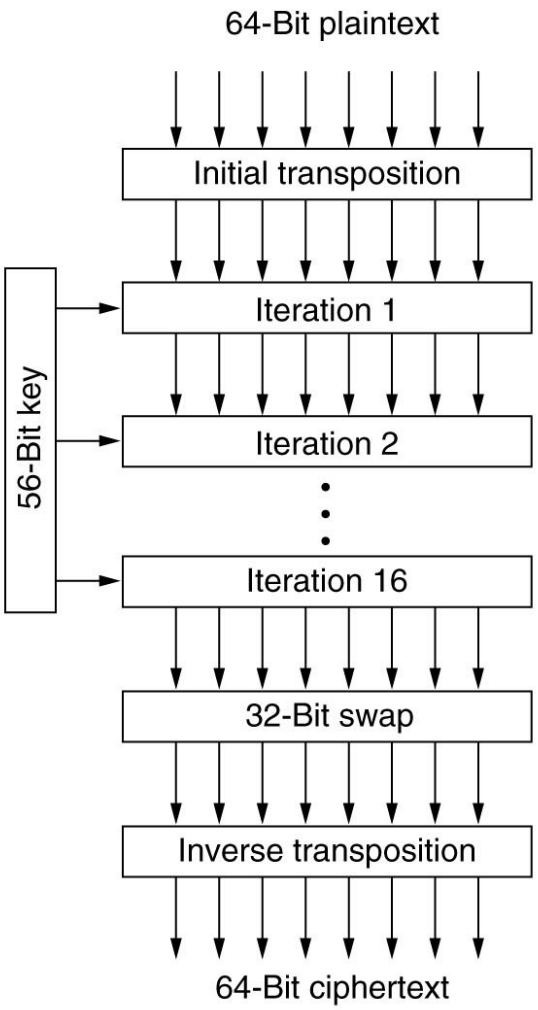


Principii pentru a obţine o securitate mai mare:

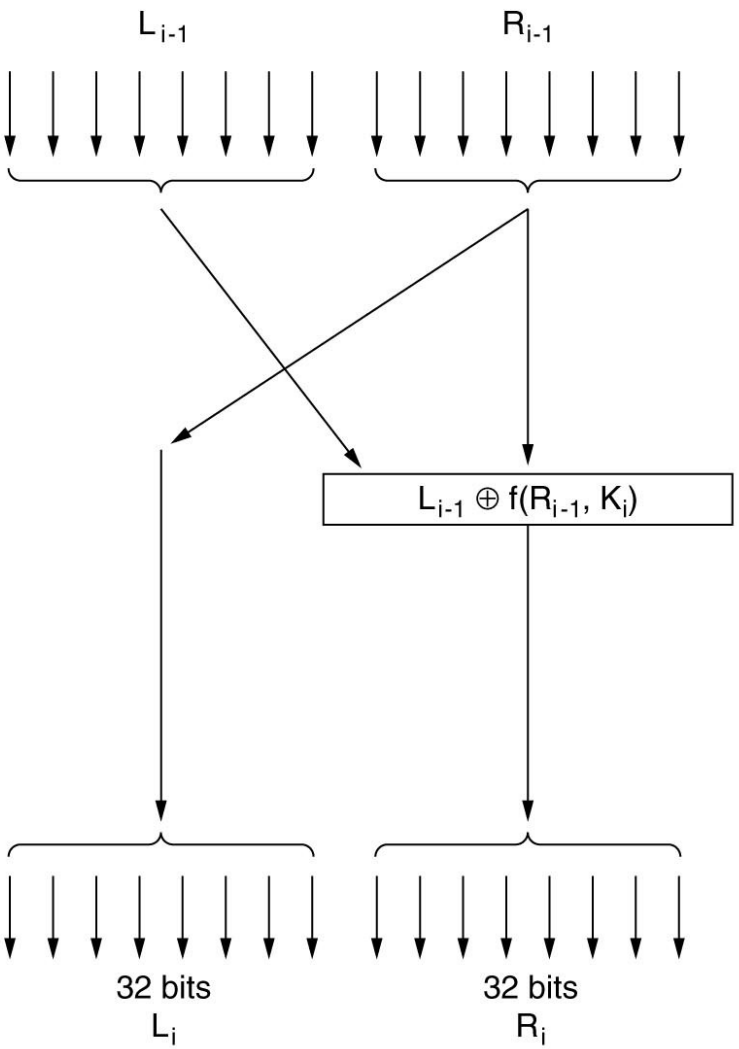
- compune două cifruri "slabe", complementare
  - P-box – permutare (transpozitie) - asigură difuzia
  - S-box – substitutie - asigură confuzia
- repetă aplicarea permutării şi substituţiei

DES (Data Encryption Standard)

Schema generală O iteraţie

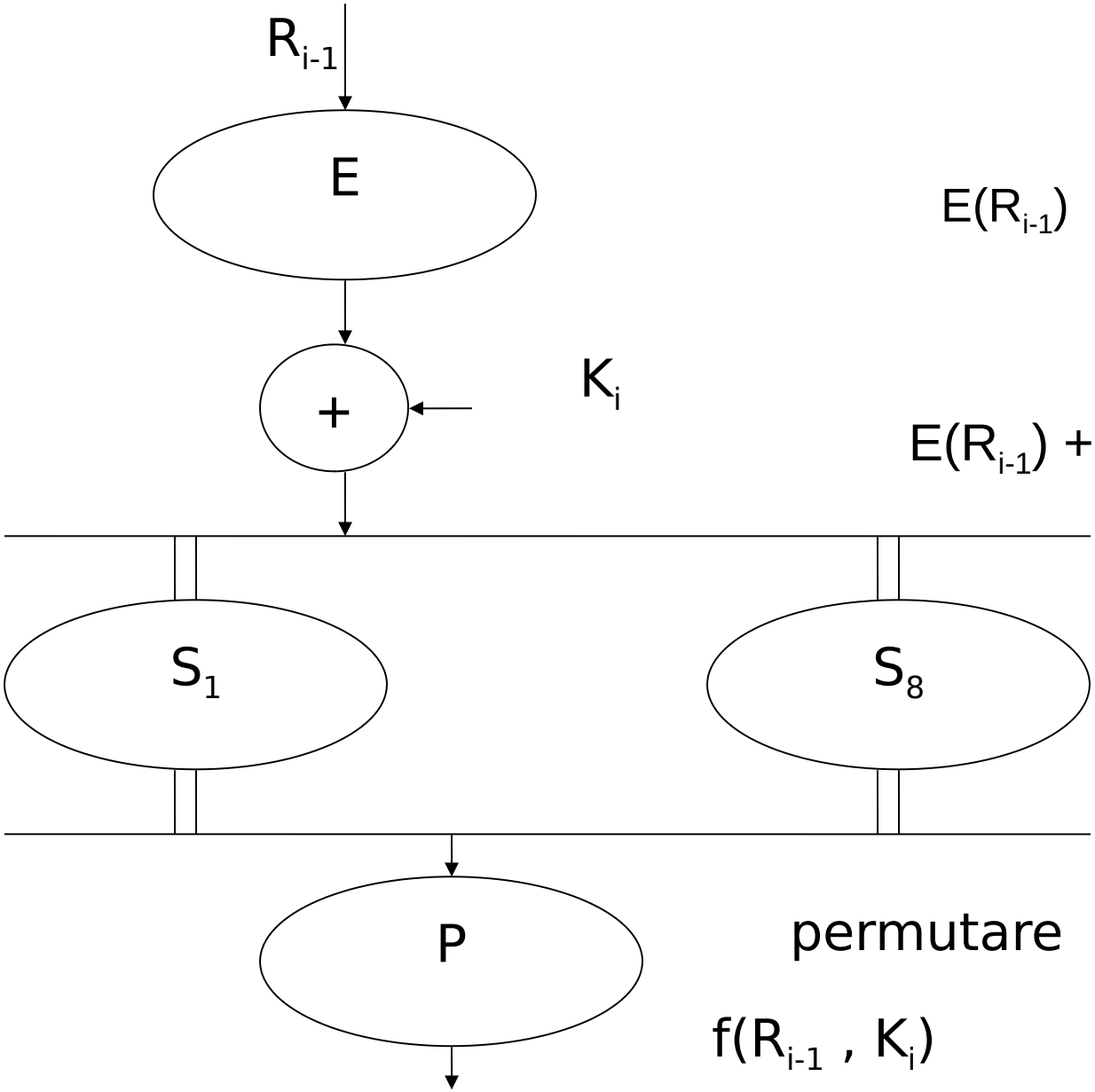


(a)



(b)

Calculul lui $f(R_{i-1},k_i)$



32 biti

$E(R_{i-1})$ expandare la 48 de biti prin repetarea unora

$E(R_{i-1}) + Ki \Rightarrow B_1B_2...B_8$

8 blocuri a 6 biti sunt intrari in S-boxes

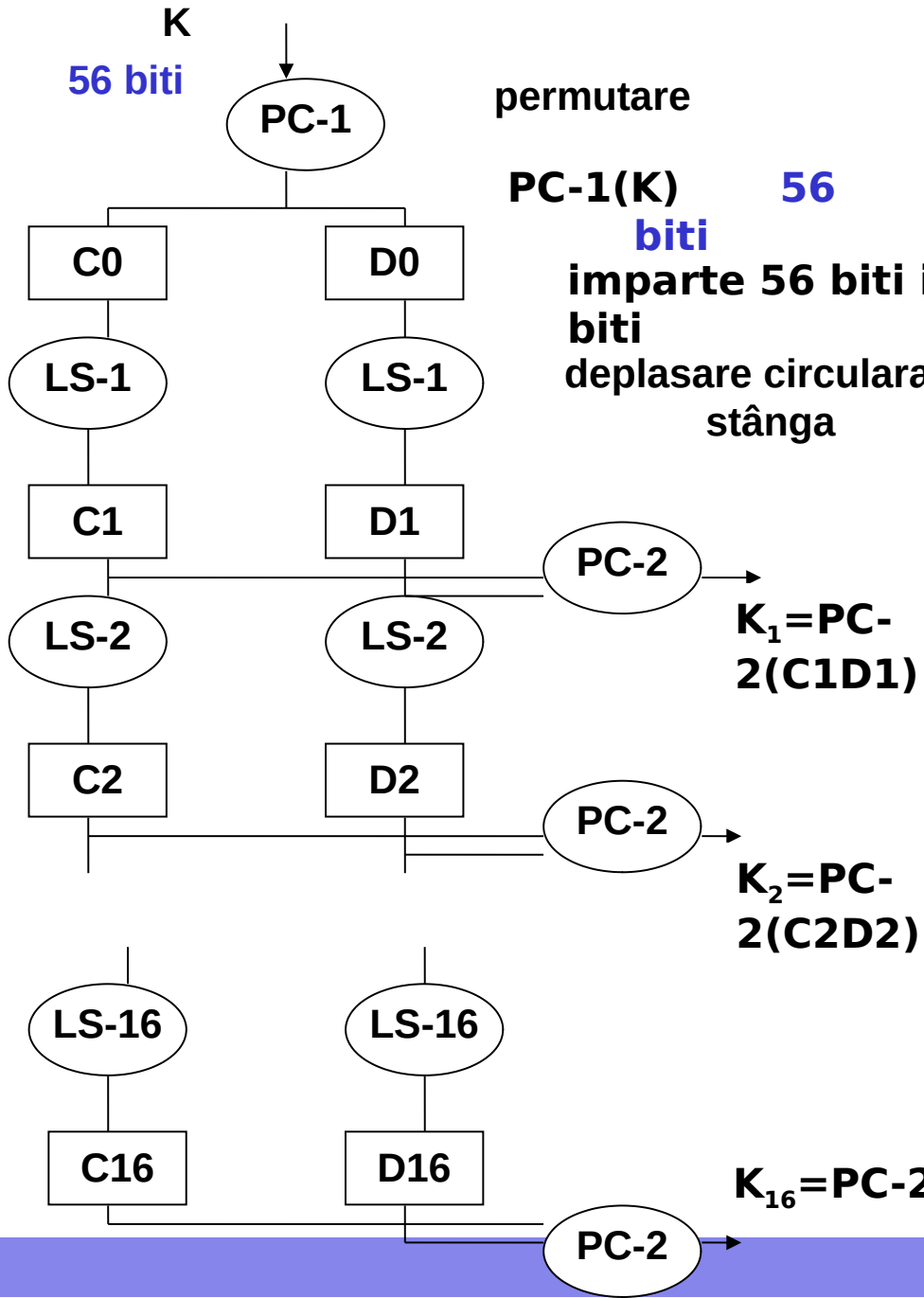
$S_j(B_j)$

Fiecare Sj produce un cod de 4 biti

permutare $P(S_1(B_1), ... S_8(B_8))$

$f(R_{i-1}, K_i)$ 32 biti

Calculul cheilor



permutare

PC-1(K) 56 biti

imparte 56 biti in 2 blocuri x 28 biti

deplasare circulara la stânga (cu 1 sau 2 biti, in functie de numarul ciclului)

K_1=PC-2(C1D1)

extrage si permuta 48 biti din cei 56 ai C1D1

K_2=PC-2(C2D2)

K_{16}=PC-2(C16D16)

Comentarii

- Elementele cheie ale algoritmului nu au fost făcute publice
  - Controverse
    - Există trape (capcane) care să uşureze decriptarea de către NSA?

      NSA declară că NU.
    - Descoperirea şi folosirea unei astfel de trape de un criptanalist răuvoitor

 unele detalii despre S-box au fost dezvăluite de NSA

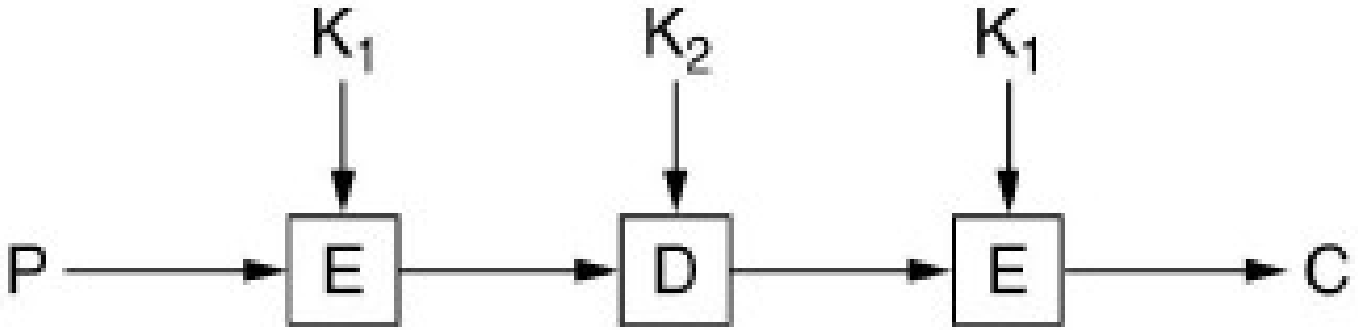
Comentarii

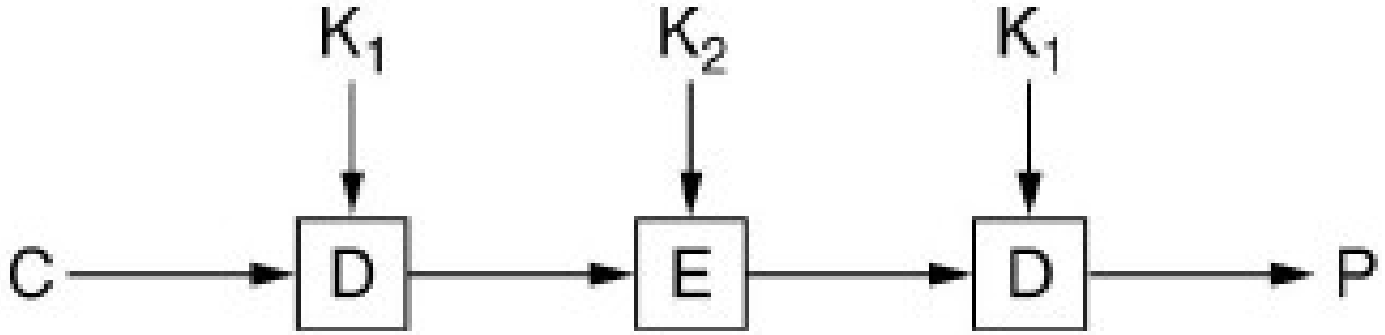
- Număr de iteraţii (16) sunt suficiente pentru difuzie?
  - Experimental, după 8 iteraţii nu se mai văd dependenţe ale biţilor de ieşire de grupuri de biţi din intrare
- Lungimea cheii
  - Cheie DES de 56 biţi spartă prin forţă brută (4 luni * 3500 maşini) în 1997
  - Dar, nu au fost raportate deficienţe în algoritm
  - Triple DES “măreşte” lungimea cheii

