RETELE, PROTOCOALE SI SERVICII ASIGURATE PRIN RETELE DE CALCULATOARE

Nicolae Sfetcu
sfetcu@teleactivities.net
www.teleactivities.net

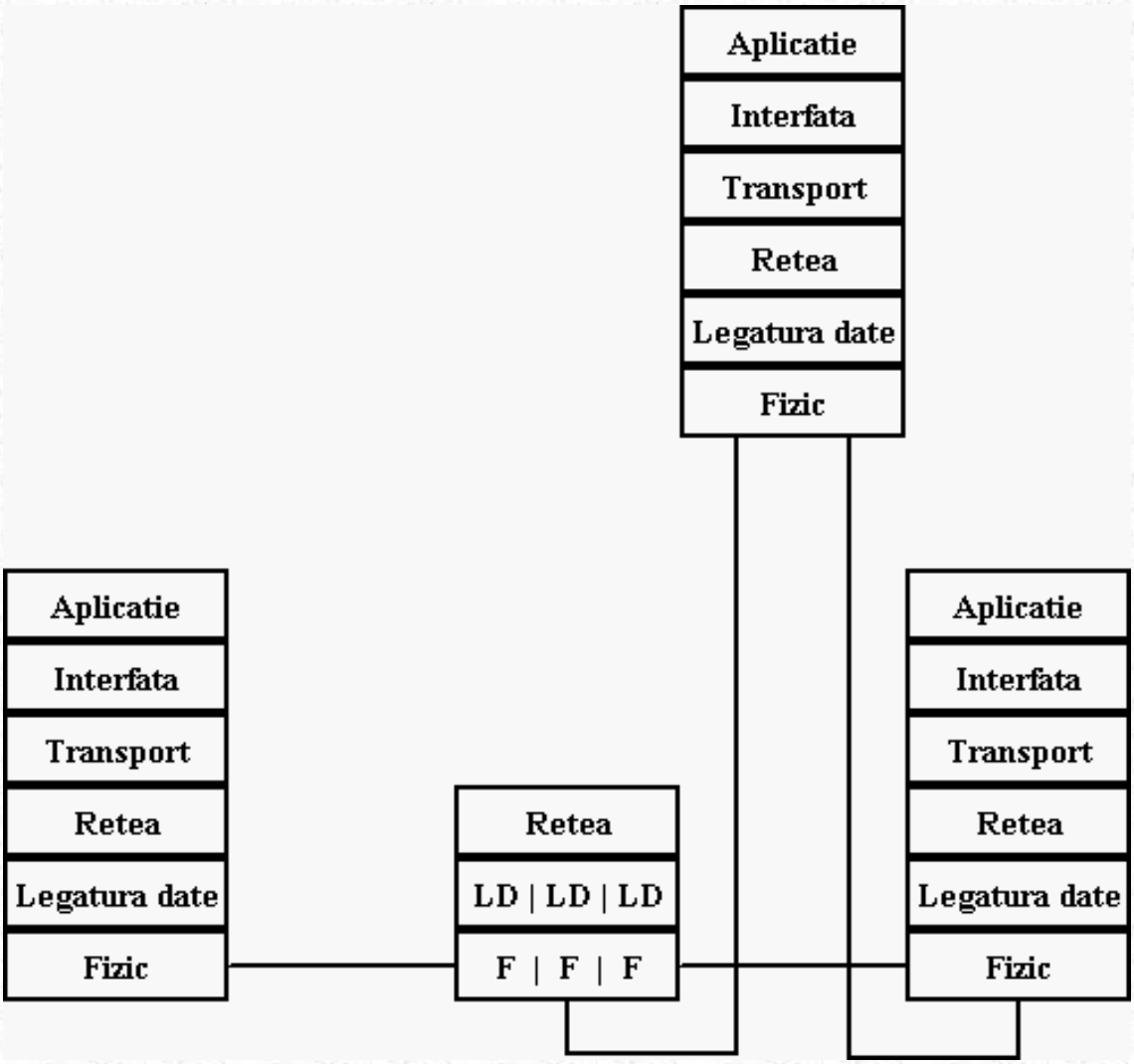
Retele, protocoale si servicii asigurate prin retelele de calculatoare

Ing. Nicolae Sfetcu

E-mail: engineering@teleactivities.net

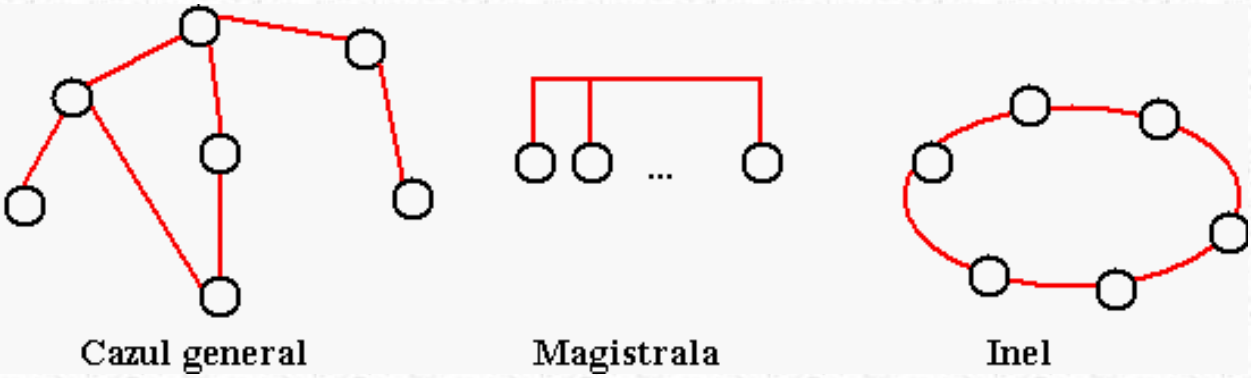
Retele, protocoale si servicii asigurate prin retelele de calculatoare

1. Introducere in retelele de calculatoare



Ce este o retea de calculatoare?

Un set de calculatoare si/sau comutatoare conectate prin linii de comunicatie. Acesta poate avea multe "topologii" posibile:



In functie de intindere, ele pot fi retele locale (local area networks - LAN) sau retele larg raspandite geografic (wide-area networks - WAN). Pot exista o mare diversitate de medii de transmitere a datelor: fibra optica, cablul coaxial, cablul torsadat, radio, prin satelit.

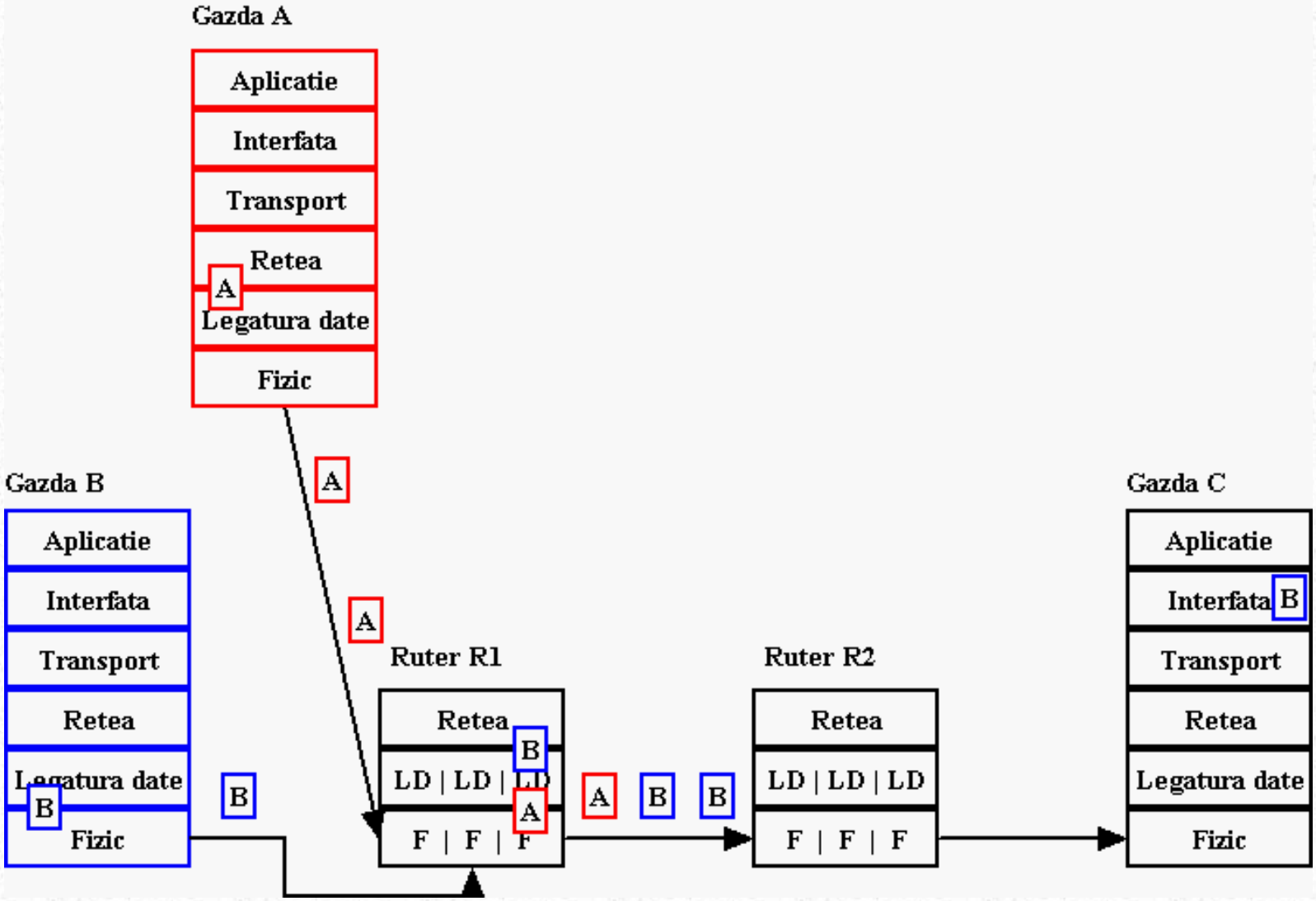
Reteaua de calculatoare mai poate fi definita drept o infrastructura software/hardware:

- *justificarea originala:* permite accesul partajat la resursele de calcul (de ex., calculatoare, fisiere, date)
- este un *mediu* prin care comunica utilizatori dispersati geografic (de ex., email, teleconferinte)

- este un mediu prin care sunt implementate servicii-aplicatii distribuite

Comutarea in pachete

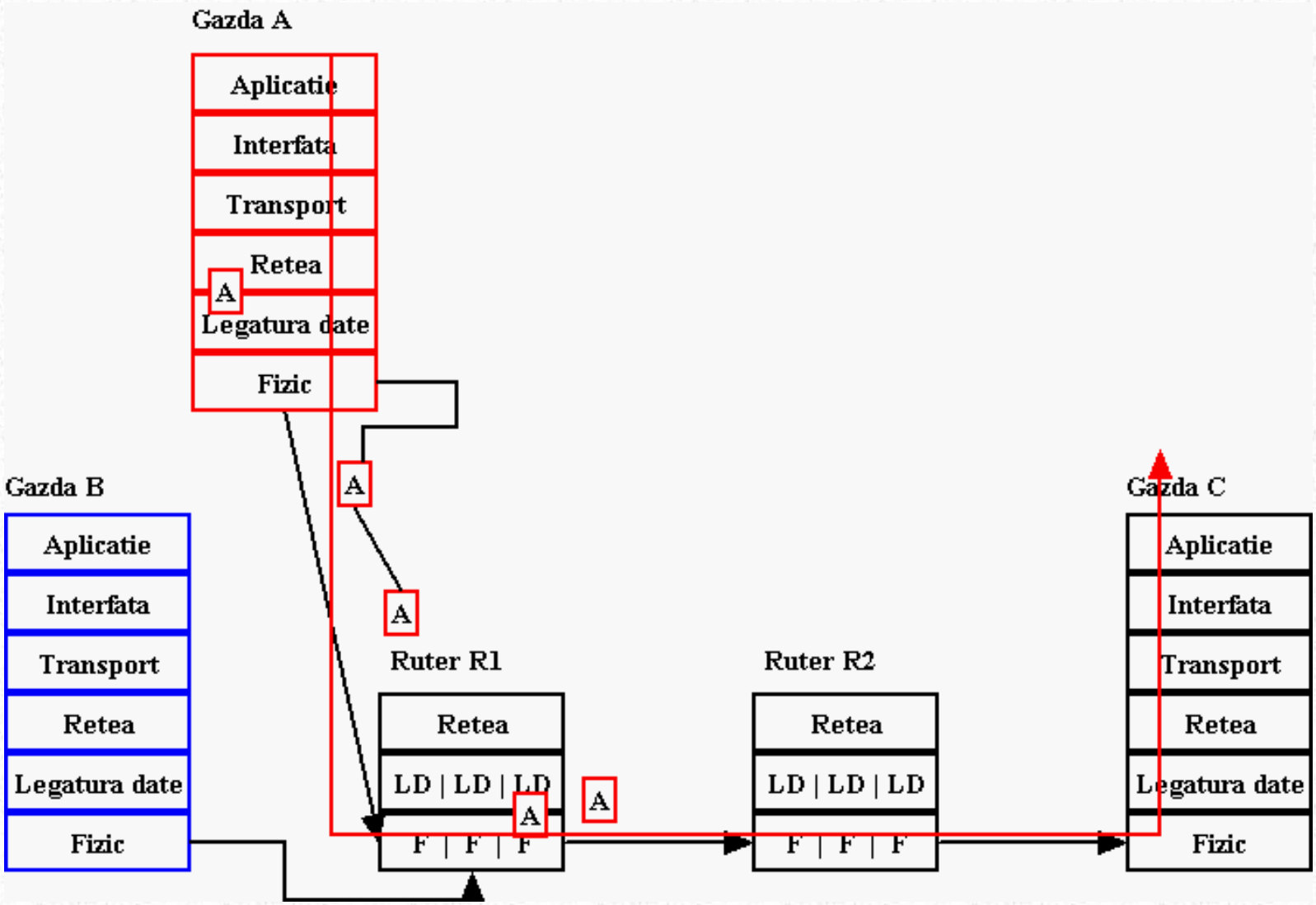
Datele de intrare sunt divizate in fragmente denumite "pachete". Pachetele care traverseaza reteaua pun in comun resursele retelei (de ex., largimea de banda a legaturii, buferele). In functie de cerinte, se foloseste si partajarea statistica a resurselor.



Nevoia de resurse poate depasi posibiltatile (de ex., pachetele A si B sosesc la R1, cu destinatia C).

Retele cu circuite comutate

Acestea folosesc exclusiv toate resursele (de ex., liniile de comunicatii) solicitate de apel pe toata durata acestuia (de ex., reteaua telefonica).



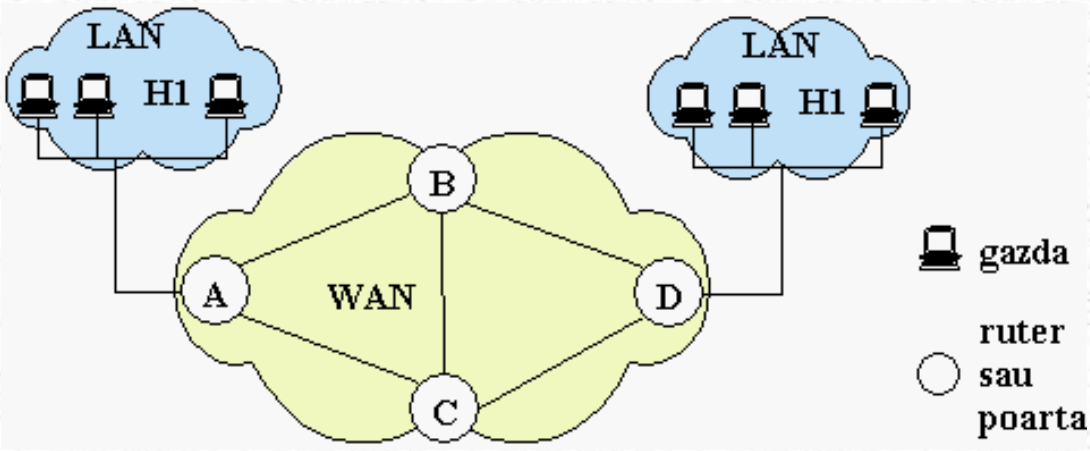
Este posibil ca nevoia resurse sa depaseasca posibilitatile existente (A si B vor sa apeleze pe C). In acest caz, se ajunge la situatia de *blocare* (semnal de ocupat)

De ce resurse partajate statistic?

Pentru ca astfel se economiseasc bani! De exemplu: o legatura de 1 Mbit/sec; fiecare utilizator primeste 100 Kbits/sec cand transmite; fiecare utilizator poate transmite date numai in 10% din timpul total. In timp ce *comutarea de circuite* permite fiecarui apelant numai o capacitate de 100 Kbits/sec rezultand posibilitatea suportarii a 10 apelanti, pentru *comutarea de pachete* cu 35 posibile apeluri probabilitatea ca 10 sau mai multi apelanti sa fie activi simultan este < 0.0004! Rezulta ca aceasta din urma poate suporta mult mai multi apelanti, cu o probabilitate mica de intrare in "conflict". Daca insa utilizatorii obisnuiesc sa apeleze frecvent reteaua, atunci este mai avantajoasa comutarea in pachete (Baran, 1965)

Elementele unei retele

- *linii de comunicatii:*
  - punct-la-punct (point-to-point), de ex., A-catre-B)
  - cu difuzare (de ex.,: LAN Ethernet)



- *gazda:* calculator care ruleaza aplicatii folosind reteaua (de ex.: H1)
- *ruter:* calculator (adesea cu programe la nivel de aplicatii) care dirijeaza pachetele de la intrare catre iesire (de ex., C)
- *poarta:* un ruter conectat direct la doua sau mai multe retele (de ex. A)
- *retea:* set de noduri (gazde/rutere/porti) din cadrul unui *unic domeniu administrativ*
- *internet:* colectie de retele interconectate

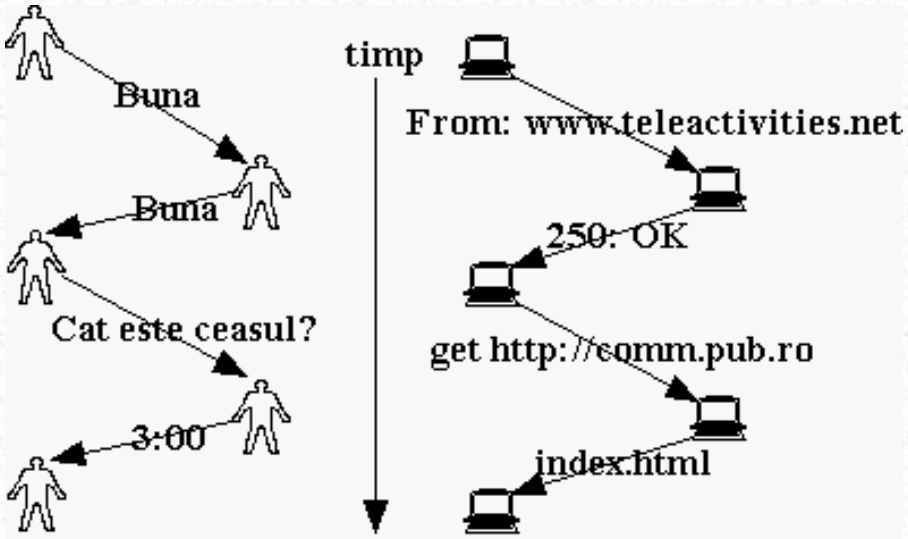
Protocoale

Protocol: reguli prin care elemente de retea active (aplicatii, gazde, rutere) comunica intre ele

Protocoalele definesc:

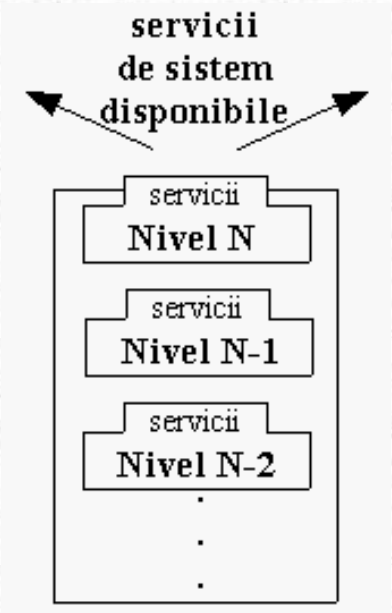
- formatul/ordinea mesajelor schimbate
- actiunile intreprinse la receptionarea mesajului.