Triplu DES

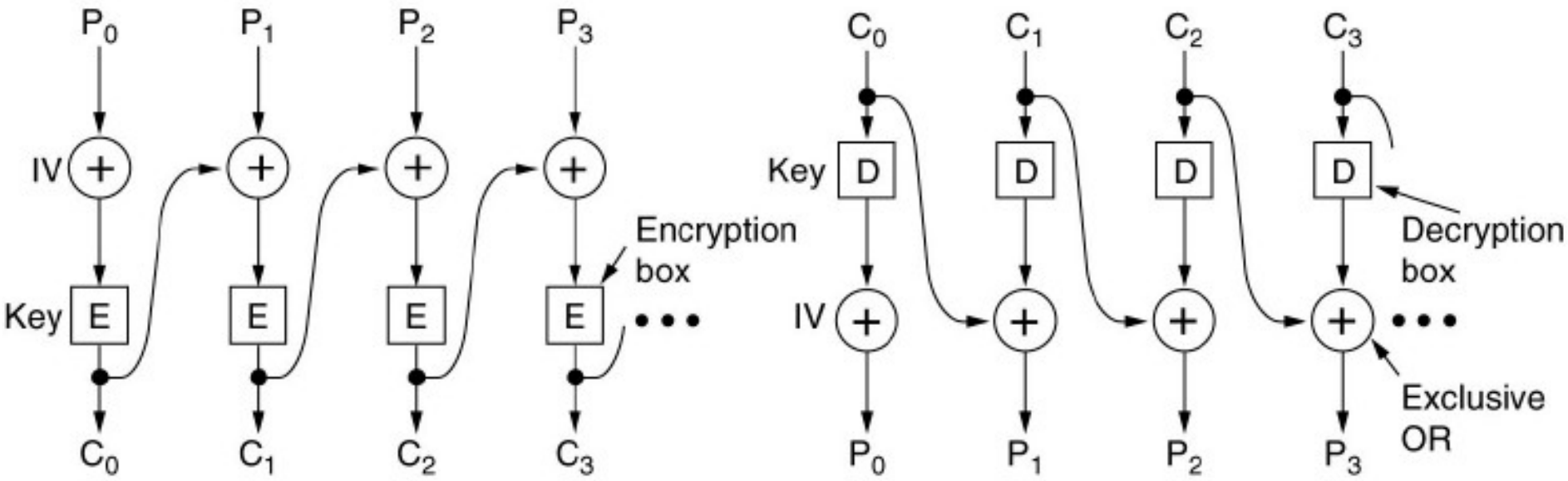
"Creste" lungimea cheii folosind

- 2 chei
- 3 runde de criptare / decriptare





Inlantuirea blocurilor cifrate: CBC - Cipher Block Chaining



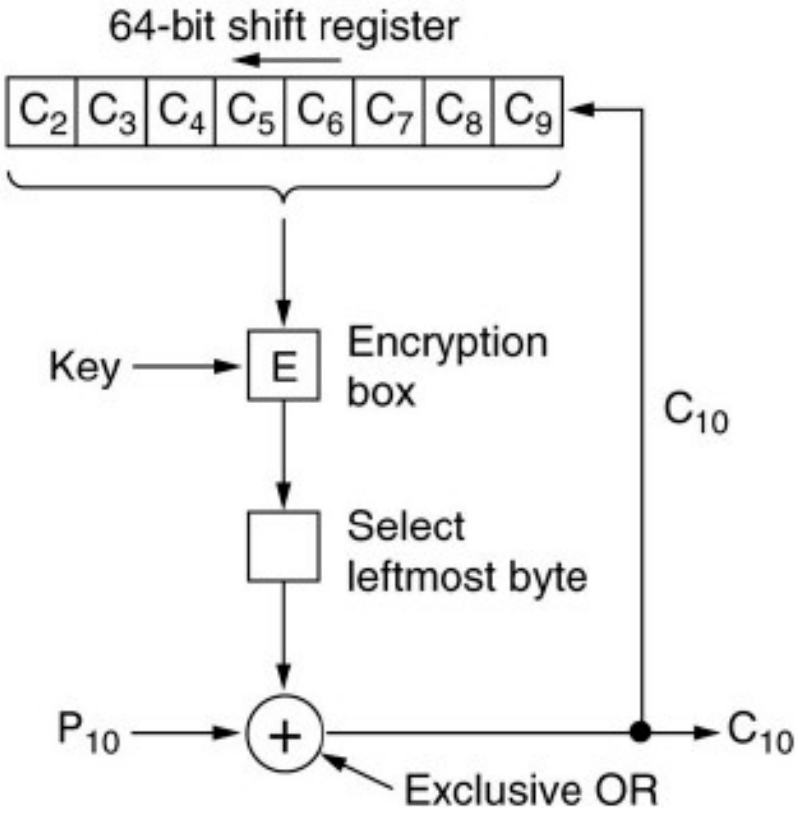
Key – cheie secreta

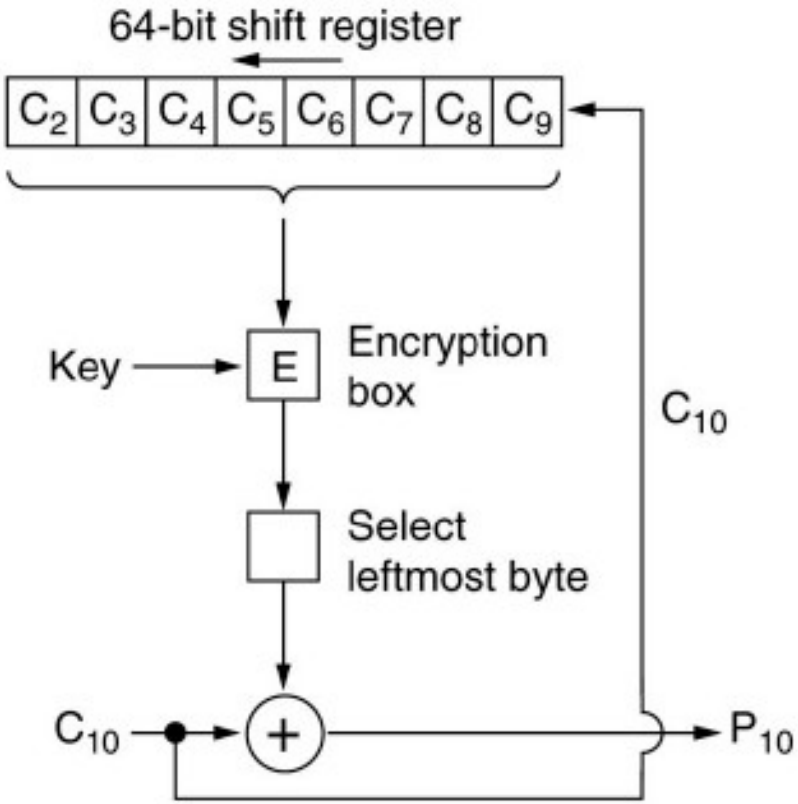
IV – Initialization Vector

- 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



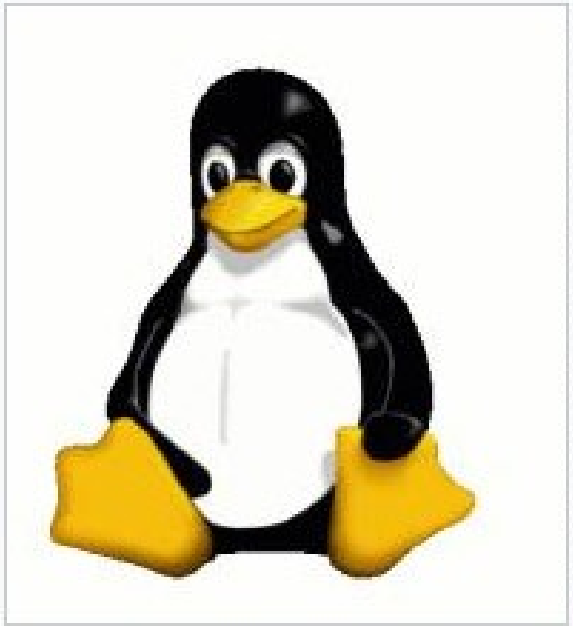


Foloseste un Initialization Vector ca prima valoare in Registrul de deplasare

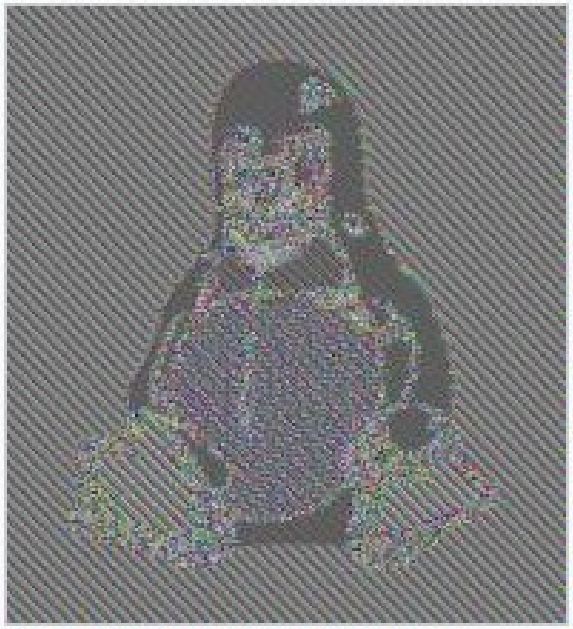
O eroare de un bit in criptograma conduce la decriptarea eronata a 8 octeti

- generati in pasii in care bitul se afla in registrul de deplasare

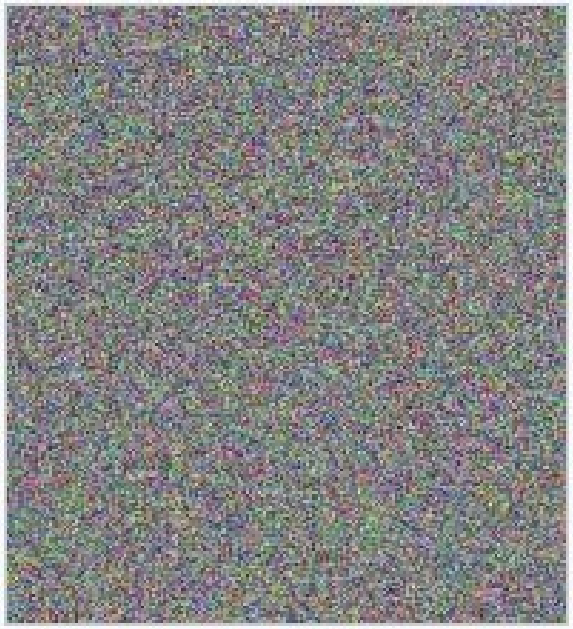
Exemplu cifrare CBC



Original image



Encrypted using ECB mode

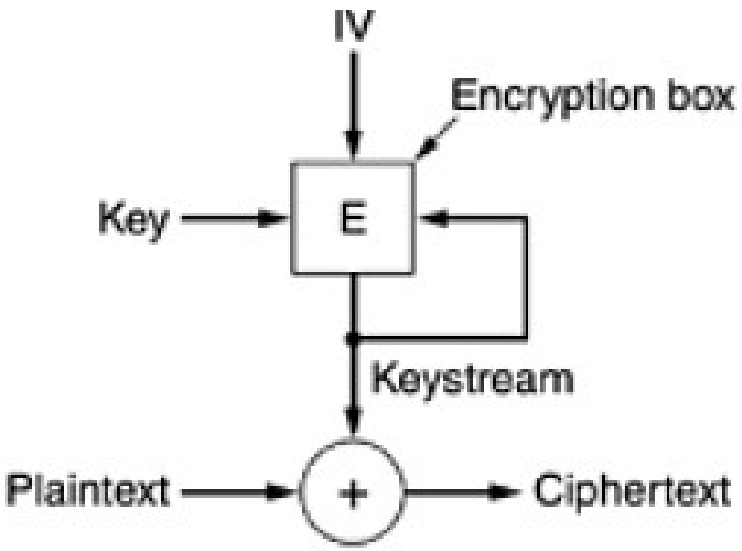


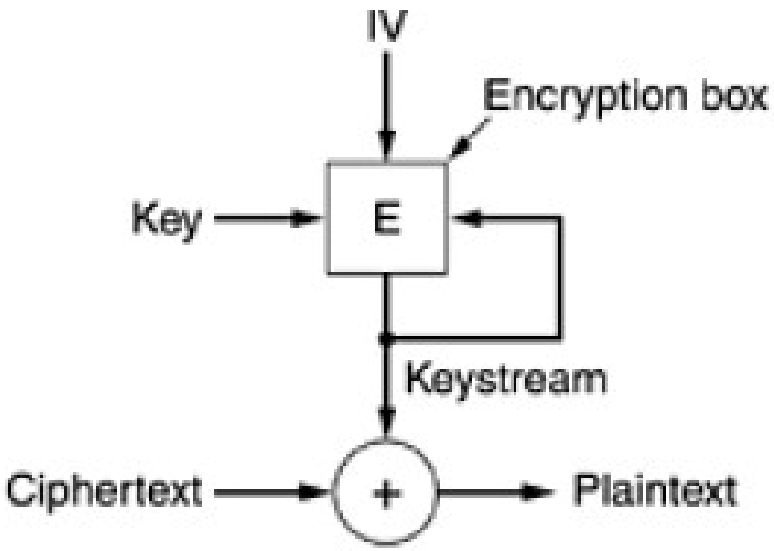
Modes other than ECB result in pseudo-randomness

The image on the right is how the image might appear encrypted with CBC, CTR or any of the other more secure modes—indistinguishable from random noise. Note that the random appearance of the image on the right does not ensure that the image has been securely encrypted; many kinds of insecure encryption have been developed which would produce output just as "random-looking".

Source:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Cifrarea secventială (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

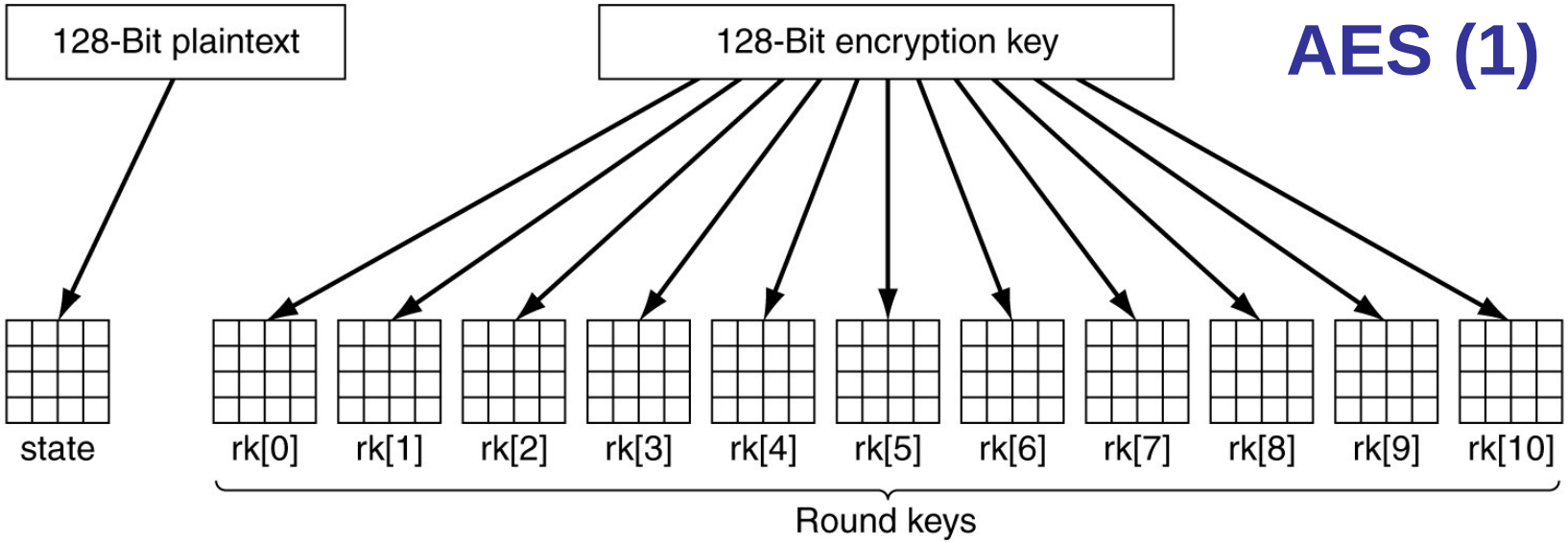
Regulile concursului organizat de NIST (ianuarie 1997) erau:

1. Algoritmul trebuie să fie un cifru bloc simetric.
2. Tot proiectul trebuie sa fie public
3. Trebuie să fie suportate chei de 128, 192, şi de 256 biţi
4. Trebuie să fie posibile atât implementări hardware cât şi software
5. Algoritmul trebuie să fie public sau oferit cu licenţă nediscriminatorie.

Finaliştii şi scorurile lor au fost următoarele:

1. Rijndael (din partea lui Joan Daemen şi Vincent Rijmen, 86 voturi)
2. Serpent (din partea lui Ross Anderson, Eli Biham şi 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)

AES (1)



Foloseste o matrice de stare de 4*4 octeţi (bloc de 128 biţi)

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 copiaza un bloc de 128 biti text clar in state
- se face XOR intre state si cheia rk[0]

AES (2)

In fiecare runda se fac patru operatii

substitutie – la nivel octet, foloseşte tabel substituţie
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 înmulţite cu o matrice

$$\begin{vmatrix} s'0i \\ s'1i \\ s'2i \\ s'3i \end{vmatrix} = \begin{vmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{vmatrix} \begin{vmatrix} s0i \\ s1i \\ s2i \\ s3i \end{vmatrix}$$

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 = coeficienţii unui polinom

Ex. $23_{(10)} = 10111_{(2)}$ este polinomul $1*x^4+0*x^3+1*x^2+1*x+1$

$x^4+ x^2+ x+1$

adunarea coeficienţilor făcută modulo 2

înmulţirea făcută ca la polinoame, dar modulo P

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)
  - substituţii şi mixare coloane folosesc operaţii cu sens în teoria algebrică (nu simple tabele greu de explicat)
- autorii nu au oferit argumente matematice
- nu sunt suspectate trape (sau “scurtături” ascunse)

Algoritmi de criptare asimetrica

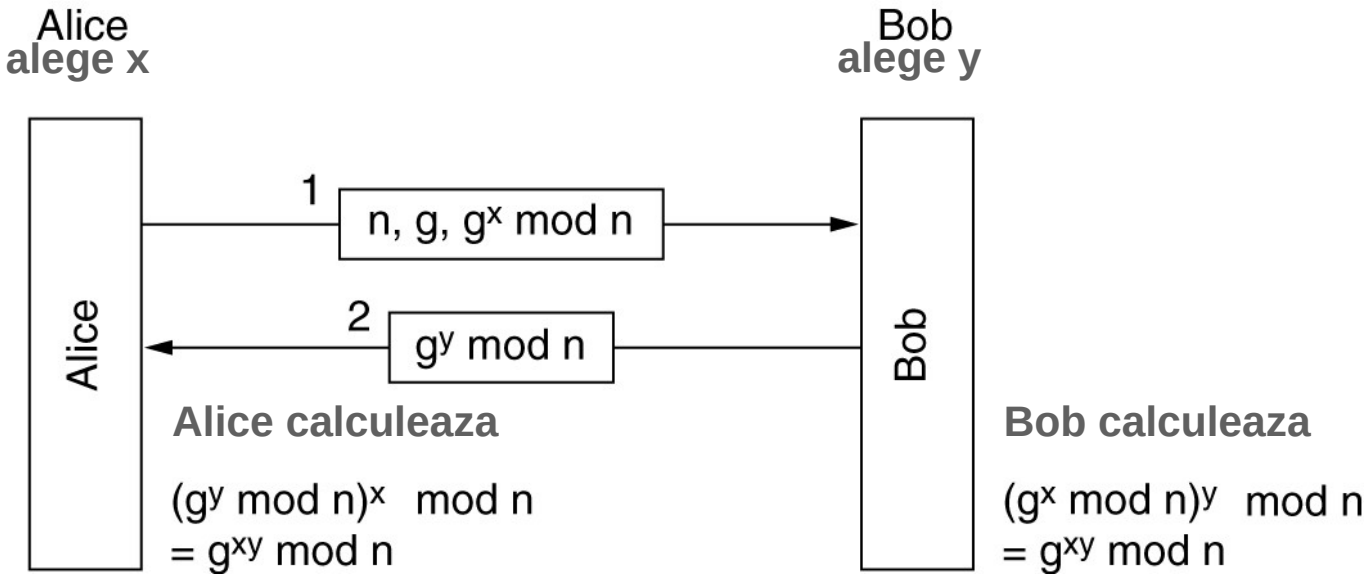
Cifrarea prin functii greu inversabile

- functii greu inversabile
  - cunoscînd 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.

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 mod n

g^{xy} mod n nu poate fi calculat din g^x mod n si g^y mod n cand n este mare

g < n (generator) are proprietatea: orice p poate fi scris ca g^k mod n

adica: pentru fiecare *p* intre 1 si *n*-1 inclusiv, exista o putere *k* a lui *g* astfel ca $p = g^k \bmod n$.

Algoritmi exponentiali – RSA

In RSA (Rivest, Shamir si Adleman):

Criptarea si decriptarea se fac prin functii exponentiale
Criptarea se face prin calculul

$$C = (M^e) \bmod n$$

unde **(e, n)** reprezintă cheia de criptare.
M este un bloc de mesaj (valoare întreagă între 0 si n-1)
C este criptograma.

Decriptarea se face prin calculul

$$M = (C^d) \bmod n$$

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 => $\Phi(p) = p-1$.

daca $n = p*q$ cu p, q prime atunci

$$\Phi(n) = \Phi(p) * \Phi(q) = (p-1)\,(q-1)$$

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* şi *q*,

   (de ex. de 1024 biţi).

2. Se calculează $n = p \xi q$

şi $z = (p - 1)\xi(q - 1)$.

3. Se alege *d* un număr relativ prim cu *z*

d poate fi un numar prim care satisface

d > (p-1) si d > (q-1)

4. Se găseşte *e* astfel încât $e \xi d = 1 \bmod z$.

5. (e, n) este cheia de criptare.

(d, n) este cheia de decriptare.

Comentarii

Cheia de criptare (e,n) se face publica

Problema:

- cunoasterea lui (e,n) sa nu permita deducerea lui d

Securitate pastrata deoarece:

- p si q sunt numere prime foarte mari
- p si q sunt pastrate secrete

Cifrarea si descrifrarea sunt comutative si mutual inverse

$$(M^d \bmod n)^e \bmod n = M$$

- RSA utilizată ptr confidentialitate si autentificare.

Nu au fost identificate atacuri reusite cu RSA

Rezumatele mesajelor

Rezumatele mesajelor

Este mai usor sa semnezi digital rezumatul decat mesajul intreg !

Folosite in semnaturi digitale datorita proprietatilor utile

1. Rezumatul lui P - MD(P) - este usor de calculat
2. Este imposibil sa se afle P din MD(P)
3. Rezumatul nu poate trucat: nimeni nu poate gasi P' avand un rezumat identic cu P, adica MD(P') = MD(P)
4. O schimbare de 1 bit a intrarii schimba multi biti din iesire

Functii hash

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

Functii Hash: MD5 - Message Digest 5

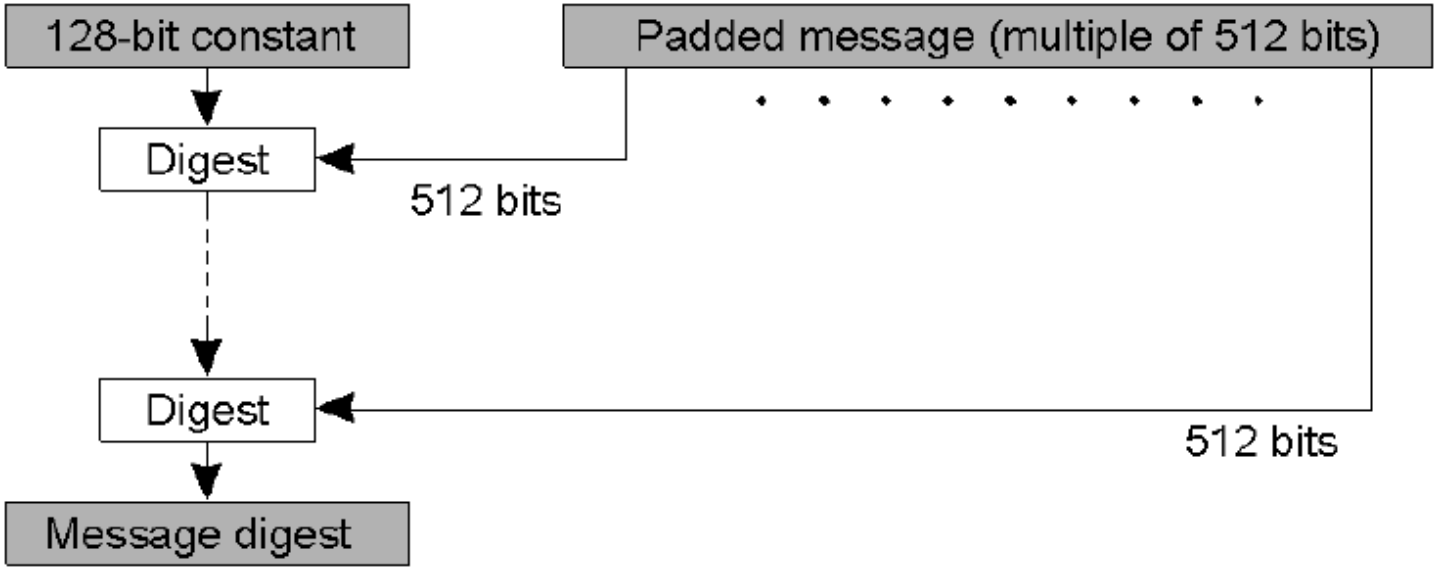
MD5 calculeaza un rezumat de 128 biti dintr-un mesaj de lungime multipla de 512 biti

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 AND y) OR ((NOT x) AND z)
G(x,y,z) = (x AND z) OR (y AND (NOT z))
H(x,y,z) = x XOR y XOR z
I(x,y,z) = y XOR (x OR (NOT z))

O runda are 16 **iteratii**.

$b_0,\ldots,b_{15}$ – **sub-blocuri** 32-biti (total 512 biti)

p, q, r, s – variabile *digest*

$C_1, \ldots, C_{16}$ – constante (in total 64)

<<< denota rotatie stanga

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

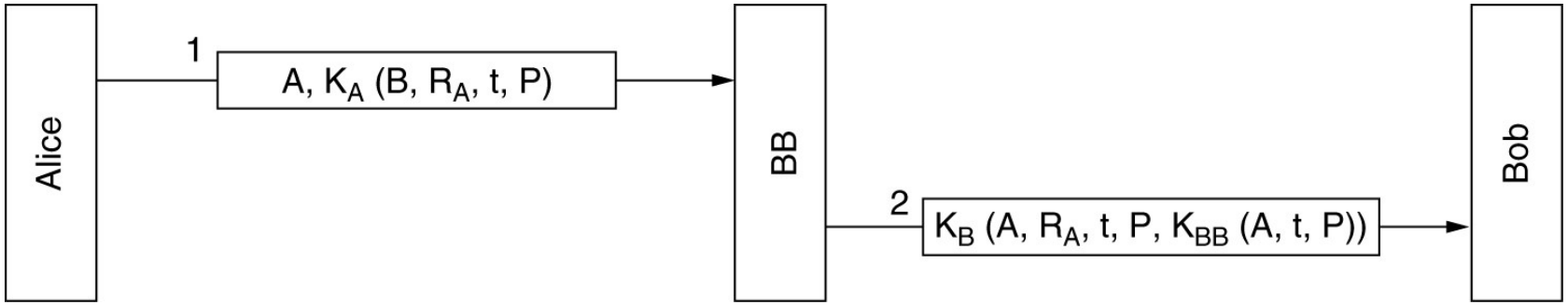
Comentarii

- Rezistenta la coliziuni
  - o functie H este rezistenta la coliziuni daca este foarte greu sa se gasesca 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

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)

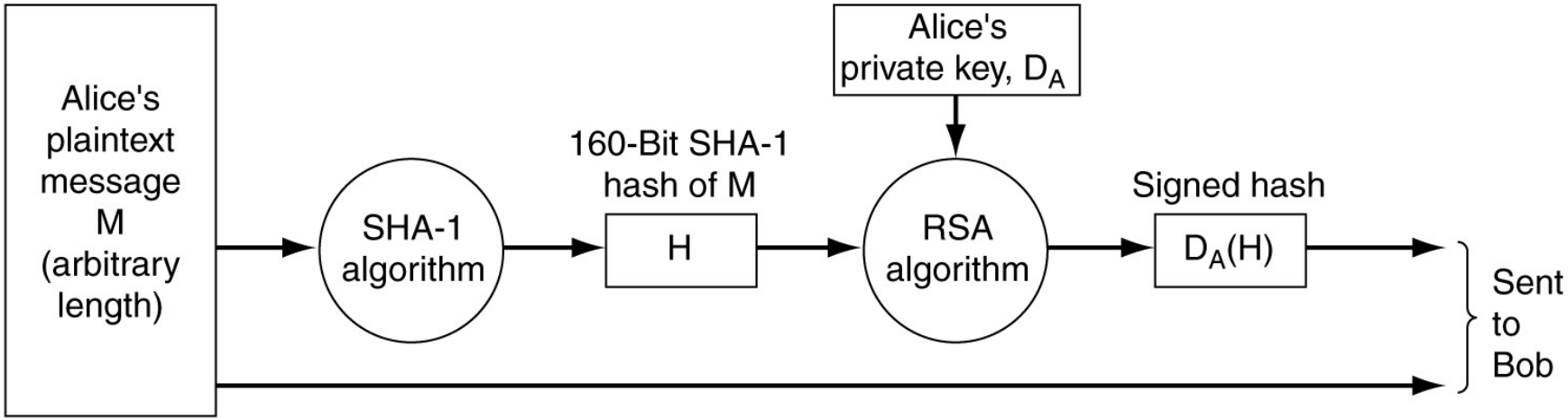
Comentarii

t si 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.

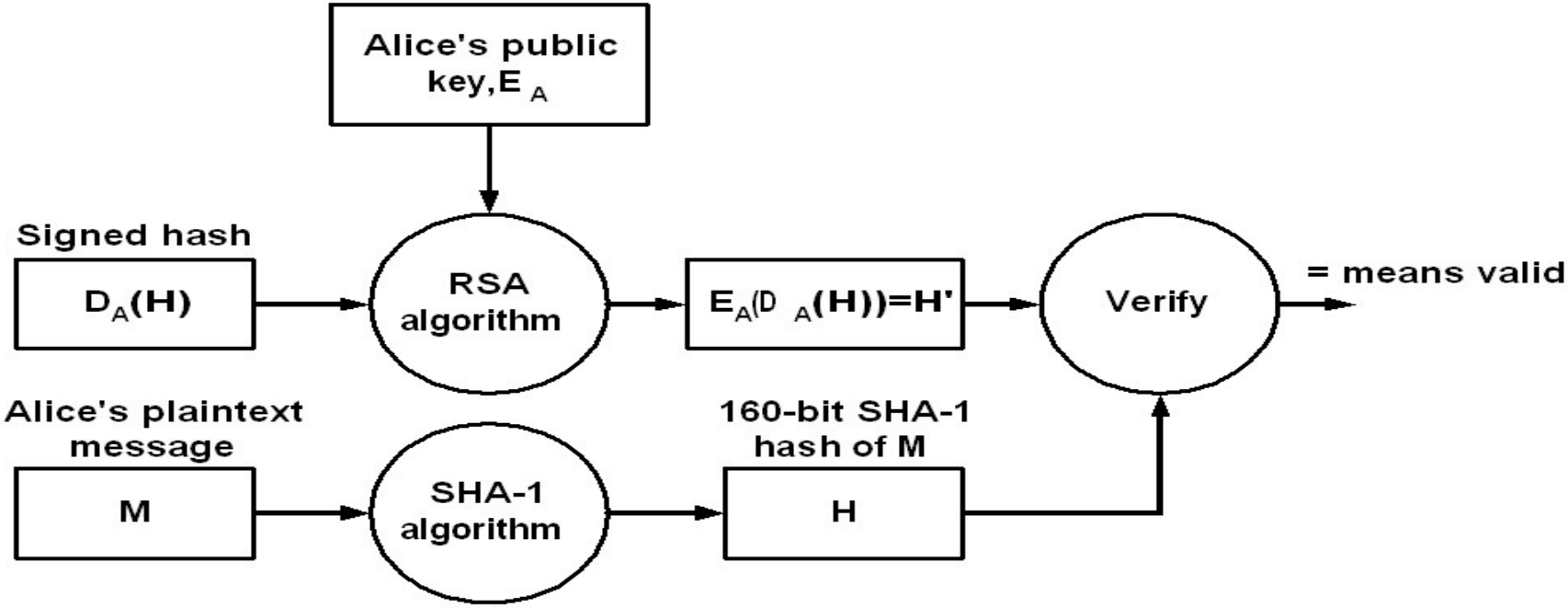


Caracteristici

Rezumatul SHA-1 este semnat cu cheia secreta a transmitatorului D_A

Mesajul M este transmis in clar

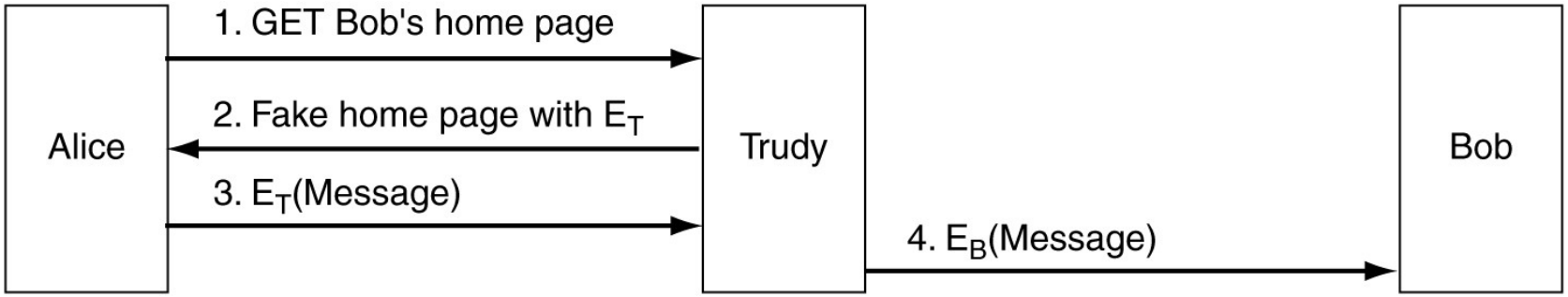
Verificare semnatura digitala



Orice modificare a textului clar M este detectata prin H<>H'

Un intrus nu poate modifica si M si 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 digitale

Certificate de securitate

- Certificate
  - Asociaza identitatea cu cheia publica
- X.509
  - Standard de certificate

Certificate

Rol: leaga cheia publica de un proprietar (principal) sau de un atribut.
Un certificat permite autentificarea unei entitati in cazul in care nu a mai fost vreun contact anterior.

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@superdupernet.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 cripteaza 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	**Semnificatie**
Versiune	Ce versiune de X.509 este utilizată
Serial Number	Acest număr împreună numele CA-ului identifică în mod unic certificatul
Algoritm de semnare	Algoritmul folosit la semnarea certificatului (ex. MD5 cu RSA)
Emitent	Numele X.500 al CA-ului
Perioada de validitate	Momentele de început si sfârşit ale perioadei de validitate
Numele subiectului	**Entitatea care este certificată**
Cheia publică	**Cheia publică a subiectului şi ID-ul algoritmului folosit (ex. RSA)**
ID emitent	Un ID opţional identificând în mod unic emitentul certificatului (nume X.500 sau DNS)
ID subiect	Un ID opţional identificând în mod unic subiectul certificatului
Extensii	ptr identificarea cheii publice a emitentului, a certificatului care contine o anumita cheie publica, scopul utilizarii cheii (criptare, semnare,…) si altele
Semnătura	**Semnătura certificatului (semnat cu cheia privată a CA-ului)**

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

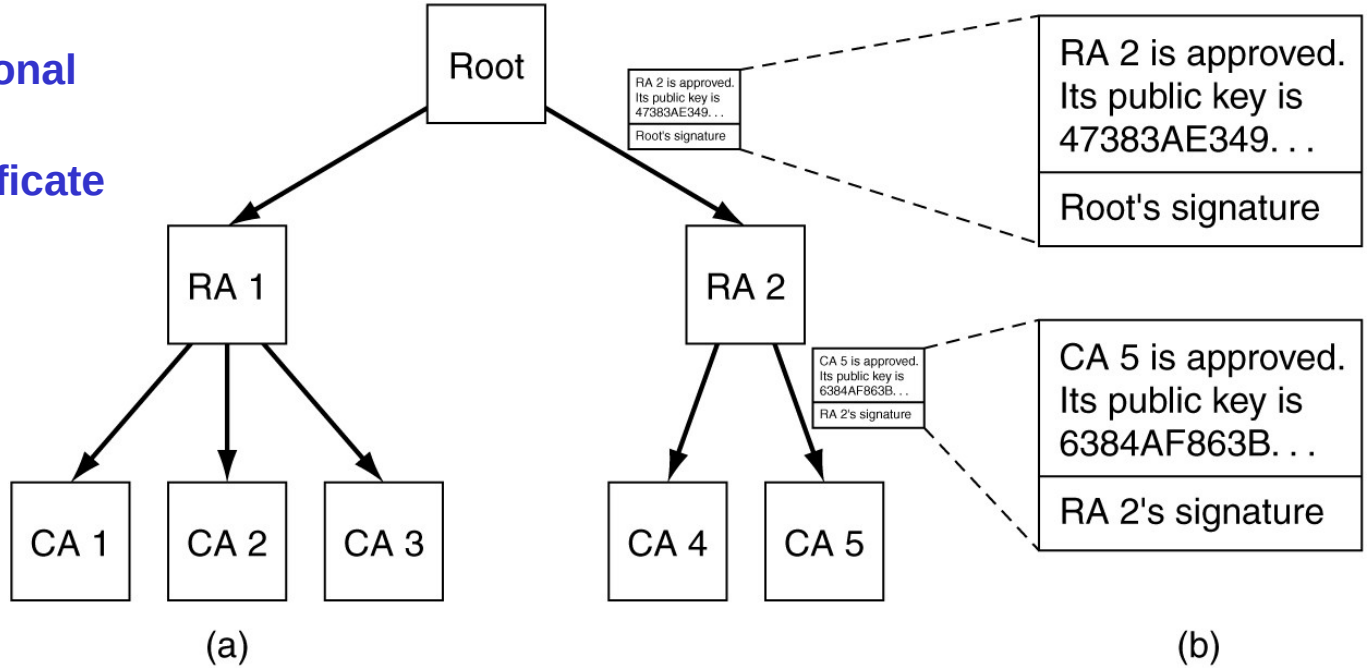
- autoritate de incredere care elibereaza certificate
  - atestă că cheia publica inclusa apartine persoanei cu numele atasat

- 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 primeste de la B tot lantul de certificate

Revocarea Certificatelor

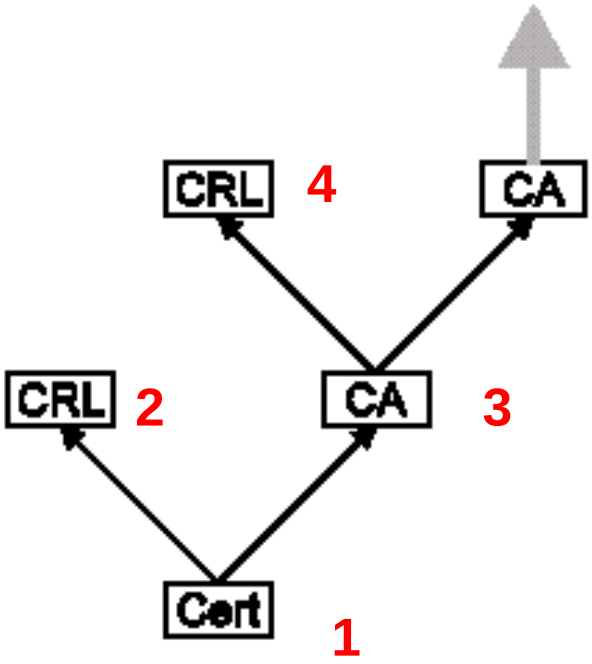
- Un certificat trebuie **revocat** cand:
  - cheia primara este **compromisa**;
  - cheia primara este **pierduta**;
  - o persoana pleaca din companie
  - altele.