Ele sunt asemanatoare regulilor prin care doi sau mai multi oameni comunica intre ei.



Arhitectura pe nivele

Arhitectura sistemelor complexe se poate simplifica prin stratificarea acesteia. Nivelul N se bazeaza pe serviciile nivelului N-1 pentru a oferi un serviciu nivelului N+1.

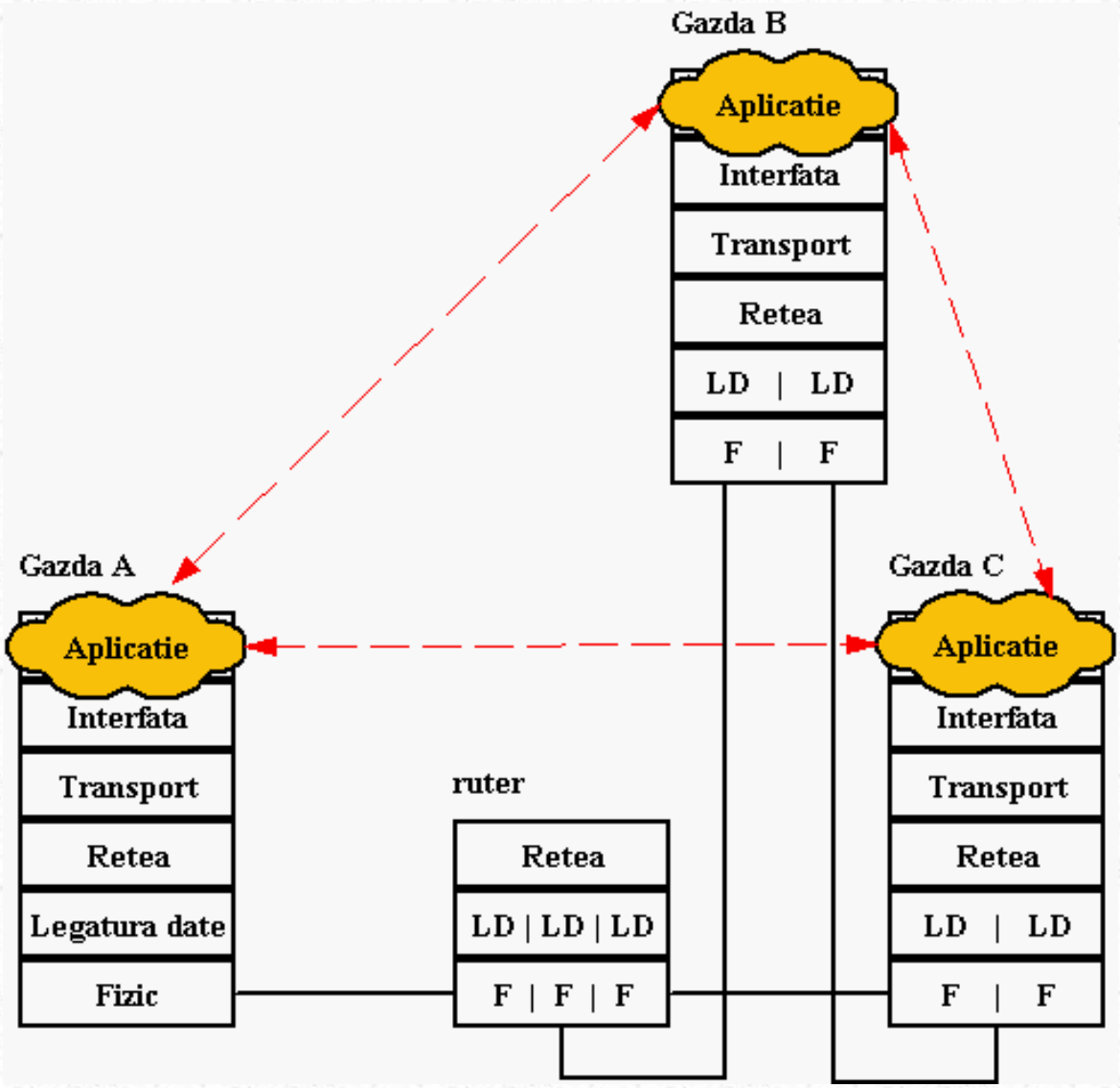


Serviciul provenit de la un nivel inferior este independent de modul cum acest serviciu este implementat. Schimbarile nivelului N nu afecteaza celelalte nivele.

Interfetele definesc modul in care sunt solicitate serviciile.

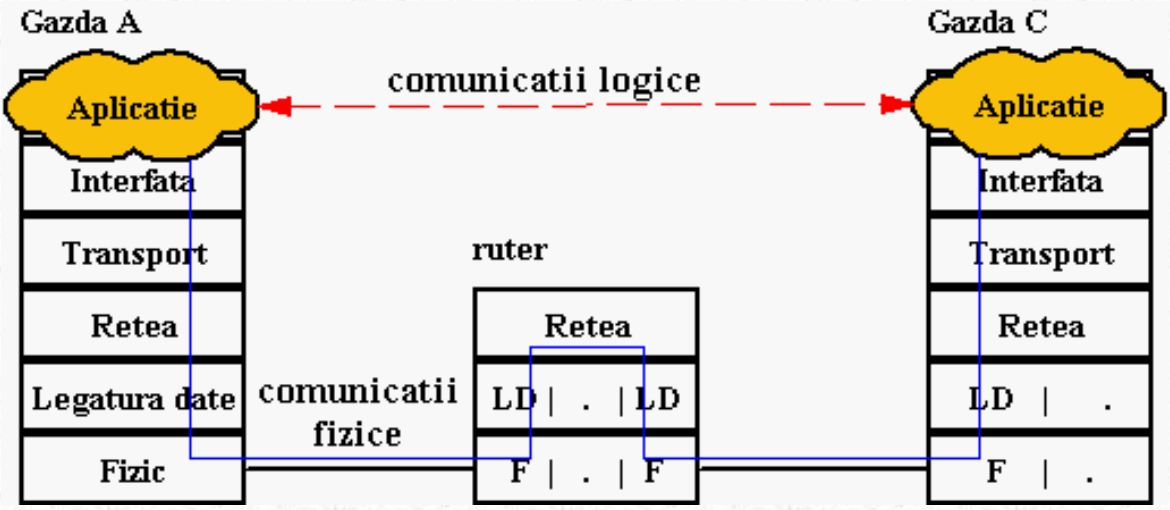
Arhitectura retelei pe nivele

Reteaua consta din componente hardware/software cu o distributie spatiala. In figura de mai jos se prezinta o imagine a unei ***retele cu distributie pe nivele***:

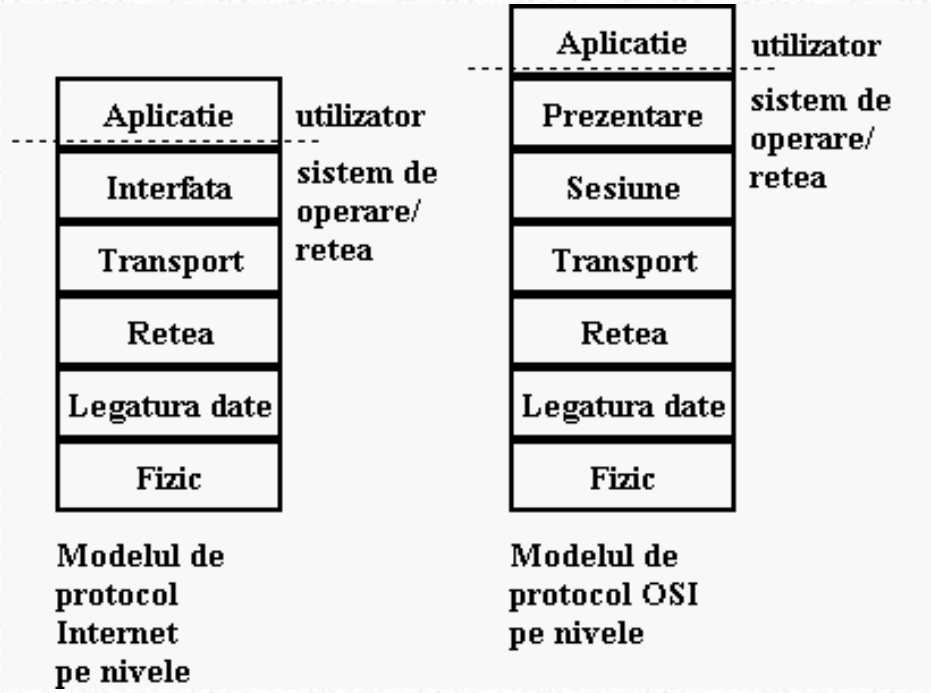


Stratificare si protocoale

Entitatile similare (de ex., procesele) din nivelul N isi ofera servicii prin comunicare (trimitand "pachete"), folosind serviciile de comunicare oferite de nivelul N-1.