- Revocarea trebuie anuntata tuturor utilizatorilor – dificil !
- Alternativa - se folosesc liste de revocare
  - **CRL – Certificate Revocation List**;

Metoda

- se verifica listele 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

Protocoale de Securitate

Securitatea Comunicatiei

- IPsec
- 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
  - parmetrii 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) si

- **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**

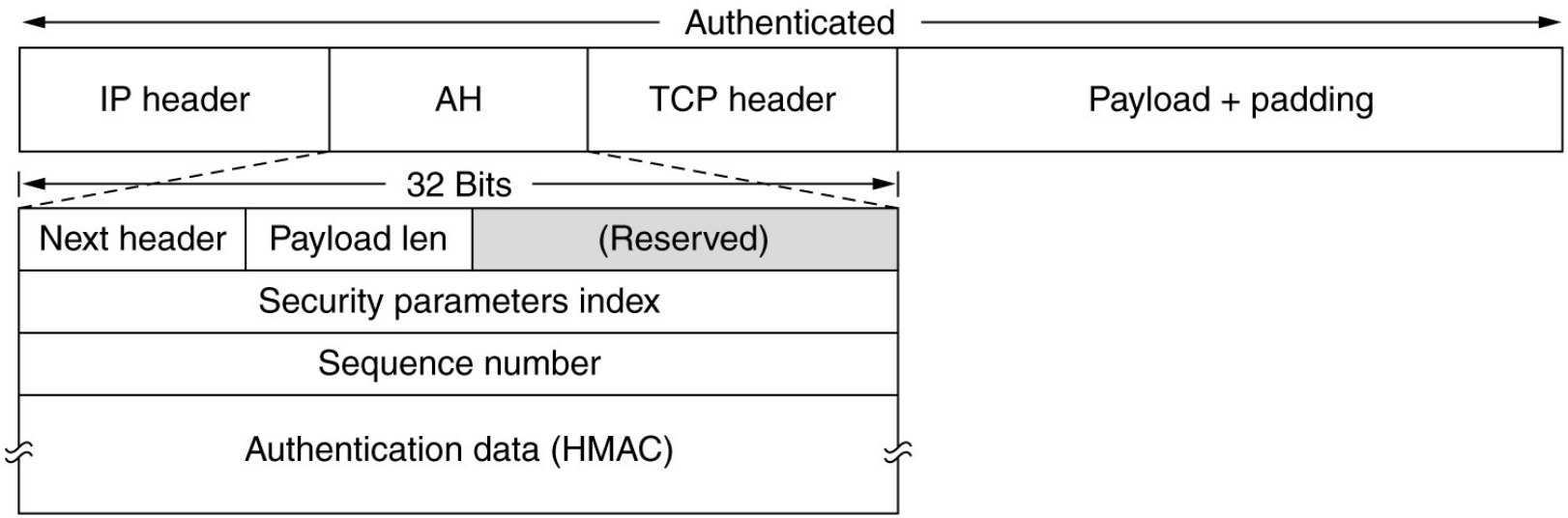
Doua protocoale de securitate:

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

Si doua moduri de lucru

- transport
- tunel

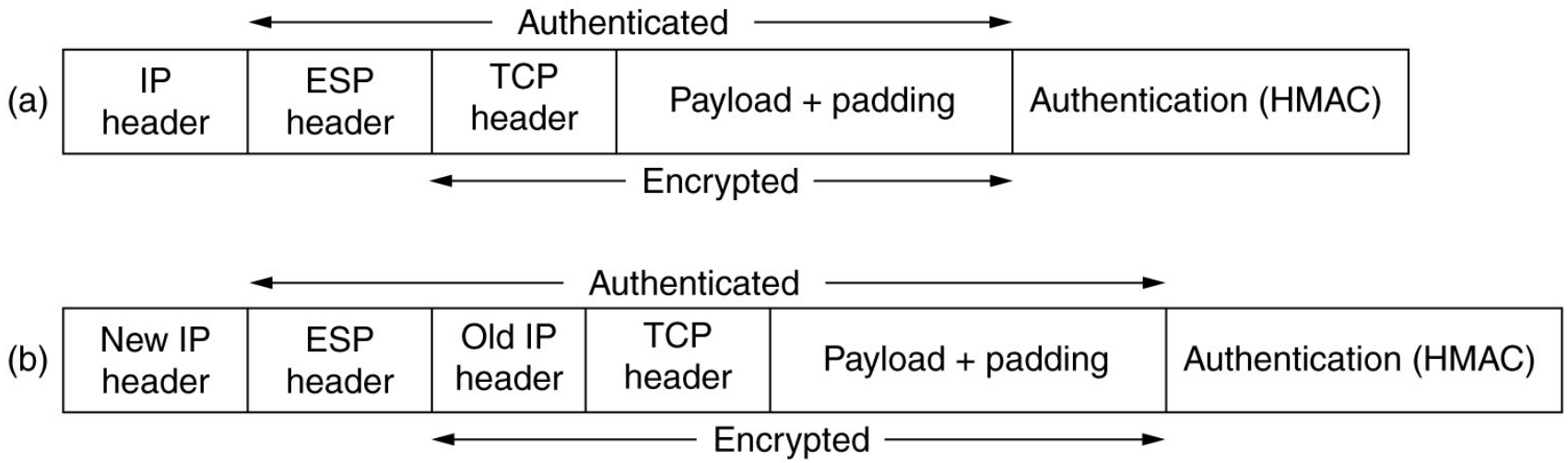
Protocol AH – in mod transport pentru IPv4



Authentication Header – inserat in datagrama IP

- Next header – preluat din IP header unde este inlocuit cu 51
- Payload len – lungime AH (nr cuvinte 32 biti) minus 2
- Security Parameters Index – indica inregistrarea din BD a receptorului
- Sequence number - evitare atacuri prin replica
- HMAC – Hashed Message Authentication Code
    - Utilizeaza cheia simetrica
    - Calculeaza rezumat peste intreaga datagrama (campurile variabile neincluse) + cheia simetrica

ESP in modurile transport si tunel



ESP – Encapsulating Security Payload

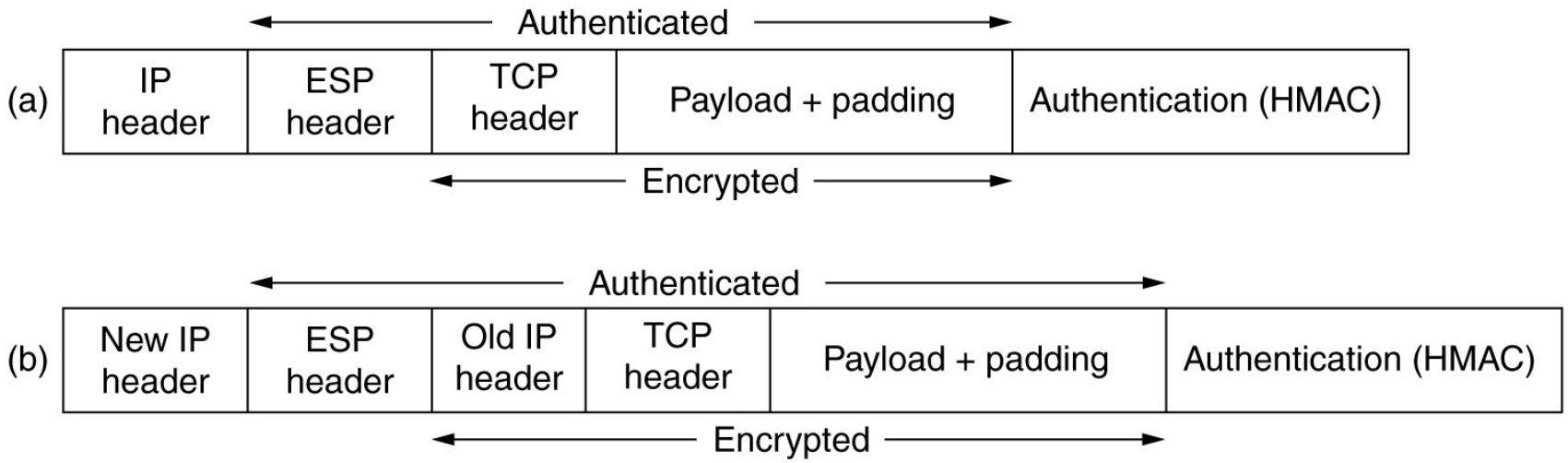
(a) ESP in mod transport.

- 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 Secventa

Vector de initalizare (pentru criptare date)

La sfarsit: HMAC – Hashed Message Authentication Code

HMAC

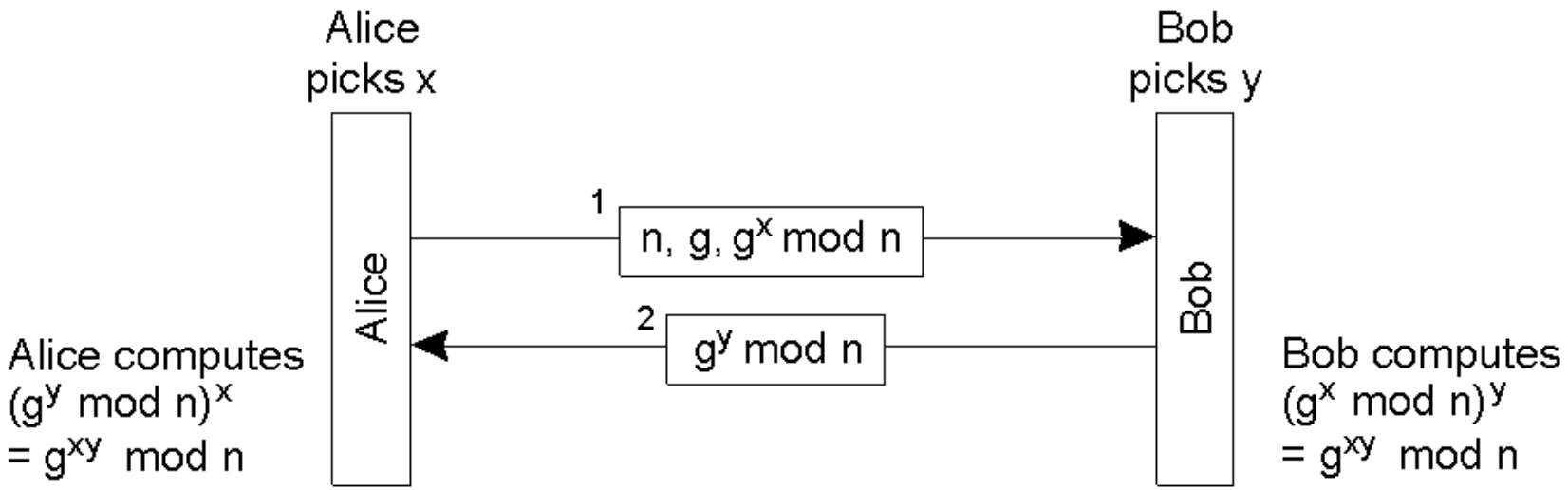
- HMAC = hash(MAC_write_secret XOR pad_2 |
- hash(MAC_write_secret XOR pad_1 | **MSG**))

- Functii hash suportate:
  - MD5
  - SHA-1

- pad_1 este 0x36 repetat
- pad_2 este 0x5C repetat

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
  - $g^x \bmod n$ este cheia publica
  - $K_{A,B} = g^{xy} \bmod n$ este cheia secreta partajata cu Bob

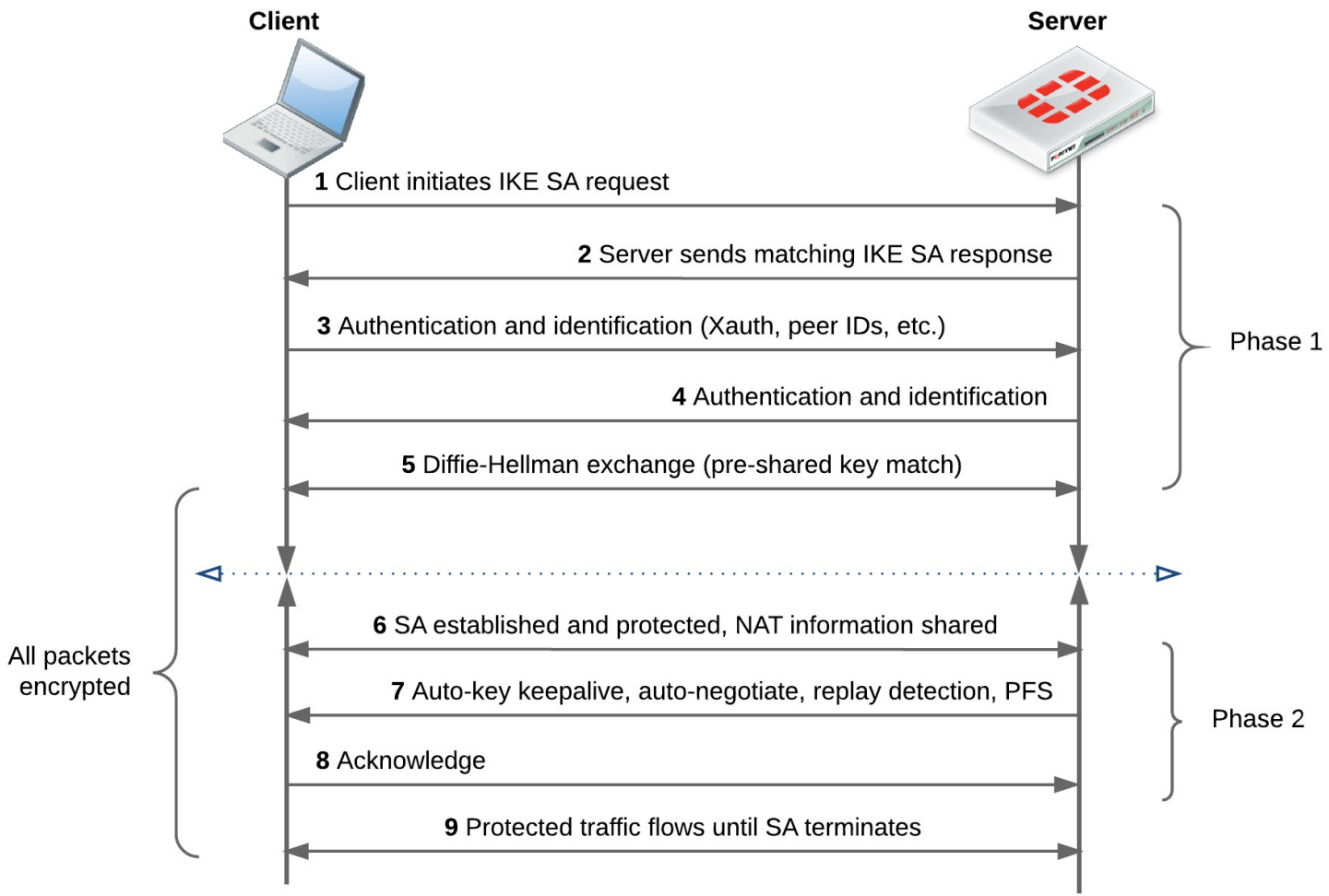


IKE

- IKEv1
  - Phase 1:
    - Clientul trimite o cerere de tipul IKE Security Association (SA)
    - Se trimit identitatile (ID-uri, certificate, etc.)
    - Include schimbul de chei Diffie-Hellman
  - Phase 2:
    - SA este stabilit si securizat
    - Parametrii SA sunt renegociati dupa un interval fix de timp

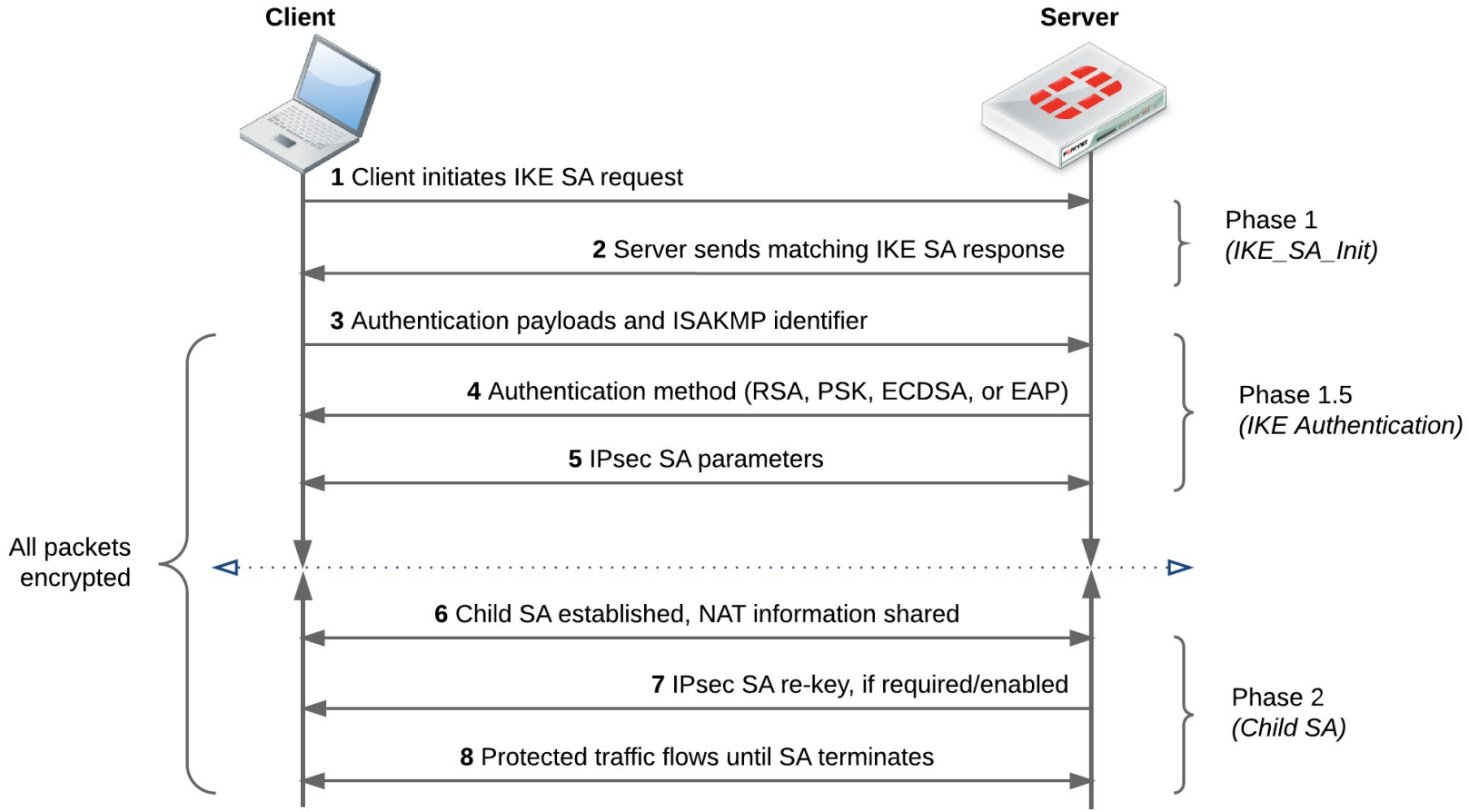
IKE

- IKEv1


Client
Server
1 Client initiates IKE SA request
2 Server sends matching IKE SA response
3 Authentication and identification (Xauth, peer IDs, etc.)
4 Authentication and identification
5 Diffie-Hellman exchange (pre-shared key match)
Phase 1
6 SA established and protected, NAT information shared
7 Auto-key keepalive, auto-negotiate, replay detection, PFS
8 Acknowledge
9 Protected traffic flows until SA terminates
Phase 2
All packets encrypted

IKE

- IKEv2
  - Pasul de autentificare este criptat
  - Poate include protocoale de autentificare diferite
  - Autentificarea este un pas separat (1.5)

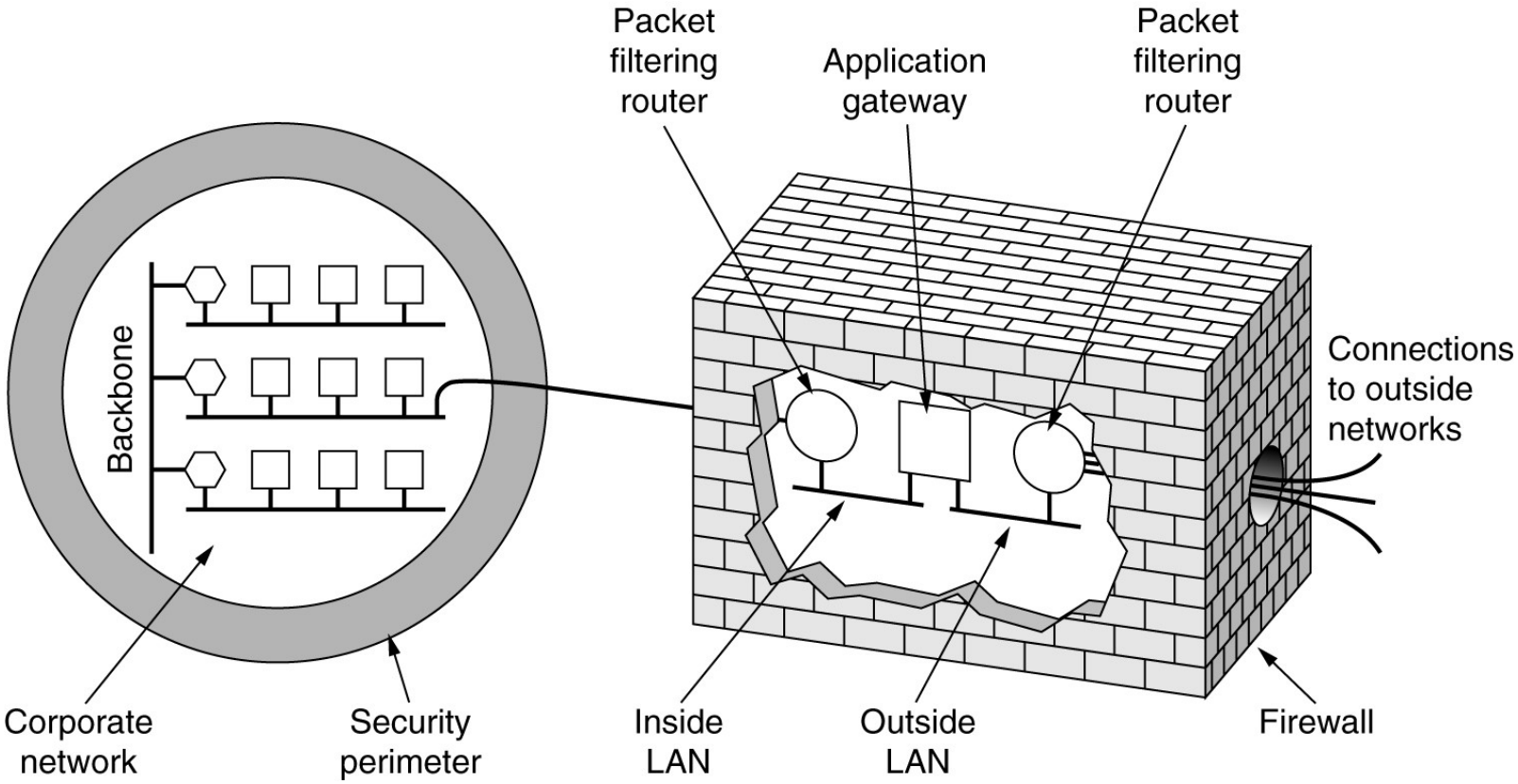


Caracteristici Protocol IPSEC

- 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, …

Firewall-uri

- Firewall: are posibilitatea de a bloca sau de a modifica traficul la nivelul retea



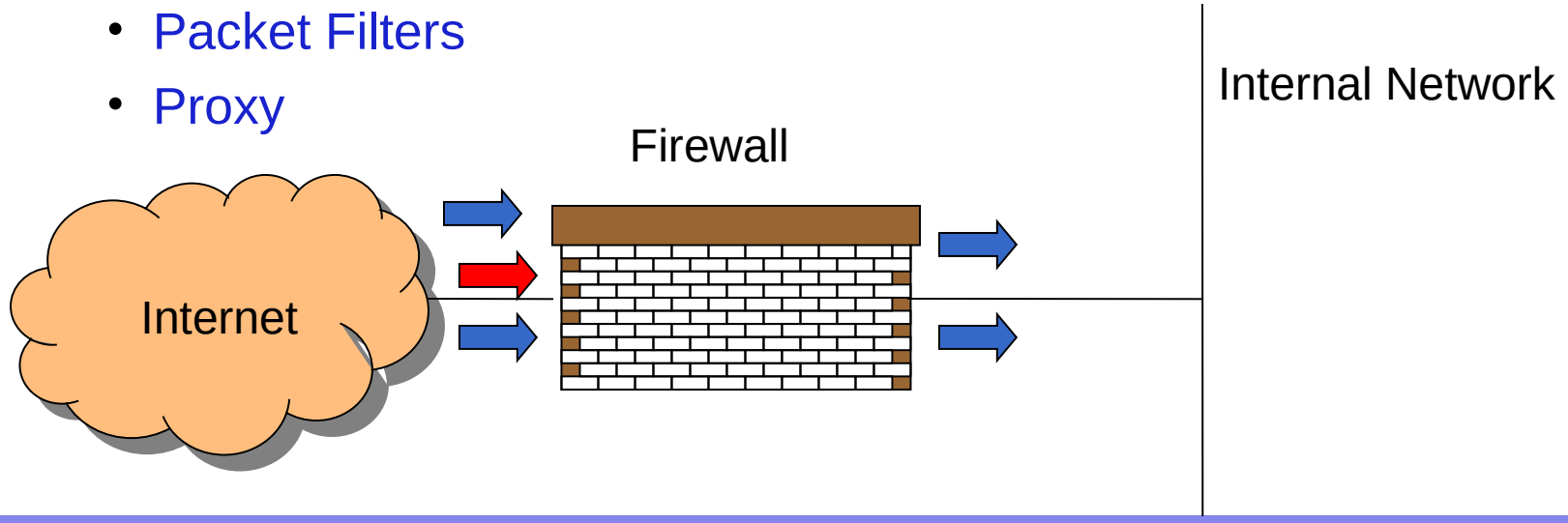
Firewall (1)

Functionalitate firewall:

- Inspecteaza traficul ce trece prin el
- Permite doar trecerea traficului de retea ce corespunde anumitor reguli
- Orice altceva este blocat

2 tipuri de firewall:

- Packet Filters
- Proxy



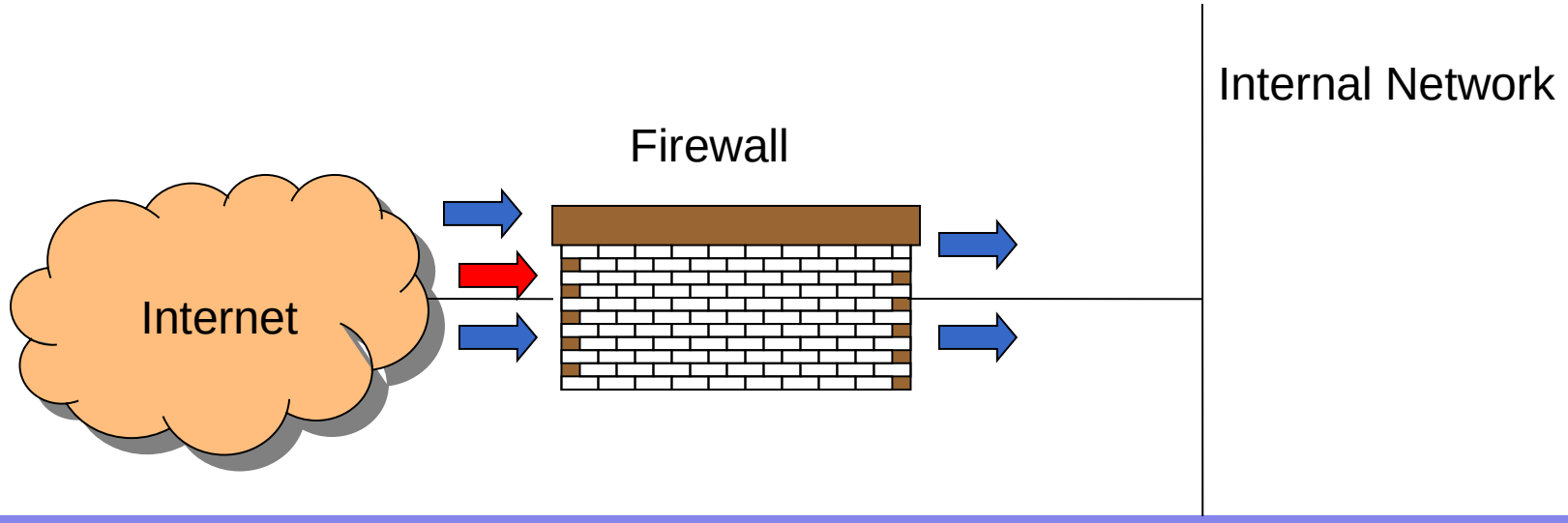
Firewalls (2)

- Foloseşte reguli la nivel reţea.
- Filtrare după:
  - Adrese origine şi destinaţie.
  - Protocol (TCP sau UDP).
  - Numere port.
- Nepotrivit pentru medii care cer analiza mai detaliată pentru protocoale de nivel superior (adică proxy servers).

Firewall (3)

Actiuni posibile:

- Permite trecerea pachetului
- Drop
- Alterarea pachetului (NAT?)
- Logging



Exemple reguli

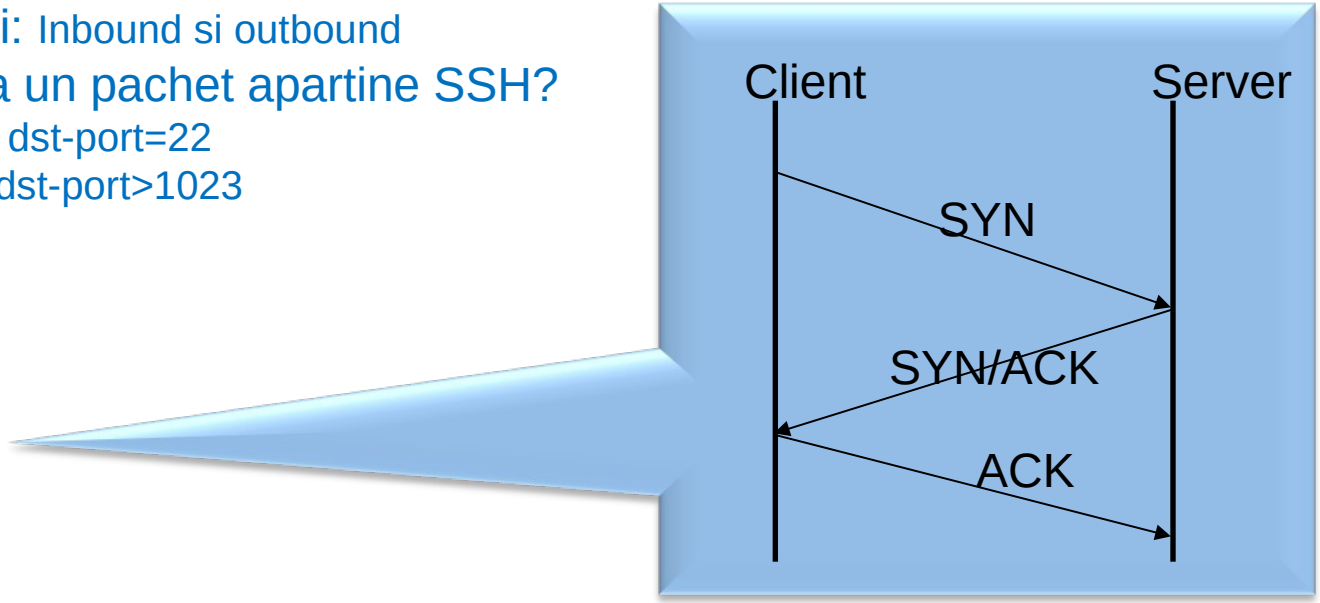
FROM <target a> TO <target b> <action> <protocol> <port>

- Blocheaza toate pachetele din exterior, mai putin pentru anumite protocoale
- Blocheaza tot traficul dintr-o lista de domenii
- Blocheaza tot traficul spre anumite domenii

- Ingress filtering
  - Toate pachetele din exterior spre interiorul retelei vor fi aruncate
- Egress filtering
  - Toate pachetele din reteaua locala spre adrese din exterior vor fi aruncate

Exemplu reguli firewall

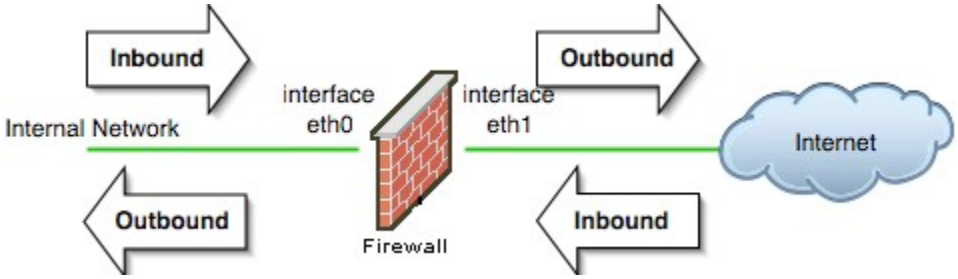
- Permite conexiuni SSH de la host-uri externe spre host-uri interne
- Este nevoie de 2 reguli: Inbound si outbound
- Cum se poate stii daca un pachet apartine SSH?
  - Inbound: src-port>1023, dst-port=22
  - Outbound: src-port=22, dst-port>1023
  - Protocol=TCP
- Ack bit Set?



Rule	Dir	Src Addr	Src Port	Dst Addr	Dst Port	Proto	Ack Set?	Action
SSH-1	In	Ext	> 1023	Int	22	TCP	Any	Allow
SSH-2	Out	Int	22	Ext	> 1023	TCP	Yes	Alow

Firewall rule example (default)

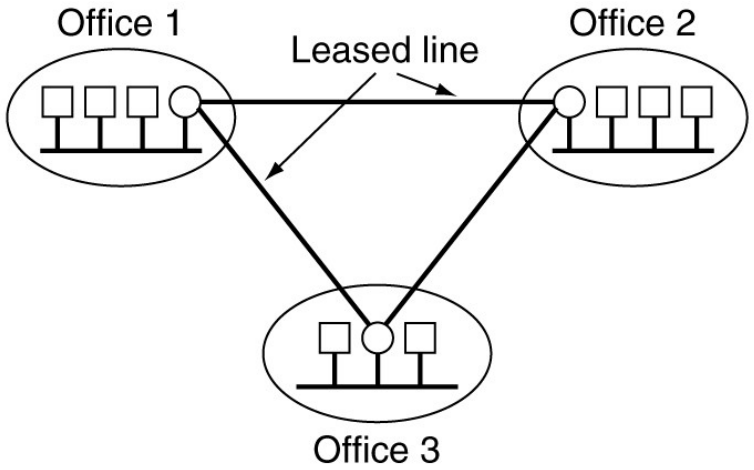
- Egress Filtering
  - Outbound traffic from external address Drop
- Ingress Filtering
  - Inbound Traffic from internal address Drop
- Default Deny



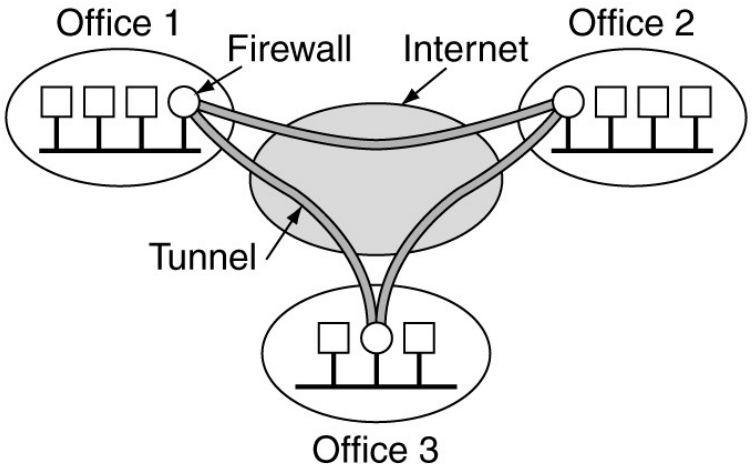
Rule	Dir	Src Addr	Src Port	Dst Addr	Dst Port	Proto	Ack Set?	Action
Egress	Out	Ext	Any	Ext	Any	Any	Any	Deny
Ingress	In	Int	Any	Int	Any	Any	Any	Deny
Default	Any	Any	Any	Any	Any	Any	Any	Deny

Virtual Private Networks

- Reţea privată:
  - (a) Cu linii închiriate.
  - (b) VPN.
- VPN se construiesc peste Internet:
  - Fiecare birou are un firewall, se creează tunele între firewalls.
  - IPsec folosit pentru tunneling (ESP în mod tunel).
  - În Internet pachetele apar ca şi cele obişnuite.
  - VPN este transparent pentru software de aplicaţii.