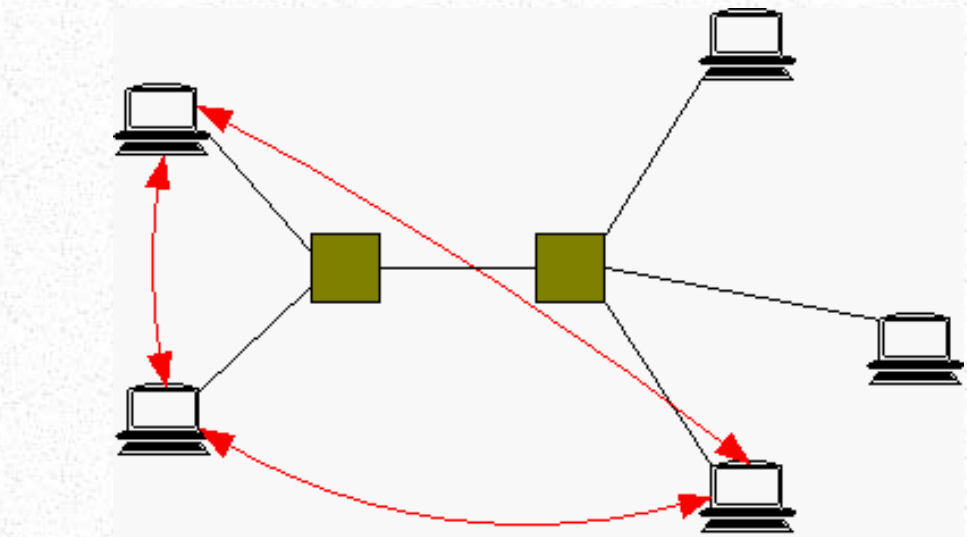


Internetul si modelele de referinta ISO/OSI



Nivelele unei arhitecturi de protocoale

Nivelul Aplicatie



- comunicare de la proces la proces
- exemple: WWW, email, teleconferinta, documentarea

Nivelul internet (socket - numai pentru Internet)

- rol de bufer si de livrare a datelor la sistemele terminale

Nivelul prezentare (numai pentru OSI)

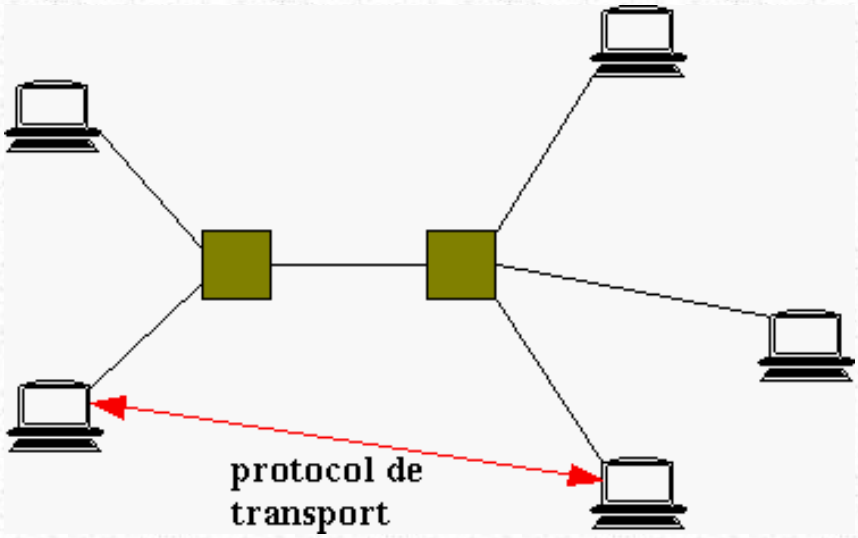
- conversia datelor intr-un format comun
- stiva Internet: conversia datelor la nivel de utilizator.

Nivelul sesiune (numai pentru OSI)

- reface sesiunea (de ex., autentificarea), prin recuperare dintr-o sesiune esuata (intrerupta)
- este un nivel "subtire".

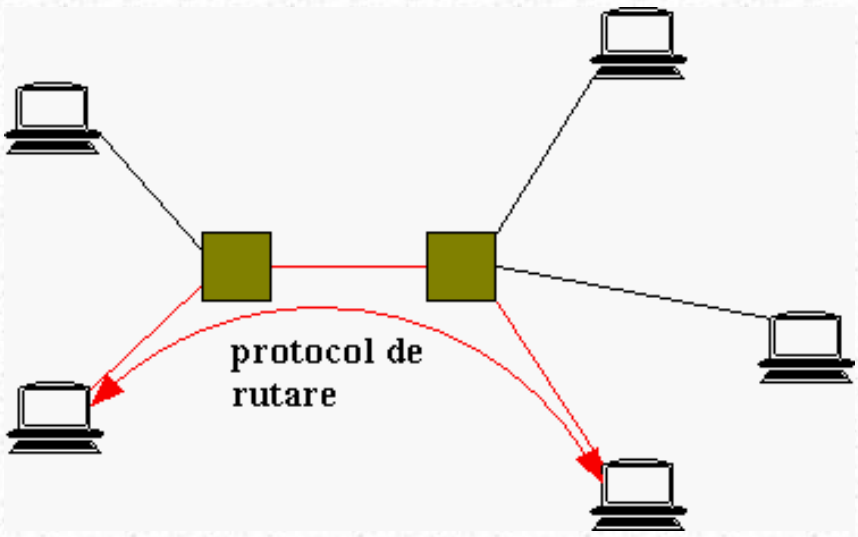
Nivelul transport

- serviciul de transport: livrarea datelor de la un terminal la altul
- poate multiplexa mai multe fluxuri provenind de la nivele superioare
- egaleaza viteza intre transmitator si receptor
- pentru Internet: TCP si UDP



Nivelul retea

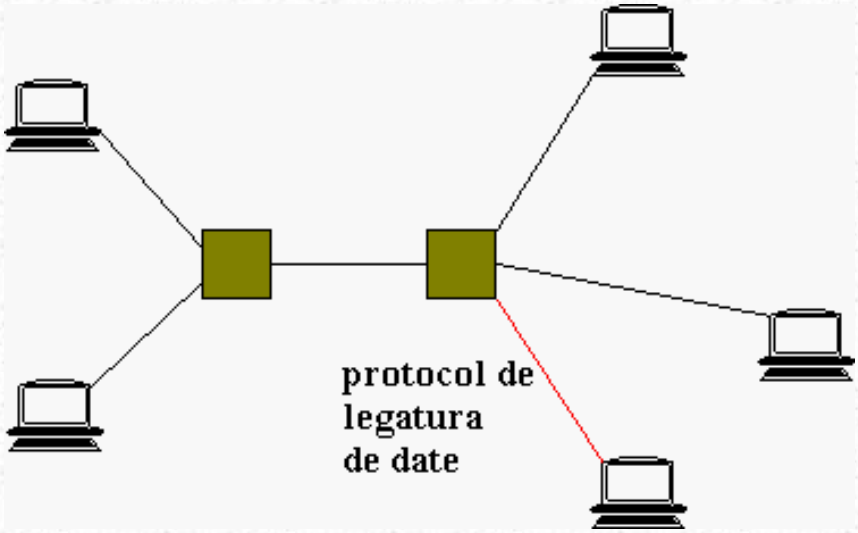
- la gazdele-terminale: initiaza transmiterea pachetelor pe traseul acestora
- la rutere: rutarea pachetelor de control
- evita blocarea si controleaza eventualele congestii
- pentru Internet: pachete IP, BGP, RIP



Nivelul legaturii de date

- asigura o comunicare punct-la-punct fara erori in cazul unei singure legaturi
- protocoale LAN de multiacces
- egaleaza viteza intre transmitator si receptor

- Ethernet, HDLC, PPP



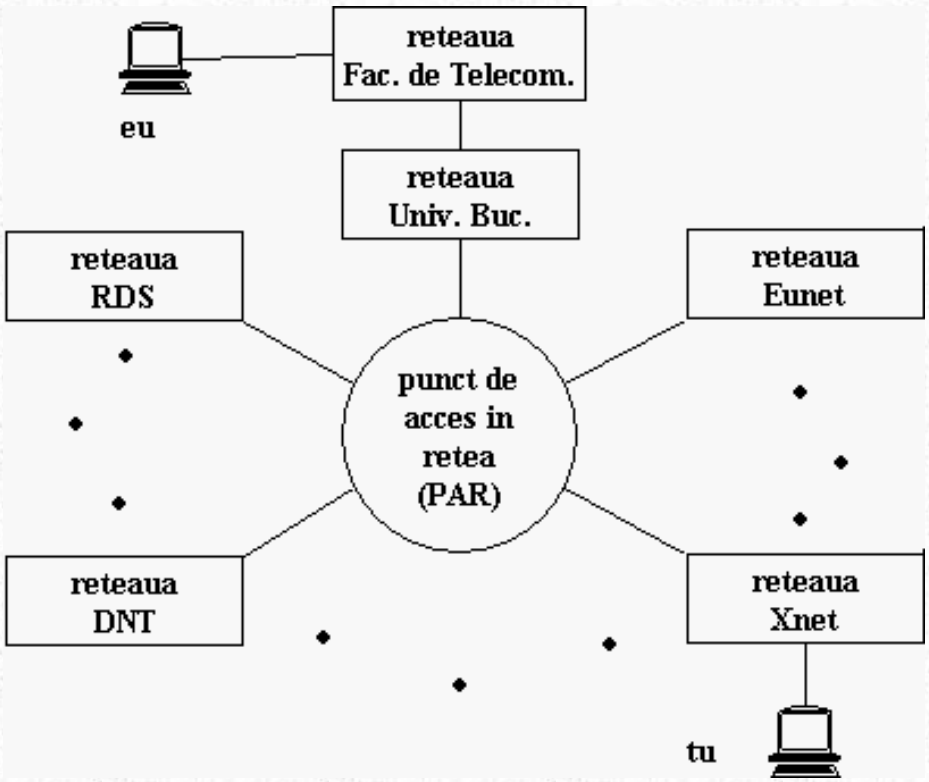
Nivelul fizic:

- transmite bitii elementari (0/1) prin linia de comunicatii

Inter-retele: Internet

Un internet reprezinta o interconectare a mai multor retele (o retea de retele). Fiecare retea este *administrata separat.*

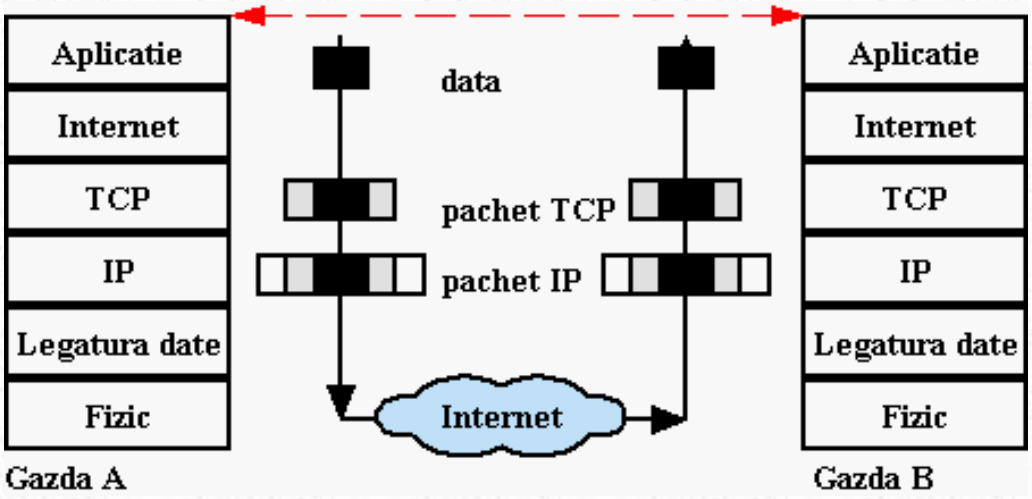
Internet (cu majuscula) este reteaua de retele in care fiecare retea ruleaza acelasi software: protocoalele Internet.