(a)



(b)

Protocoale de Autentificare

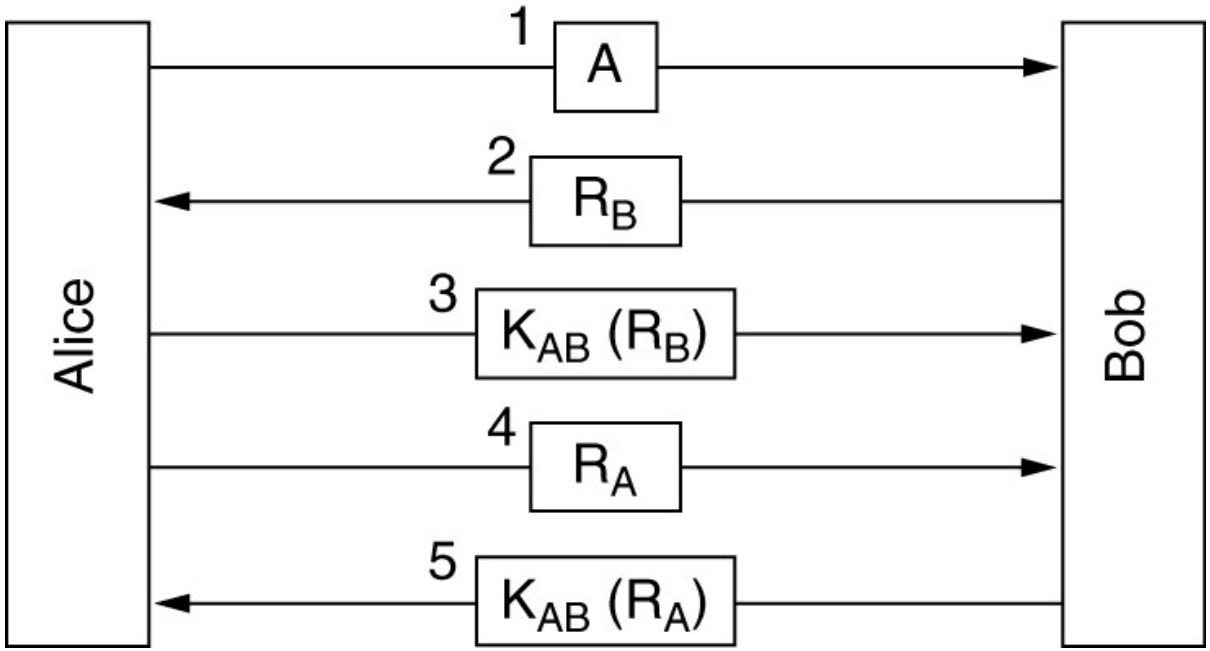
Determina daca o entitate (utilizator, proces) este cu adevarat cine / ce pretinde ca este

- 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

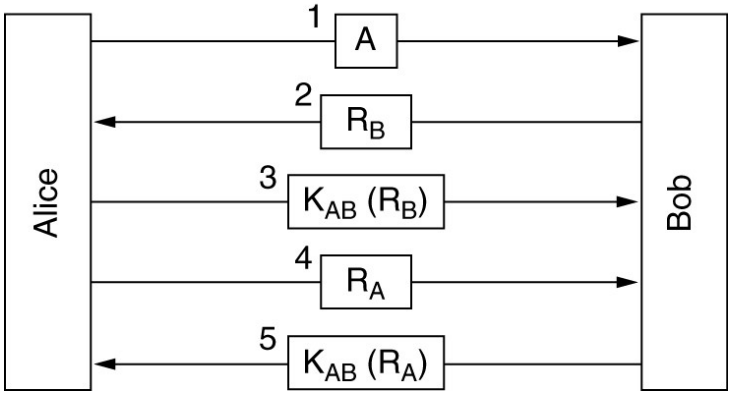


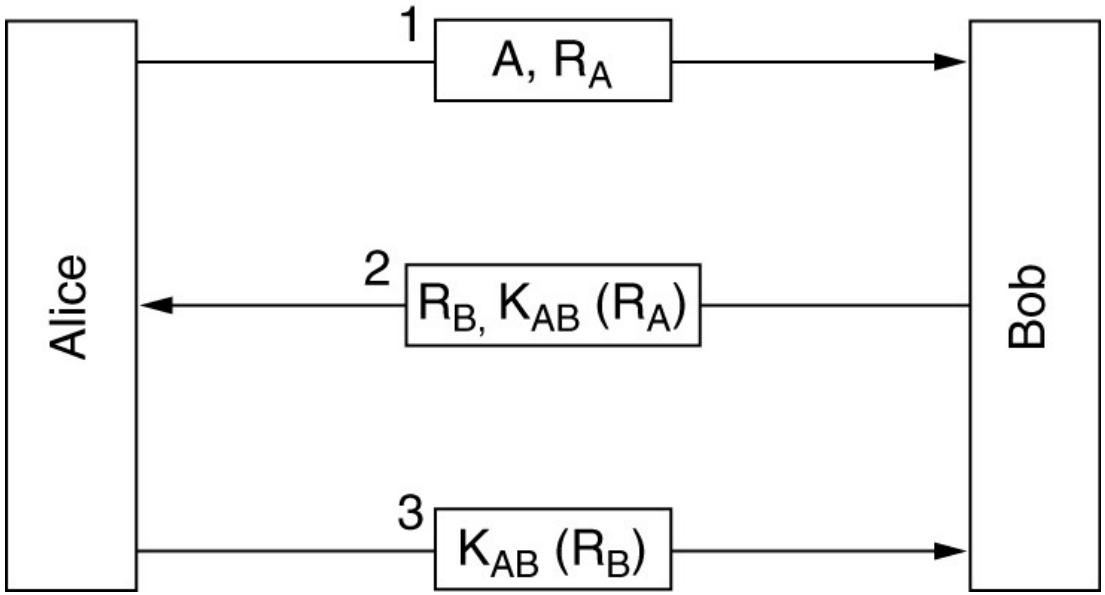
Autentificare reciproca cu un protocol challenge-response

Alice si Bob partajeaza cheia $\mathbf{K_{AB}}$

$\mathbf{R_A}$, $\mathbf{R_B}$ - numere aleatoare foarte mari, folosite contra atac prin replica

Autentificare cu cheie secreta partajata (2)





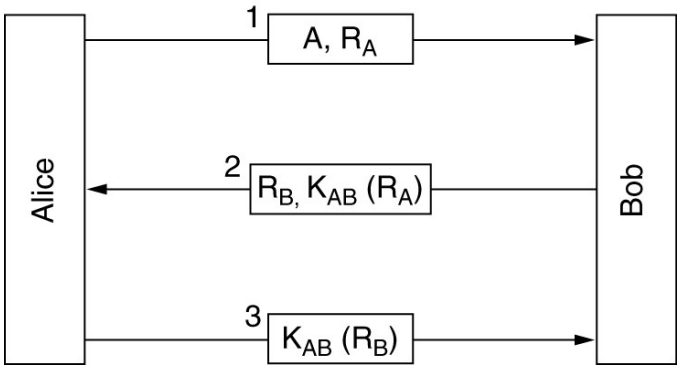
Reducere numar de pasi

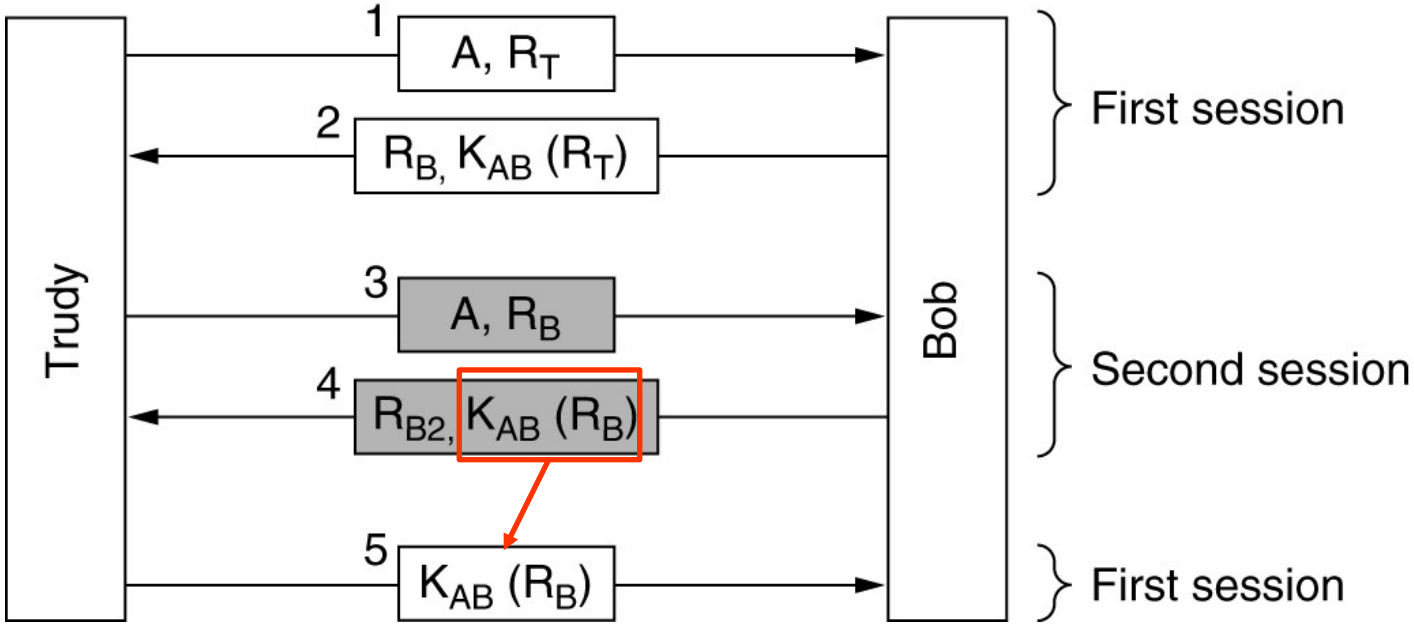
Atacul prin reflexie

Trudy nu poate cripta R_B din mesajul 2

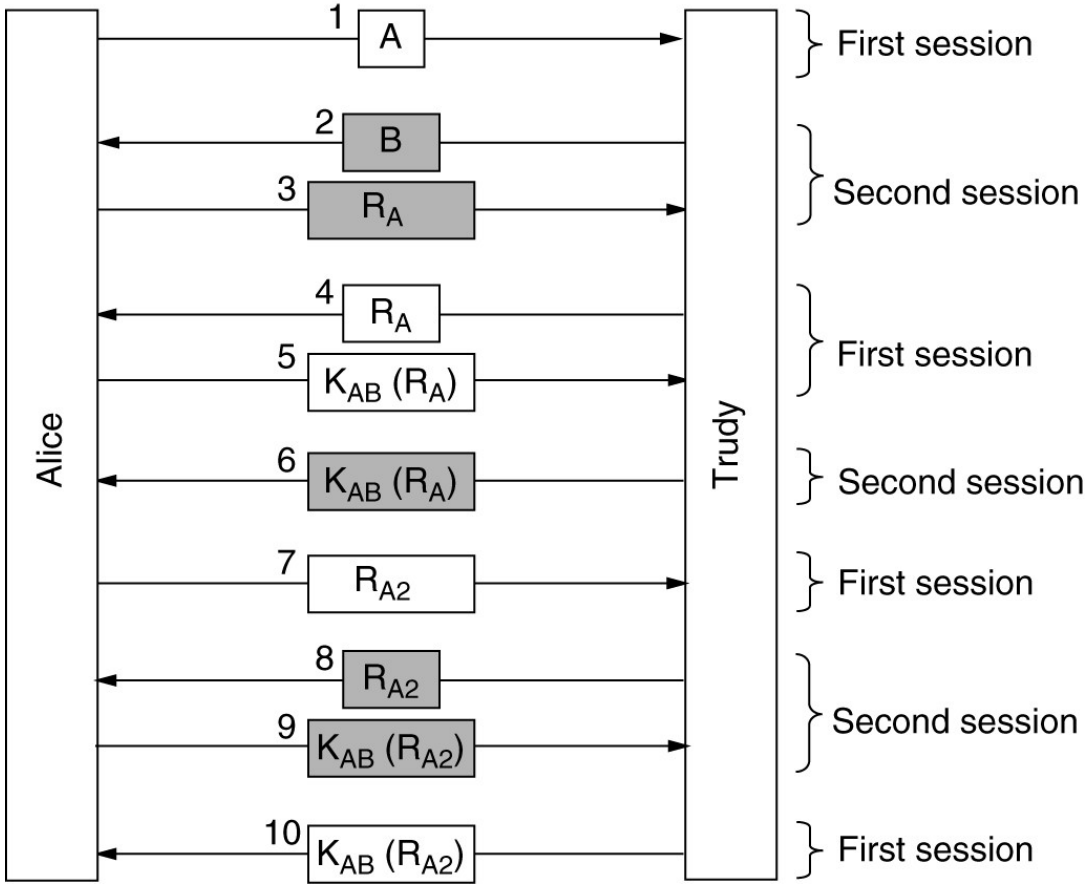
Dar retransmite un mesaj produs de Bob (4) si

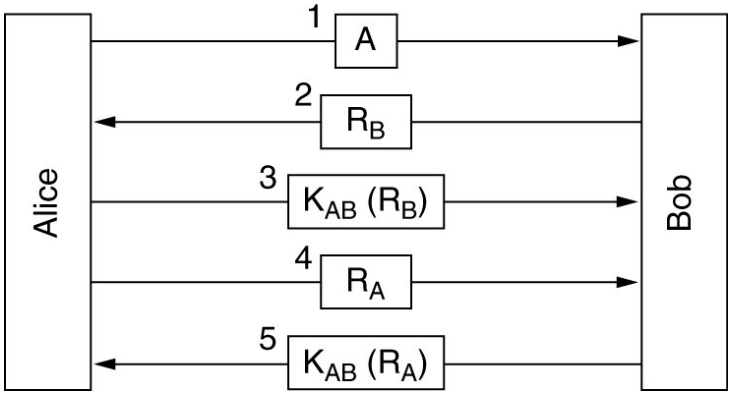
reuseste sa stabileasca o sesiune autentificata cu Bob





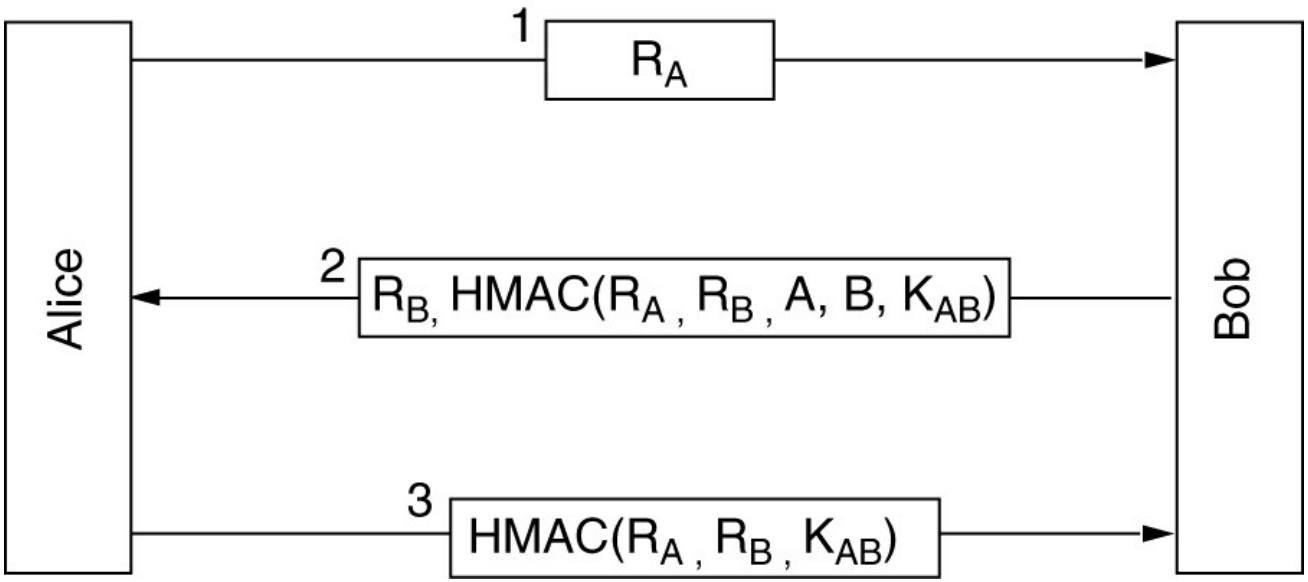
Atacul prin reflexie pe protocolul initial





Refolosind mesajele 5 si 9, Trudy reuseste sa stabileasca doua sesiuni autentificate cu Alice

Autentificarea cu HMAC



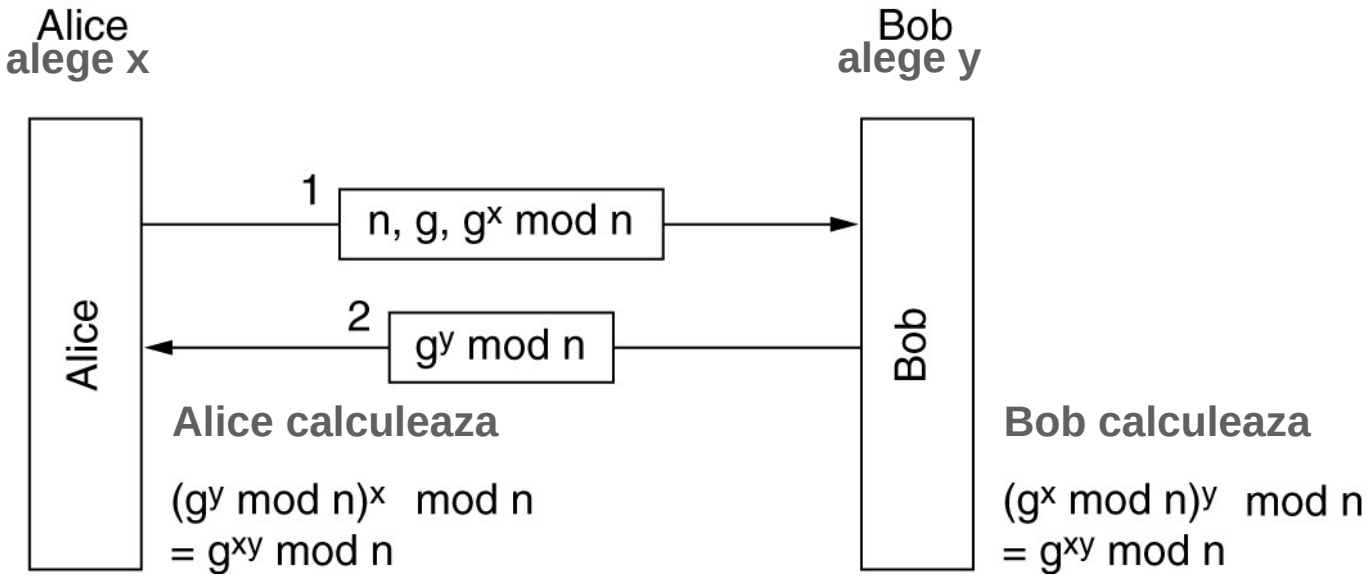
Alice si Bob partajează cheia K_{AB}

Fiecare parte poate calcula rezumatul HMAC - Hashed Message Authentication Code – (deoarece contine 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

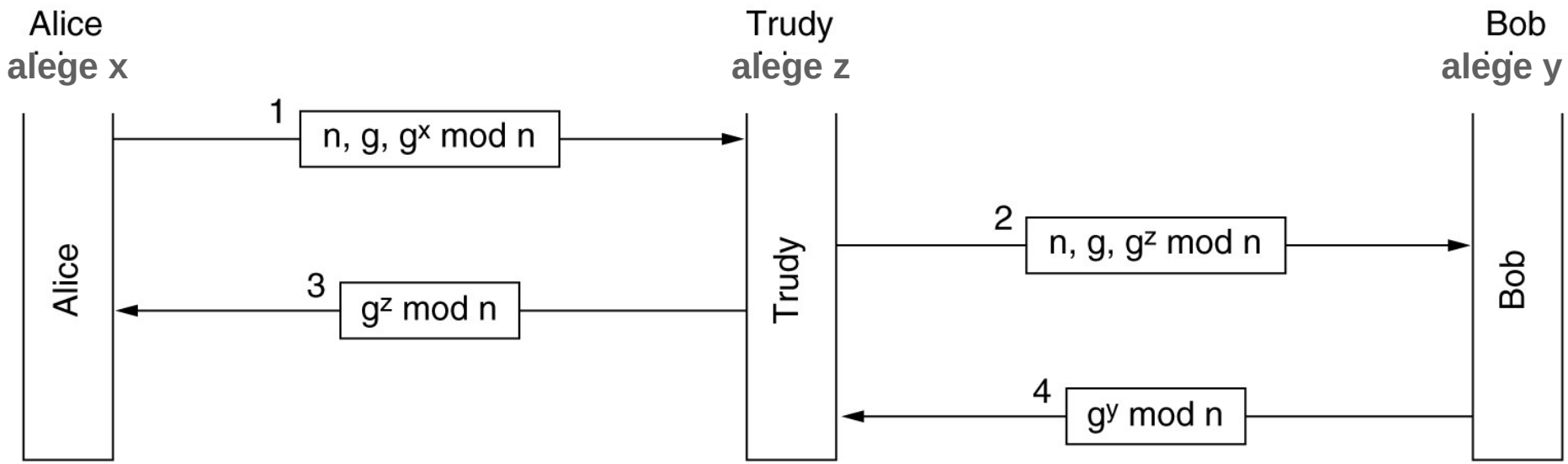
x nu poate fi calculat din g^x mod n

g^{xy} mod n nu poate fi calculat din g^x mod n si g^y mod n cand n este mare

g < n (generator) are proprietatea: orice p poate fi scris ca g^k mod n

adica: pentru fiecare *p* intre 1 si *n*-1 inclusiv, exista o putere *k* a lui *g* astfel ca $p = g^k \bmod n$.

Atacul man-in-the-middle

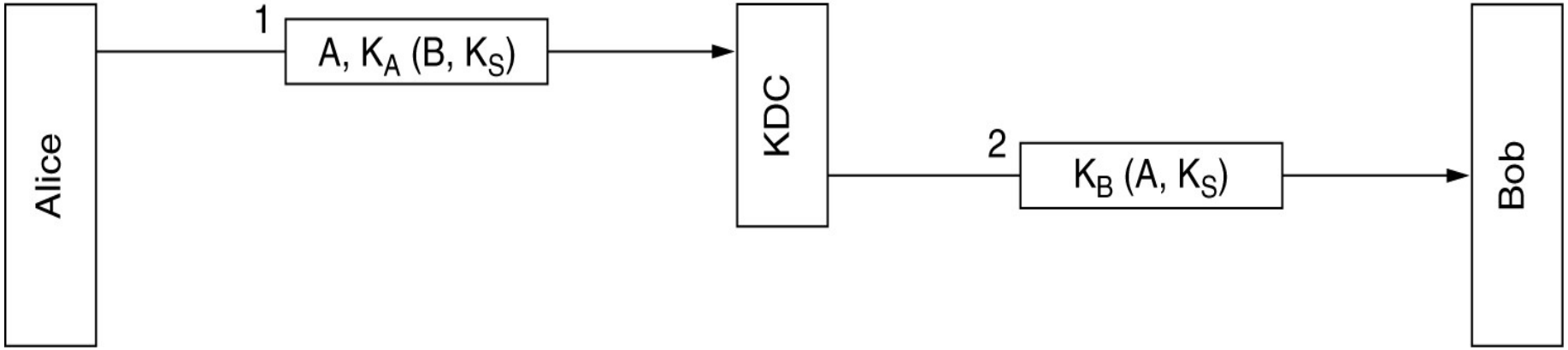


Vulnerabilitate – **g** si **n** sunt publici

- permite stabilirea a doua chei: intre Alice si Trudy - $\mathbf{g^{xz}}$ **mod n** si intre Trudy si Bob - $\mathbf{g^{zy}}$ **mod n**

Rezolvare: Alice si Bob semneaza mesajele schimbate intre ei
Trudy nu poate modifica mesajele

Autentificarea folosind Key Distribution Center



Alice si Bob folosesc un Centru de distributie a cheilor

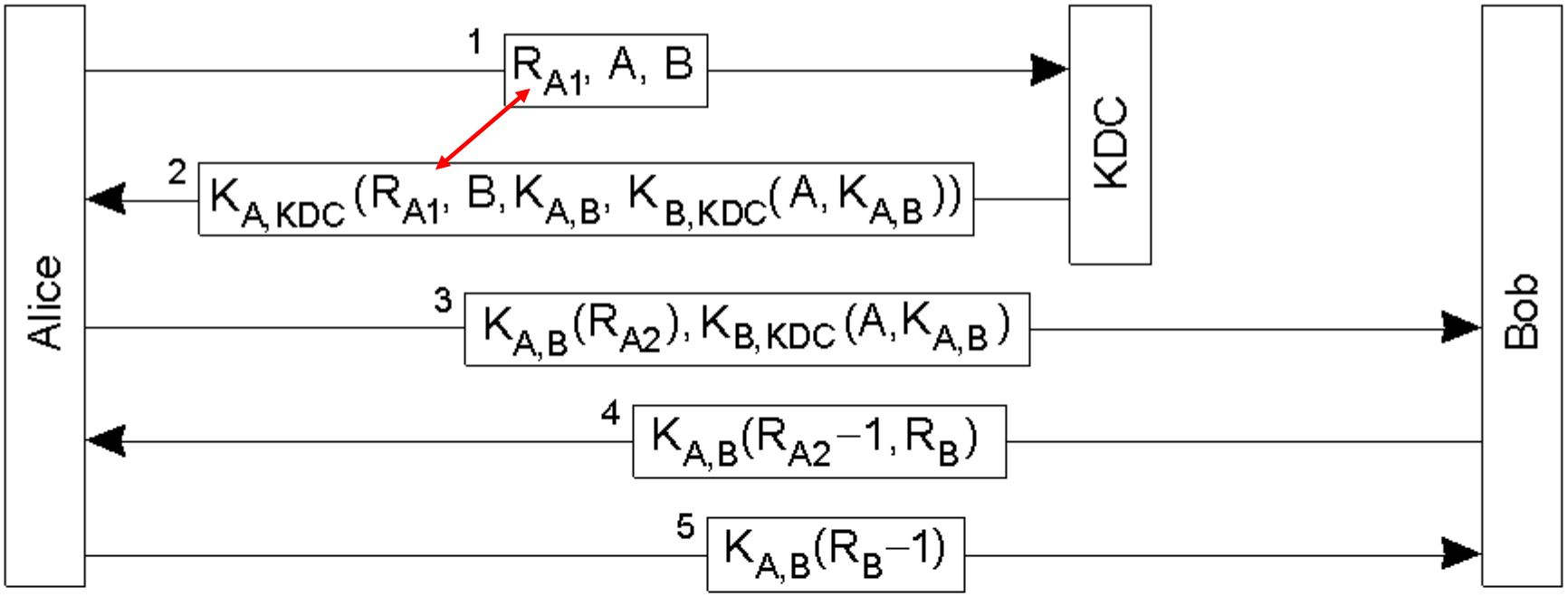
- in care au incredere
- cu care impart cheile secrete $\mathbf{K_A}$ respectiv $\mathbf{K_B}$

Prima incercare, vulnerabila la replay attack

Trudy **retransmite** mesajul 2 si

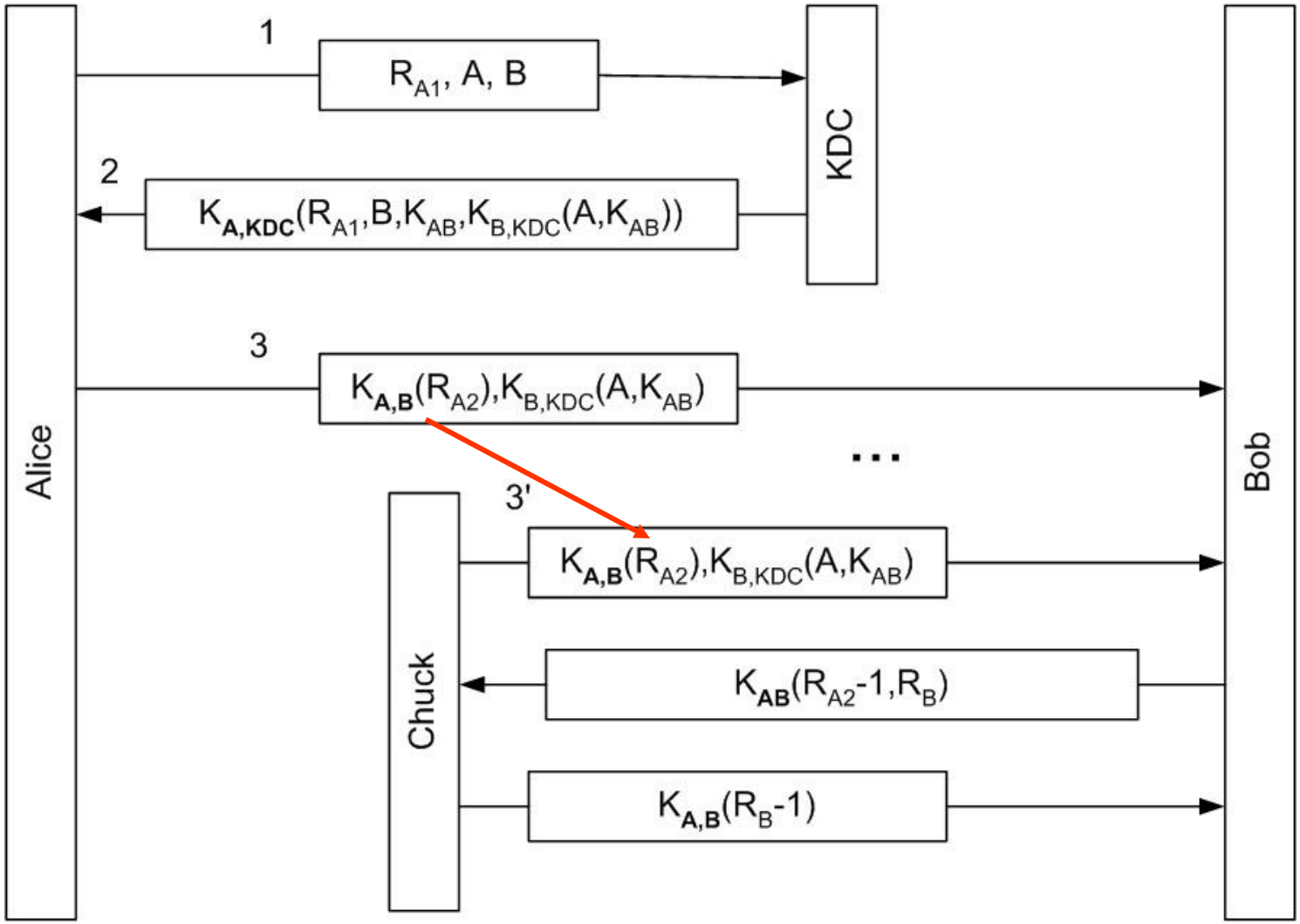
mesajul asociat cu el, criptat deja cu $\mathbf{K_S}$ (de ex. extragerea din contul lui Alice a unei sume de bani)

Autentificarea cu protocolul Needham-Schroeder



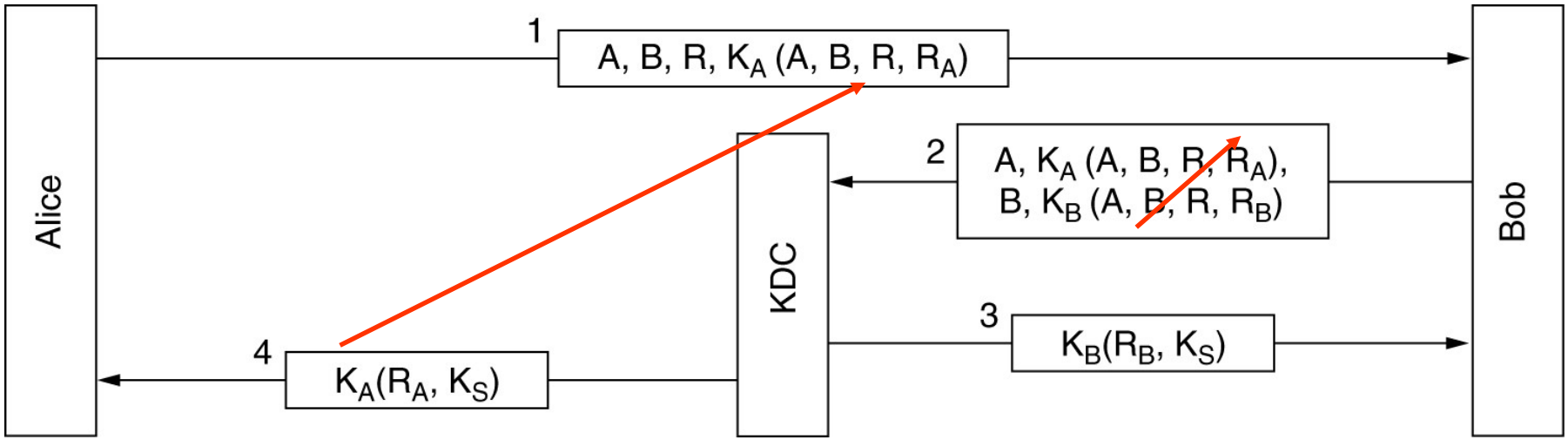
- foloseste tichete - ex. $K_{B,KDC}(A, K_{AB})$
  - Alice nu poate intelege sau modifica tichetul, Bob poate
  - Bob capata incredere in cheia K_{AB} (care vine de la KDC)
- numere aleatoare (nonce) ex. R_{A1}, folosite contra atac prin replica
  - ex. Alice afla ca mesajul 2 este un raspuns la 1, nu un mesaj rejucat de Trudy

Slabiciune Needham-Schroeder



Chuck afla cheia K_{AB} si retrimite mesajul 3, pretinzând ca e Alice

Autentificarea folosind Protocolul Otway-Rees



Protocolul Otway-Rees (simplificat).