Pachete de protocoale

Pachetul este unitatea de date schimbata intre entitatile protocoale intr-un nivel dat.

Datele de la un nivel sunt **incapsulate** in pachet la nivelul inferior (in genul "*camerei in anvelopa*").



Solutii generice in cadrul unui nivel

- *controlul erorii:* face "canalul" mai sigur
- *controlul fluxului:* evita floodarea punctelor mai lente
- *fragmentarea:* imparte cantitatile mari de date in bucati mai mici, care sunt apoi reasamblate
- *multiplexarea:* mai multe sesiuni ale nivelelor superioare folosesc in comun o singura conexiune a nivelului inferior
- *setarea conexiunii:* optimizarea elementelor
- *adresare/numire:* localizarea si administrarea identificatorilor asociati cu entitati.

Dificultati ale stratificarii

Stratificarea are avantaje conceptuale si structurale. Este posibil ca nivelul N sa copieze functionalitatea nivelului inferior rezultind, de ex., eliminarea erorilor de salt.

Diferite nivele s-ar putea sa aiba nevoie de aceleati informatii (de ex., frecventa de tact).

Nivelul N poate avea nevoie de informatii de la nivelul N-2 (de ex., dimensiunile pachetului nivelului inferior).

O scurta istorie a retelelor

- **1830:** telegraful
- **1876:** telefonul (comutare de circuite)
- **anii 1960:** comutarea de pachete (Baran, Davies)
  - Arpanet ajunge la 4 noduri
- **anii 1970:**
  - companii: DECnet, IBM SNA
  - Arpanet ajunge la 100 noduri
- **anii 1980:**
  - retele locale (local area networks)
  - sfarsitul anilor 80: 100 Mbps
  - proliferarea retelelor larg raspandite geografic (wide area networks): CSNET, MILNET, NSFNET, ARPANET
  - Internet depaseste 100.000 noduri in 1989

- **anii 1990**
  - Arpanet, NSF desfiintat: guvernul american nu mai ofera servicii de backbone
  - crestere exploziva: 10 milioane de gazde in 1996
  - 150Mbps, 660 Mbps
  - reselele fara fir
  - WWW conduce in mania Internet
- **tendinte curente:**
  - expansiunea va continua
  - se vor dezvolta serviciile comerciale
  - se va acorda o mai mare atentie securitatii

Top Next

E-mail: engineering@teleactivities.net

2. Nivelul Aplicatie

Aplicatii de retea

Aplicatiile pentru cerintele serviciilor se plaseaza in infrastructura retelei.

Aplicatiile distribuite ale protocoalelor se folosesc pentru a implementa aplicatia.

O lista partiala de aplicatii:

posta electronica
grupuri de stiri
transfer de fisiere
acces la distanta: telnet
teleservicii: audio/video/text
teleconferinta
telestiinta
studiu la distanta
telelucru
CSCW
video la cerere
cumparaturi de la domiciliu
servicii de chat
fax
procesarea tranzactiilor
banca/depozit electronic
votul electronic
baze de date distribuite
WWW
biblioteci digitale
controlul traficului aerian
monitorizarea traficului
telemetrie
retransmisie la distanta
telemedicina
alte aplicatii

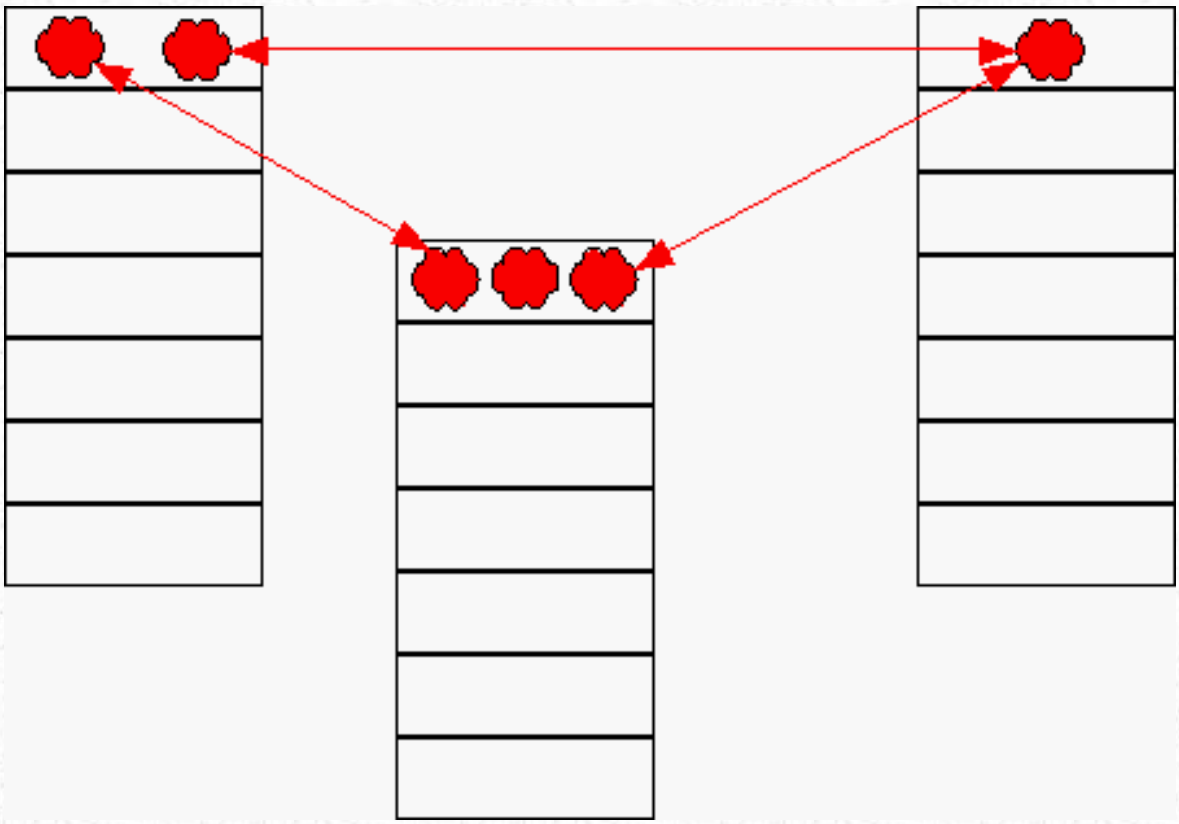
Cerintele Aplicatiei

- ***largimea de banda:*** cati biti/sec sunt necesari? Traficul este uniform, sau neregulat?
- ***procesarea protocolului:*** citi MIPS (milioane de instructiuni per secunda) sunt necesari pentru software necesar aplicatiei si protocoalele corespunzatoare?
- ***timpul de mentinere:*** cat timp ruleaza aplicatia?
- ***siguranta nivelului de date:*** este necesara livrarea in siguranta (in ordine, fara pierderi)?
- ***performanta:*** sunt necesare constrangeri asupra intarzierilor maxime de la aplicatie la aplicatie, crearea de cozi pentru distributia intarzierilor?
- ***calitatea serviciilor:*** sunt necesare garantii privind calitatea serviciilor?
- ***structura comunicatiilor*****:** 1-1, 1-supraunitar, supraunitar - supraunitar?
- ***securitatea:*** sunt necesare autentificarea si incriptarea?

aplicatia	largimea de banda	siguranta	calitatea serviciilor	mentinere	securitate
pachet audio	10K - 1M	n	d/n	min/h	d/n
pachet video	10K - 10M	n	d/n	min/h	d/n
video-la-cerere	1M - 10M	n/d	d	min/h	n
email	10K	d	n	min	d/n
tranzactii	1K	d	d/n	sec	d/n
raze X	> 10M	d/n	d/n	min	d/n
fax	10K	d	n	min	d/n

Structura Aplicatiei

Aplicatii de retea distribuite in natura. Set de procese la nivel aplicatie Comunicatii (de obicei) pe diferite gazde, care ofera / implementeaza serviciul.



Modelul client/server:

Acest model este asimetric: serverul ofera servicii via o interfata bine definita, clientul primeste serviciul. Problema clientului este localizarea / receptia serviciului. Problema serverului este cum / daca sa ofere un anumit serviciu.

Exemplu: client (browser) WWW, server

Modelul "peer/peer"

Este simetric: fiecare proces are un egal.

Exemplu: teleconferinta.

Amindoua modele necesita transportul solicitarilor / raspunsurilor, si partajarea datelor.

Exemplu de aplicatie: email

Probleme generice:

- adresarea: identitatile receptorului / transmitatorului, informatii despre formatul adresei
- confidentialitatea/securitatea
- notificarea receptiei, citirea, dispunerea
- o diversitate de tipuri de media: text, audio, video, documente tiparite

Trebuie rezolvata problema standardelor email care nu sunt interoperationale: portile de mail trebuie sa poata face conversii intre diferite tipuri de formate.