KDC trimite cheia de sesiune K_S dupa ce verifica daca identificatorul comun R apare in ambele parti criptate ale mesajului 2

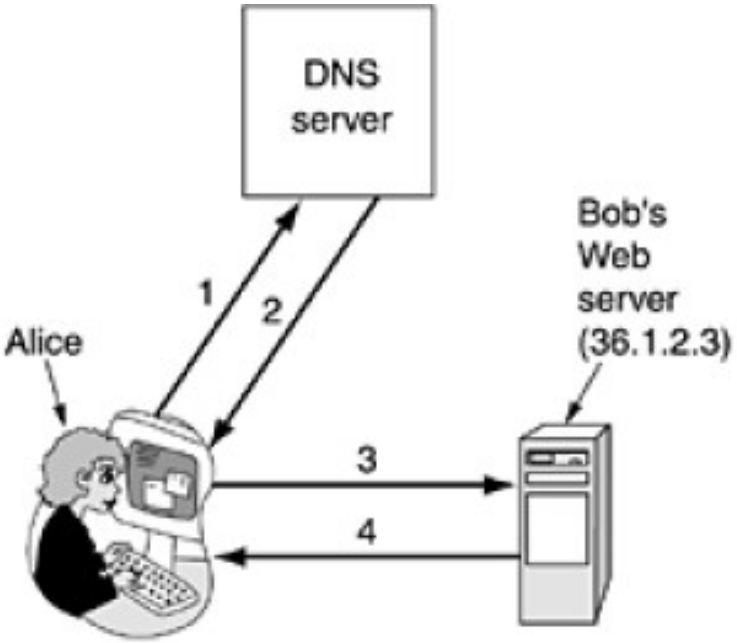
R_A, R_B – numere aleatoare folosite de KDC in mesajele 3 si 4 pentru a face legatura cu mesajele 1 si 2

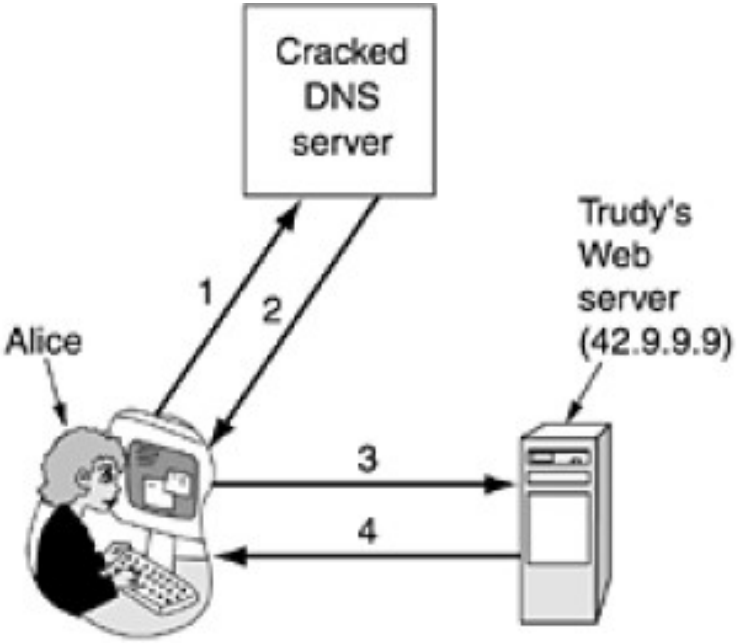
Problema: Alice ar putea folosi cheia secreta inainte ca Bob sa afle de ea

Securitatea Web

- Atacuri
  - inlocuire Home page
  - Denial-of-service
  - Citire mail-uri
  - Furt numere credit card
- Solutii
  - Secure Naming
  - SSL – The Secure Sockets Layer

Necesitate Secure Naming



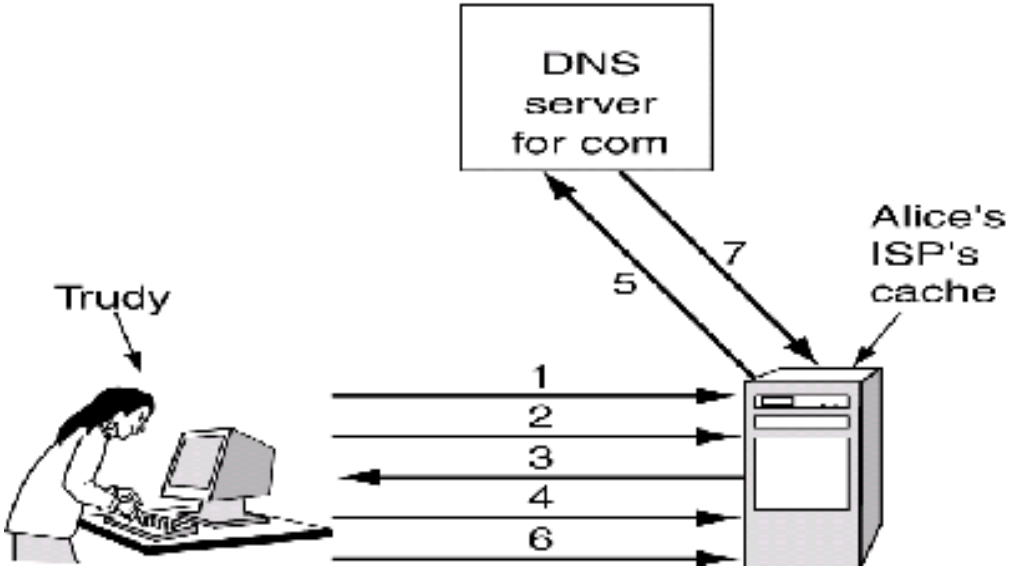


Situatie Normala.

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 inregistrarii lui Bob in 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

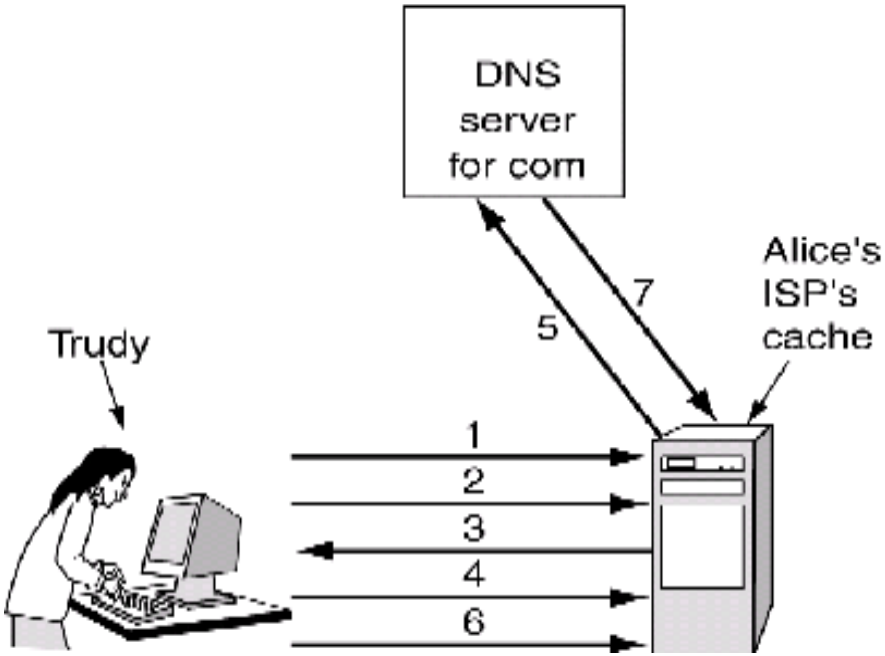
Probleme: ISP verifică adresa IP de la care vin răspunsurile DNS

- Trudy poate folosi adresa IP a unui server DNS de nivel inalt (care este publica) pentru a construi un raspuns fals

DNS se bazeaza pe UDP DNS foloseste

sequence numbers pentru a mapa cererile si raspunsurile

- Trudy inregistreaza un domeniu ***trudy-the-intruder.com*** (IP 42.9.9.9) si
- Creaza un server DNS ***dns.trudy-the-intruder.com*** (aceeasi IP 42.9.9.9)

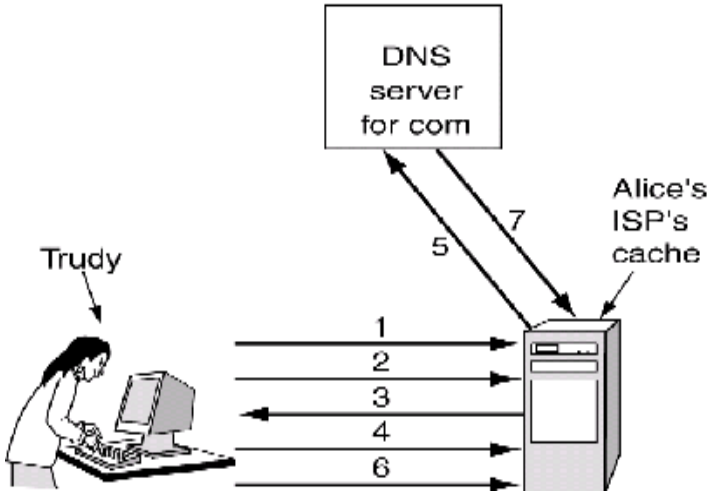


Trudy pacaleste ISP-ul lui Alice (2)

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 secventa, **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

Inregistrarile 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 publica a unei zone, utilizator, host, etc.

***SIG* record** - **hash** semnat al inregistrarilor *A* si *KEY* pentru verificarea autenticitatii.

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)

Analiza algoritmilor criptografici (Optional)

Problema:

Pot criptogramele interceptate de un intrus sa faciliteze spargerea confidentialitatii criptografice ?

O solutie este **Teoria Informatiei**

- care masoara cantitatea medie de informatie transmisa de o sursa
- si, echivalent, cantitatea de incertitudine inlaturata (in medie) de un mesaj

Bazata pe notiunile de entropie si entropie conditionata

Entropia

Entropia este cantitatea medie de informatie transmisa de o sursa

Fie S o sursa de informatii

- care transmite mesajele $X_1,..., X_n$

cu probabilitatile $p(X_1), ..., p(X_n)$, ptr. care $\sum_{i=1,n} p(X_i) = 1$.

Entropia H(X) este:

$$\mathbf{H(X) = \sum_{i=1,n} p(X_i)*\log (1/p(X_i)) = -\sum_{i=1,n} p(X_i)*\log (p(X_i))}$$

- $\log(1/p(X_i))$ = cantitatea de informatie primita la receptia lui X_i.
- baza logaritm = 2 -> cantitatea masurata in numar de biti

Exemplu

- aruncarea monedei – cap sau pajura
- probabilitati egale, 1/2

informatie 1 bit: $H(X) = \sum_{i=1,2} (1/2)*\log (1/(1/2)) = 1$

Incertitudinea

Entropia = cantitatea de incertitudine inlaturata (in medie) de un mesaj

Exemplu:

o sursa poate trimite n = 4 mesaje, cu probabilitati egale p(X) = 1/4

$$H(X) = \Sigma_{i=1,4} (1/4)*\log (4) = 2$$

fiecare mesaj inlatura o incertitudine de 2 biti
(inainte nu se stia care din cele 4 mesaje va fi primit)
Entropia depinde de distributia probabilitatior mesajelor

- H(X) maxim când $p(X_1) = p(X_2) = ... = p(X_n) = 1/n$.

$$H(X) = \Sigma_{i=1,n} (1/n)*\log (n) = \log (n)$$

- H(X) descreşte când distribuţia mesajelor se restrânge.
- 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 şi 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 mulţimea mesajelor $Y_1,..., Y_n$ cu $\Sigma_{i=1,n}\ p(Y_i) = 1$,

- fie: $p_Y(X)$ probabilitatea mesajului X condiţionat de Y.

  p(X,Y) probabilitatea mesajelor X şi Y luate împreună:

  $$p(X,Y) = p_Y(X)*p(Y).$$

  - Echivocitatea este entropia lui X condiţionat de Y:

$$H_Y(X) = \Sigma_{X,Y}\ p(X,Y)*\log(1/p_Y(X))$$

$$H_Y(X) = \Sigma_{X,Y}\ p_Y(X)*p(Y)\ \log(1/p_Y(X))$$

$$= \Sigma_Y\ \mathbf{p(Y)}\ \Sigma_X\ \mathbf{p_Y(X)*\log(1/p_Y(X))}.$$

Exemplu:

- n = 4 şi p(X) = 1/4 pentru fiecare X => H(X) = log 4 = 2.
- Fie m=4 şi p(Y) = 1/4 pentru fiecare Y.
- Presupunem că fiecare Y restrânge X:
- Y1: X1 sau X2, Y2: X3 sau X4, Y3: X2 sau X3, Y4: X4 sau X1.
- Echivocitatea este: $H_Y(X)$ = 4 ((1/4) 2 (1/2) log 2) = log 2 = 1.

Adică: cunoaşterea lui Y reduce incertitudinea lui X la un bit.

Confidenţialitatea perfectă

Cunoasterea unor criptograme nu reduce confidentialitatea

Fie:

- M - multime texte clare cu probabilitatea p(M), Σ_M p(M) = 1.
- C - criptograme, cu probabilitatea p(C), Σ_C p(C) = 1.
- K - chei cu probabilitatea p(K), Σ_K p(K) = 1.
- $p_C(M)$ probabiltiatea să se fi transmis M când se recepţionează C.

Confidenţialitatea perfectă <=> $p_C(M) = p(M)$.

Fie $p_M(C)$ probabilitatea să se recepţioneze C când s-a transmis M:

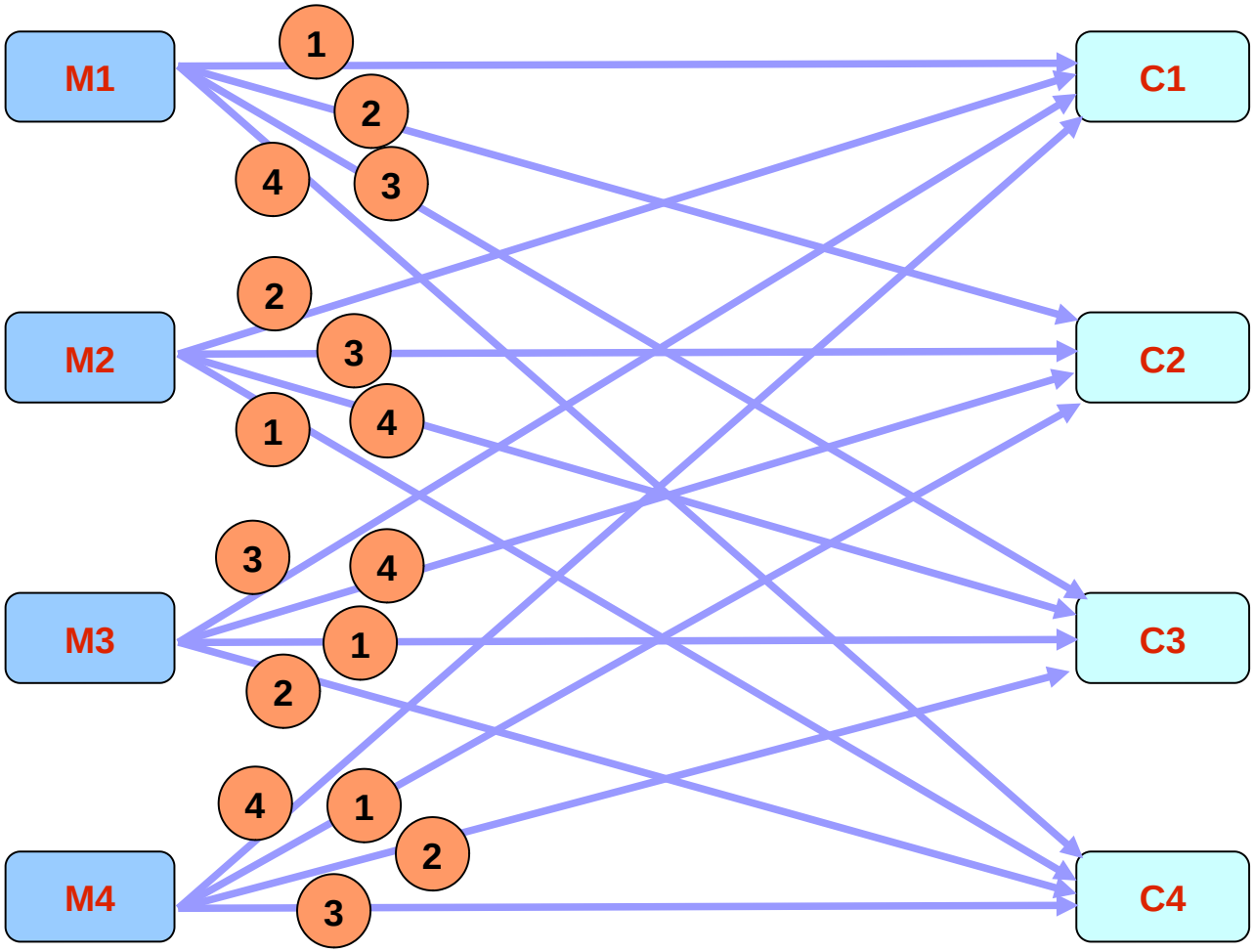
$p_M(C) = \Sigma_{k,\ Ek(M)=C}\ p(k)$.

- Confidenţialitatea perfectă:

$p_M(C) = p(C)$, pentru toate M şi orice C.

Confidenţialitatea perfectă

- Confidenţialitatea perfectă este posibilă dacă se folosesc chei la fel de lungi ca mesajele codificate.



Distanţa de unicitate

- "Spargerea" confidenţialității depinde de cantitatea de criptograme de care intrusul dispune
  - cantitatea de incertitudine în K cunoscand C, este exprimată ca entropia (echivocitatea) cheii conditionata de criptograme:

$$H_C(K) = \Sigma_C\, p(C) \Sigma_K\, p_C(K) \log (1/p_C(K))$$

- Dacă echivocitatea $H_C(K)=0$ nu există incertitudine şi cifrul se poate sparge.
- Când creşte lungimea N a textelor cifrate echivocitatea scade.
- Distanţa 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 mulţimea mesajelor X de lungime N

- cu entropia H(X)
- fiecare mesaj este o secventa 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.
- Redundanţa 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 ... 1.5 biti pe litera

- Numarul total al mesajelor de lungime N cu sens este 2^{rN}

Rata absoluta a limbajului

Rata absoluta 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:

$$R = \Sigma_{i=1,L} (1/L) \log (L) = \log L$$

- pentru limba engleză R = log 26 = 4.7 biţi pe literă

- Numarul de simboluri L se poate rescrie **$L = 2^R$**
- Numarul de mesaje de lungime N (cu sau fara sens) este **2^{RN}**

Calcul aproximativ distanţă unicitate (1)

Distanta 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 aceeaşi probabilitate, $1/2^{rN}$.
  - Toate mesajele fără sens au probabilitate 0.
  - Sunt $2^{H(K)}$ chei cu probabilităţi egale.
  - Cifrul este aleator:
    - Pentru fiecare k şi C, descifrarea $D_K(C)$ este variabilă aleatoare independentă uniform distribuită pe toate mesajele, cu sau fără sens.

Calcul aproximativ distanţă 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 soluţie falsă (adica C se obţine criptând un alt mesaj M' cu înţeles)

cu aceeaşi probabilitate (= mesaje cu sens / total mesaje)

$$q = 2^{rN} / 2^{RN} = 2^{(r-R)N} = 2^{-DN}$$

- Numărul de soluţii false F (= nr chei incorecte * probabilitatea unei chei de a da solutie falsa):

  $F = (2^{H(K)} -1)q = (2^{H(K)} -1)\, 2^{-DN} \approx 2^{H(K)-DN}$

  conditia de unicitate -> $\log F = H(K)-DN = 0$

  $N = H(K) / D$

Analiza cifrării prin substituţie

- Substituţie monoalfabetică:
  - N = H(K) / D = log n! / D sunt n! chei posibile
  - Pentru limba engleză:
    - N = log 26! / 3.2 = 27.6
- Substituţie periodică cu perioada d si s simboluri in alfabet:
  - Sunt s^d chei posibile pentru fiecare substiţutie 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 transpoziţie

- Caracteristici:
  - Cifrul permută 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 şi D = 3.2 rezultă:
  - N = 27.9