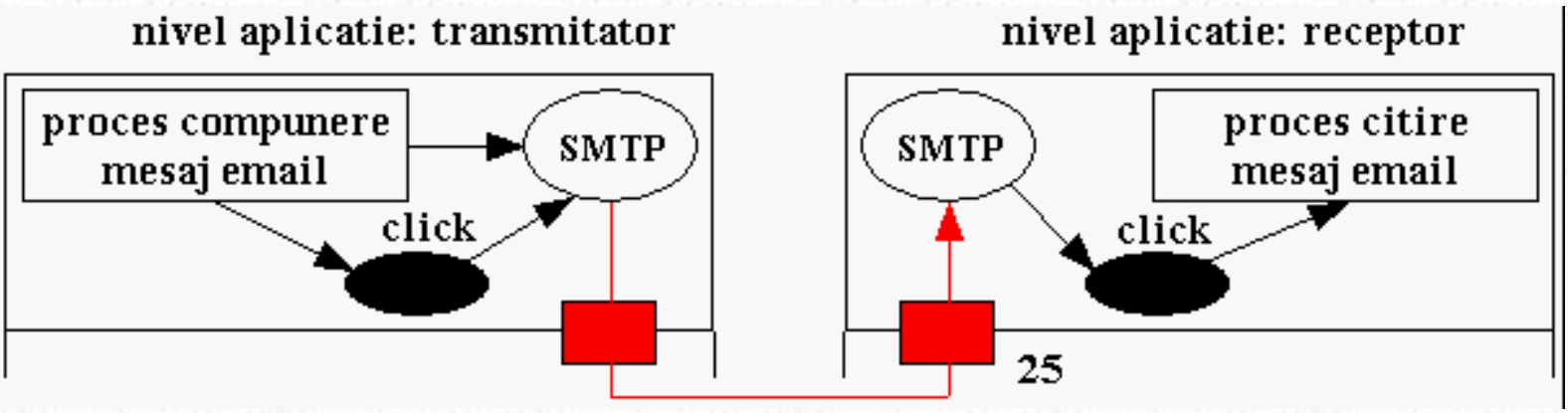
Posta Internet: SMTP

SMTP este protocolul de transfer postal simplu. RFC 821 si 822 definesc activitatile de protocol si structura mesajului pentru SMTP. RFC ("request for comments" - solicitare de comentarii) sunt proiecte, standarde si documente informative pentru Internet.

Locatii: ftp://teleactivities.org, http://telelucru.hzpermart.net/ro/ghid/legi/plata.htm

Transmiterea mesajelor:

- practic este vorba de un transfer de fisiere mediat de SMTP: entitatea protocol SMTP al expeditorului transmite catre entitatea protocol SMTP al destinatarului.
  - expeditorul afla adresa de gazda a destinatarului
  - expeditorul contacteaza destinatarul la "portul" bine cunoscut cu numarul (25) pentru acea gazda
- se foloseste serviciul de transport TCP.



Comenzile SMTP ale expeditorului - destinatarului

Expeditorul si destinatarul SMTP schimba interactiuni tipice peer-peer:

- comenzi de control, raspunsuri, date
- trei trepte de baza: "greeting", schimbul de date, "goodbye"

Comenzi selectate ale expeditorului

comanda	argumentul	semnificatia
HELO	domeniul expeditorului	"sunt com.pub.ro"
MAIL FROM:	userid	identifica expeditorul mesajului
RCPT TO:	userid	indentifica recipientul
DATA		urmeaza textul mesajului
<crlf>.<crlf>		sfarsitul textului mesajului
RESET		esuarea protocolului, renuntarea
VERIFY	userid	este valid userid - ul?
QUIT		semnarea de parasire a transmisiunii a expeditorului

Raspunsuri selectate ale destinatarului

numarul raspunsului	semnificatia
500	eorare de sintaxa, ultima comanda nerecunoscuta
220	serviciu gata (gata pentru a primi mesaj)
221	OK. Inchid si eu conexiunea
250	OK. Comanda executata
354	incepe sa imi trimiti textul mesajului
552	renunt la mesaj, depaseste spatiul alocat
550	actiune neindeplinita, casuta postala indisponibila

Exemplu de schimb de SMTP

expeditor proces SMTP destinatar proces SMTP

220

<----------

HELO com.pub.ro

<----------

250

Mail From: sfetcu@com.pub.ro

---------->

250

<----------

RCPT TO: office@remat.ro

---------->

250

<----------

RCPT TO: nic@remat.ro

---------->

550 no such user

<----------

DATA

---------->

354 start mail input

<----------

Marian Ionescu

---------->

Thank you for your email

---------->

bla bla bla

---------->

<crlf>,<crlf>

---------->

250

<----------

quit

---------->

221

<----------

SMTP: Comentarii finale

Extensii ale SMTP:

- MIME (multipurpose Internet mail extensions): componente multiple ale corpului mesajului, posta multimedia, fonturi multiple si seturi de catactere, RFC 1324
- PEM (privacy-enhanced mail): RFC 1422-1424

Post Office Protocol (POP3)

- SMTP presupune ca exista un server SMTP la destinatar
- SMTP poate de asemenea sa faca livrari la un Oficiu Postal (server)
- clientul poate prelua mesajul de la distanta folosind Post Office Protocol (POP3) pentru a interactiona cu serverul. Sunt trei faze in acest caz:
  - salutul
  - tranzactiile (listare, preluarea mesajului)
  - inchiderea transmisiei

Hypertext Transfer Protocol: http

WWW este implementat folosind paradigma client/server:

- ***clientul*** (browserele) solicita si afiseaza documentul html primit
- ***serverele*** primesc cererea, raspunzand cu documentul html solicitat

Protocolul http defineste formatul pachetelor schimbate intre client si server, actiuni derulate cu confirmare

http este un protocol orientat pe tranzactii:

- clientul contacteaza serverul pe portul 80, folosind TCP
- solicitare de la client catre server
- raspuns de la server catre client
- conexiunea TCP se inchide

Solicitarea / raspunsul au headerele si corpurile similare cu SMTP si MIME

RFC 2068 defineste http

HTTP Mesajul de solicitare

Se transmite de la client catre server.

Formatul general:

- linia solicitarii (metoda, identificator, versiune)
- header (informatii suplimentare)
- corp

Solicitari http

partea mesajului	numele campului	descriere
header	method	actiunea care va fi derulata (GET, PUT, DELETE)
	version	ce versiune http
	identifier	URL (adresa) obiectului
header	From	adresa mea de email
	Accept-Language	limba pe care o voi accepta
	If-Modified-Since	returneaza documentul daca este mai nou
	Content-Type	tipul continutului
	....	
body	message body	

Exemplu Mesaj solicitare http

Sa presupunem ca dvs. accesati http://www.teleactivities.net/ebusiness/guides/webdesign/design.htm:

Mostra de comanda GET command plus header:

GET /webdesign/design.htm HTTP/1.0

User-Agent: Mozilla/2.01 (X11; I; IRIX 5.2 IP7)

Accept: image/gif, image/x-bitmap, image/jpeg

/* a blank line */

http Mesaj de raspuns

Transmis de la server catre client

Formatul general:

- linie de stare (cod de stare, fraza textului)
- header (informatii suplimentare)
- corp

Coduri html de raspuns selectate

cod	semnificatie
200	OK
400	solicitare gresita
402	plata solicitata
404	nu a fost gasit
503	serviciu indisponibil
505	versiune HTTP nesuportata

Exemplu Mesaj de raspuns http

O mostra de raspuns http al serverului:

HTTP/1.0

200 Document follows

MIME-Version: 1.0

Server: Hzpermart

Date: Wednesday 10-Apr-01 03:59:47 GMT

Content-type: text/html

Content-length: 2168

Last-Modified: Friday 06-Oct-00 07:16:52 GMT

/* a blank line */

<html>

<head>

: /* HTML text of the Web page */

</html>

Aplicatii de Teleconferinta: cerinte

Scop: comunicare interactiva (in timp real) cu componente media multiple (audio, video, text).

Media continuu, precum audio, video

- receptionarea fara discontinuitati si neteda a transmisiei
  - Internetul nu ofera comunicatii lipsite de fluctuatii
    - fluctuatia: modificare a intarzierilor din retea
  - destinatarul bufereaza pachetele pe care le ruleaza periodic
  - rularea intarziata a primului pachet permite buferului de la destinatar sa "absoarba" fluctuatiile retelei
  - pachetele intarziate se pierd
- se poate tolera "unele" pachete pierdute

Back | Top | Next

Retele, protocoale si servicii asigurate prin retelele de calculatoare

3. Programarea aplicatiei de retea

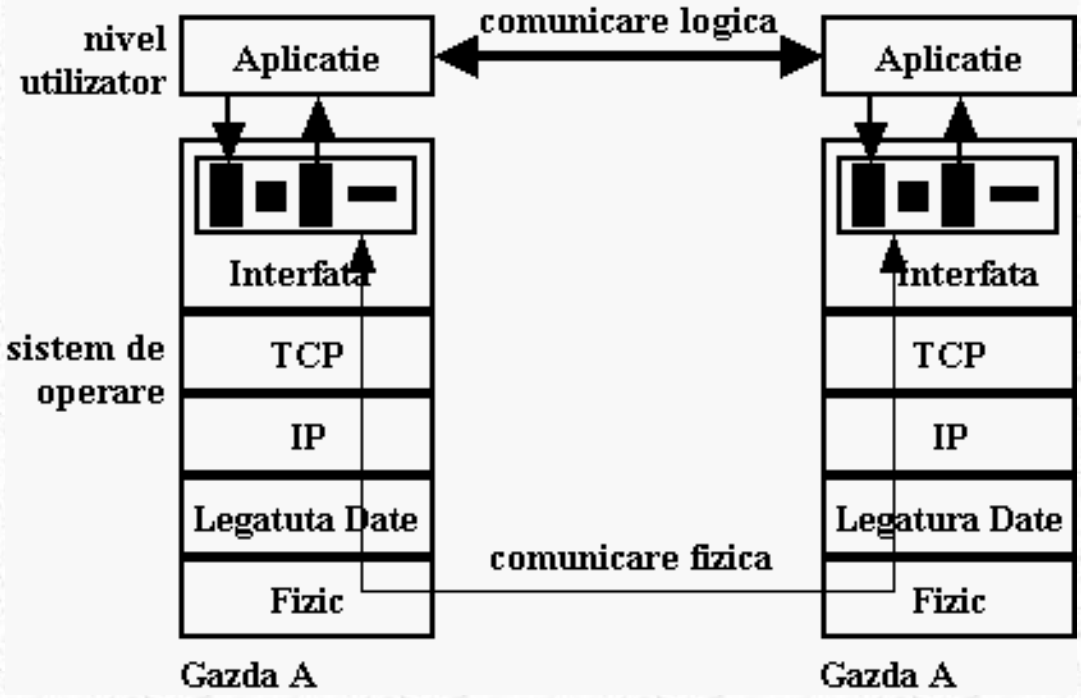
Interfata de Programare a Aplicatiei - IPA

IPA pentru SOCKET

Introdusa in 1981, BSD 4.1 UNIX

O ***interfata locala, creata/apartinand de aplicatie, controlata de SO*** in care procesul

aplicatiei poate atat ***transmite cat si receptiona mesaje*** catre/de la alt proces al aplicatiei (la distanta sau local)



Doi socketi pe gazde separate "conectati" prin rutine de management al lacasului pentru SO. Aplicatia vede numai socketul local.

Socketii sunt creati explicit, utilizati si puti in functiune de catre aplicatii. Ei se bazeaza pe paradigma client/server. Exista doua tipuri de servicii de transport via IPA pentru socket:

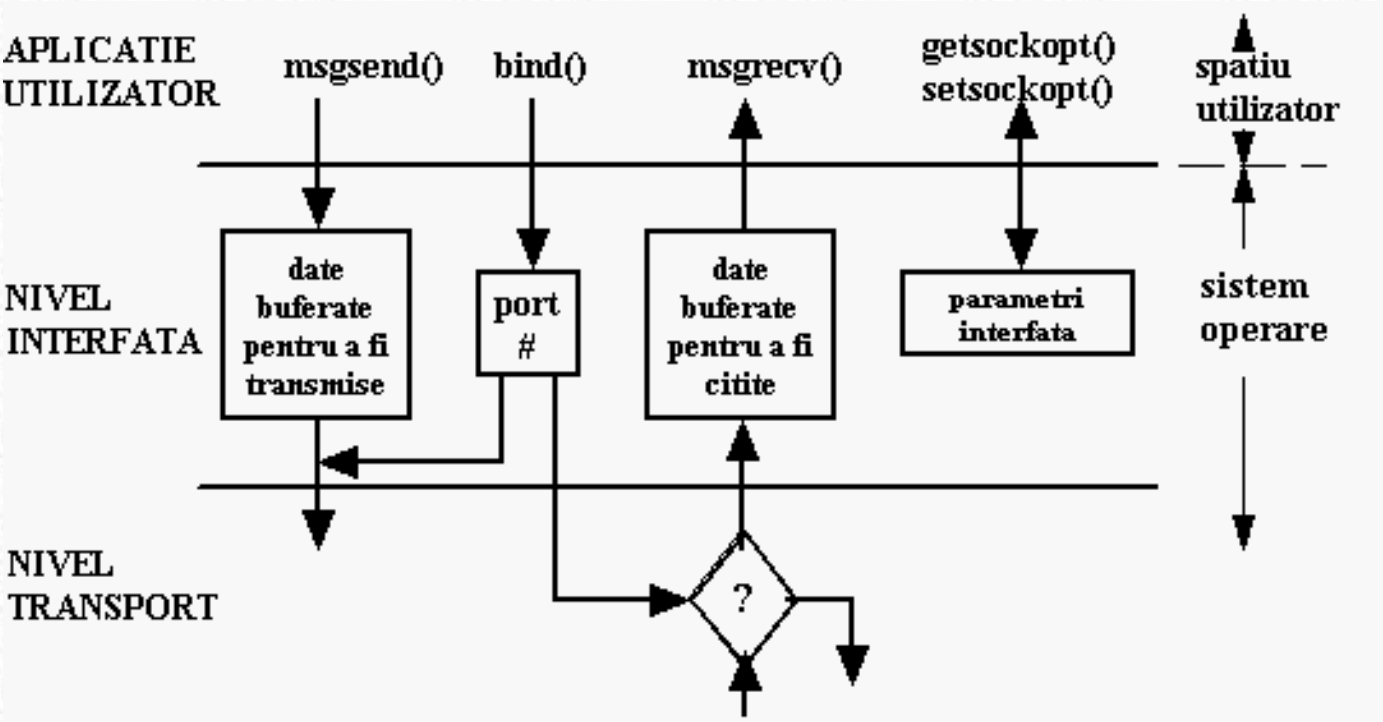
- datagrame nesigure
- sigure, orientate pe flux

Nivelele de prezentare si sesiune lipsesc in retelele UNIX.

Socketi: aspectul conceptual

Fiecare socket are bufere de transmisie si receptie separate, posibilitatea de identificare a portului, parametri (aplicatie interogativa si setabila).

Operatiile socketului implementat ca sistem apeleaza la SO. Limitele utilizator/kernel se incuciseaza pe sus.



Servicii fara conexiune

Serviciu datagrama: protocoalele de transport de baza ***nu garanteaza livrarea.***

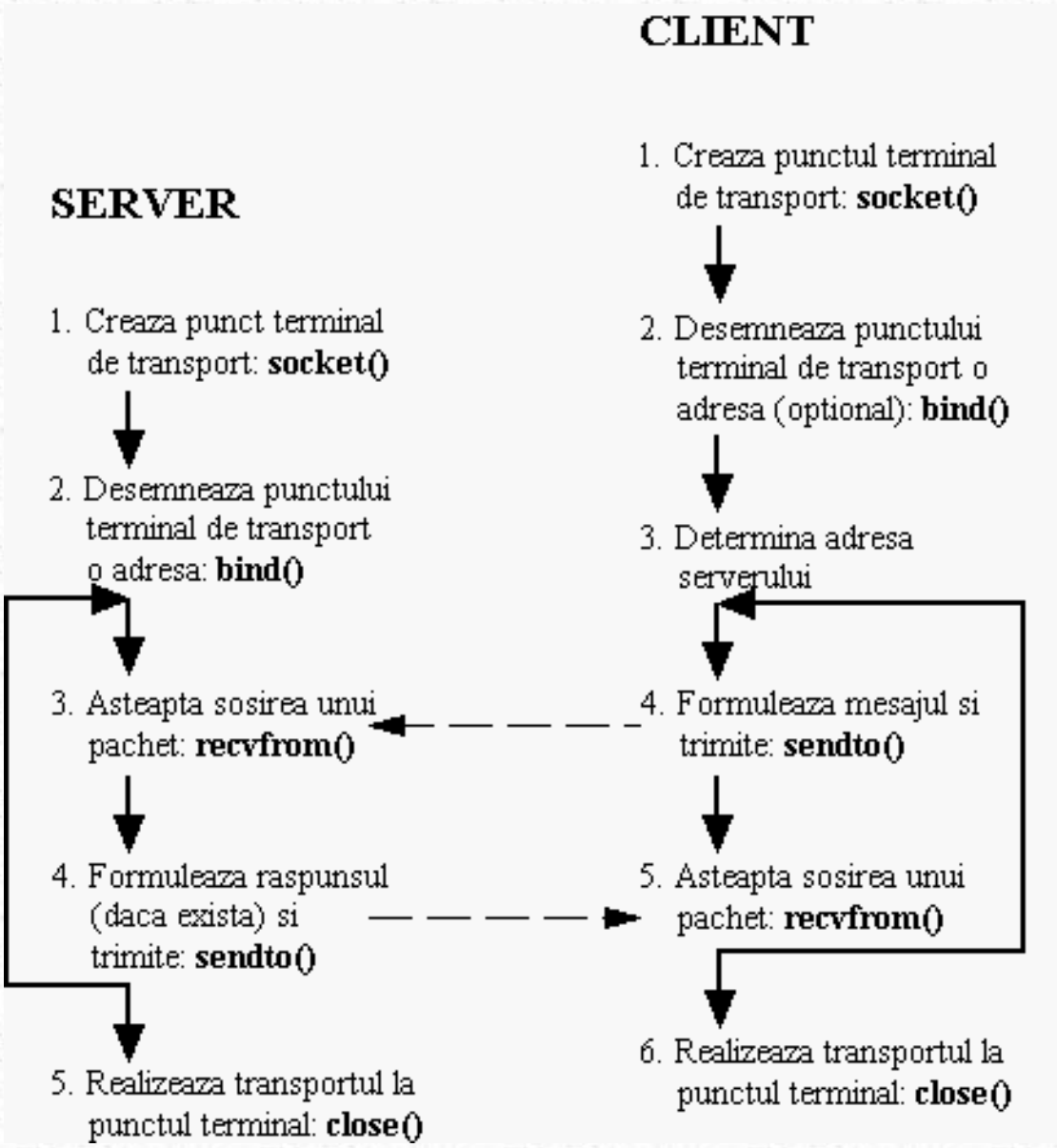
Nu exista o identificare explicita a serverului sau clientului.

Daca se initiaza contactul cu cealalta parte, trebuie cunoscute.

- adresa IP
- numarul portului sau procesul care asteapta sa fie contactat

Daca se asteapta contactul de la cealalta parte, trebuie declarat:

- numarul portului la care se asteapta de cealalta parte.



Crearea unui socket

Acelasi terminal (socket) folosit pentru a transmite/receptiona date.

Nu exista o asociere a priori a socketului cu reteaua.

Trebuie sa se specifice familia de protocoale de transport, si serviciul specific la nivel de transport care se va folosi cu socketul:

Familia de protocoale	Nume simbolic
TCP/IP Internet	AF_INET
Xerox NS	AF_NS
intra-host UNIX	AF_UNIX
DEC DNA	AF_DECNET

Tipul serviciului	Nume simbolic	Comentariu
datagrama	SOCK_DGRAM	protocol UDP in AF_INET

sigura, in ordine	SOCK_STREAM	protocol TCP in AF_INET
socket brut	SOCK_RAW	acces direct la nivelul retea

Socket int (familia int, serviciul int, protocol int)

Familia este numele simbolic al familiei de protocoale.

Serviciul este numele simbolic al tipului de serviciu.

`Protocolul` permite o specificare mai buna a socketului brut. Pentru noi, acesta va fi 0.

Codul de raspuns de la `socket() este` este un descriptor de socket, folosit in toate apelurile de sistem legate de socket.

Exemplu:

```
#include <sys/types.h>
#include<sys/ socket.h>

 int sockfd;

 if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
    { /* handle error */ }
```

Adresarea Internetului

Fiecare gazda Internet are una sau mai multe adrese IP pe 32 bit unite la nivel global.

Gazda poate avea doua sau mai multe adrese:

- adresa este asociata cu fiecare placa de interfata.

Notatia decimala cu punct:

- numere intregi de 4 cifre, fiecare definind un bit din adresa IP:

Nume gazda	**com.pub.ro**
adresa pe 32 bit	10000000 01110111 00101000 10111010
zecimal cu punct	128.119.40.186

Procedura pentru biblioteca `inet_addr()` converteste sirul de adrese zecimale cu punct intr-o adresa pe 32 bit

Procedura pentru biblioteca `gethostbyname()` converteste numele din format text in format zecimal cu punct.

Structura sistemului de domenii pe Internet

Administrare ierarhica a numelor. De ex., com.pub.ro

Cel mai din stanga nume este de obicei un nume de gazda (are o adresa IP).

Urmatorul nume este al organizatiei care administreaza aceasta gazda, uneori identificandu-se si cu administratorul tuturor subdomeniilor din stanga - com, tel, tcm (de ex., Universitatea Politehnica Bucuresti).

Cel mai din dreapta nume desemneaza o organizatie, structura, tara, etc.

Domeniu	Utilizare	Exemplu
com	afaceri	teleactivities.com
edu	educational	cs.umass.edu
org	organizatie non-profit	teleactivities.org
net	resurse in retea	teleactivities.net
ro	Romania	com.pub.ro

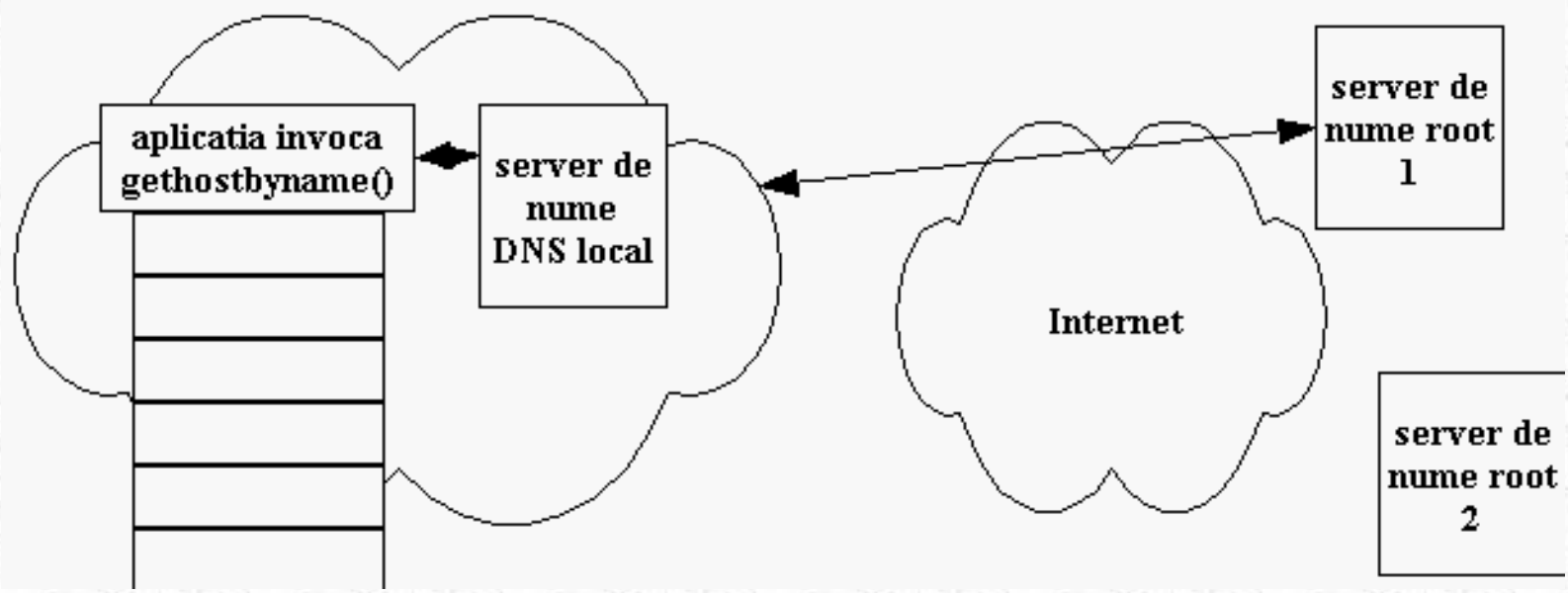
DNS (Domain Name System): sistemul de nume de domenii pe Internet

Reprezinta o **baza de date distribuita** folosita de catre aplicatii TCP/IP pentru a realiza mapari catre/de la nume de gazde de la/catre adrese IP.

Servere de nume

Rutinele de biblioteca la nivel de utilizator `gethostbyname()` si `gethostbyaddress()` contacteaza serverul local de nume via port 53

Serverul de nume returneaza adresa IP a numelui de gazda solicitat.



DNS: nume ne-locale

Gasirea numelor ne-locale

Nici macar un server de nume nu are informatii complete

Daca serverul de nume local nu poate afla adresa, contacteaza radacina serverului de nume:

- exista 9 servere de nume de radacina in toata lumea

- fiecare are adresele serverelor de nume pentru toate serverele de nivel doi (de ex., pub.ro, teleactivities.com)
- serverele de radacina contactate returneaza adresa serverul de nume care trebuie contactat
- serverul de nume de nivel doi contactat poate, la randul lui, sa indice adresarea catre un alt server de nume

Rezolutia numelui este un proces iterativ prin care se urmaresc serverele de nume indicate.

Protocolul DNS specifica formatele de pachete care se schimba cu serverele DNS.

Desemnarea de socket unei adrese de retea: bind()

Fiecare socket trebuie sa fie asociat cu un numar de port pe 16 bit pentru gazda unica.

Trebuie sa se asocieze socketul cu o adresa de retea unica globala (adresa de gazda si port)

- SO stie ca mesajele care vin adresate acestei adrese de gazda si port trebuiesc livrate (demultiplexate catre) acest socket
- o adresa de retur pentru toate mesajele care pleaca.

Numere port	Comentariu
1 - 255	rezervat pentru servicii standard
21	serviciu ftp
23	serviciu telnet
25	SMTP email
80	demon http
1 - 1023	accesibile numai utilizatorilor privilegiati
1024 - 4999	utilizabile de catre sistem si procesele utilizatorului
5000 -	utilizabile numai de catre procesele utilizatorului

Adresarea socketului: structuri de adrese predefinite

Specificarea adreselor socketului: anumite structuri de date predefinite pentru dvs.:

```
struct sockaddr_in { /* INET socket addr info */
short sin_family; /* set me to AF_INET */
 u_short sin_port; /* 16 bit number, nbo */
 struct in_addr sin_addr; /* 32 bit host address */
  char sin_zero[8]; /* not used */
 };
struct in_addr {
u_long s_addr; /* 32 bit host addr.,nbo */
};
```

Apelarea sistemului: bind()

- int bind (int sockfd, struct sockaddr *myaddr, int addresslen)
  - sockfd daca variabila desemnata socket() raspunde cu o valoare
  - *myaddr: structura adresei sockaddr_in care pastreaza informatii despre adresa locala. Trebuie sa ai alocat dreptul pentru a face apelul sockaddr.
  - addresslen este dimensiunea structurii de adresa.

retur eroare indica numar de port deja in uz, iesirea din interval

```
#include <sys/types.h>
#include <sys/socket.h>
#include "inet.h"
 int sockfd;
struct sockaddr_in myaddr;
 if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
          { /* handle error */ }
      myaddr.sin_family = AF_INET;
myaddr.sin_port = htons(5100); /* > 5000 */
 myaddr.sin_addr.s_addr = htonl(INADDR_ANY);
 /* INADDR lets OS determine hostid */
 if ( bind(sockfd, (struct sockaddr *)
&myaddr, sizeof(myaddr)) < 0)

{ /* handle error */ }
```

Exemplu: server fara conexiune

```
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <sys/socket.h>
4 #include <netinet/in.h>
5 #include <arpa/inet.h>
6 #include <errno.h>
7 #define MY_PORT_ID 6090 /* a number > 5000 */
8
9 main()
10 {
11 int sockid, nread, addrlen;
12 struct sockaddr_in my_addr, client_addr;
13 char msg[50];
14
15 printf("Server: creating socket\n");
16 if ( (sockid = socket (AF_INET, SOCK_DGRAM, 0)) < 0)
17 { printf("Server: socket error: %d\n",errno); exit(0); }
18 printf("Server: binding my local socket\n");
19 bzero((char *) &my_addr, sizeof(my_addr));
20 my_addr.sin_family = AF_INET;
21 my_addr.sin_addr.s_addr = htons(INADDR_ANY);
22 my_addr.sin_port = htons(MY_PORT_ID);
23 if ( (bind(sockid, (struct sockaddr *) &my_addr,
24 sizeof(my_addr)) < 0) )
25 { printf("Server: bind fail: %d\n",errno); exit(0); }
26 printf("Server: starting blocking message read\n");
27 nread = recvfrom(sockid,msg,11,0,
28 (struct sockaddr *) &client_addr, &addrlen);
29 printf("Server: retrun code from read is %d\n",nread);
30 if (nread >0) printf("Server: message is: %.11s\n",msg);
31 close(sockid);
32 }
```

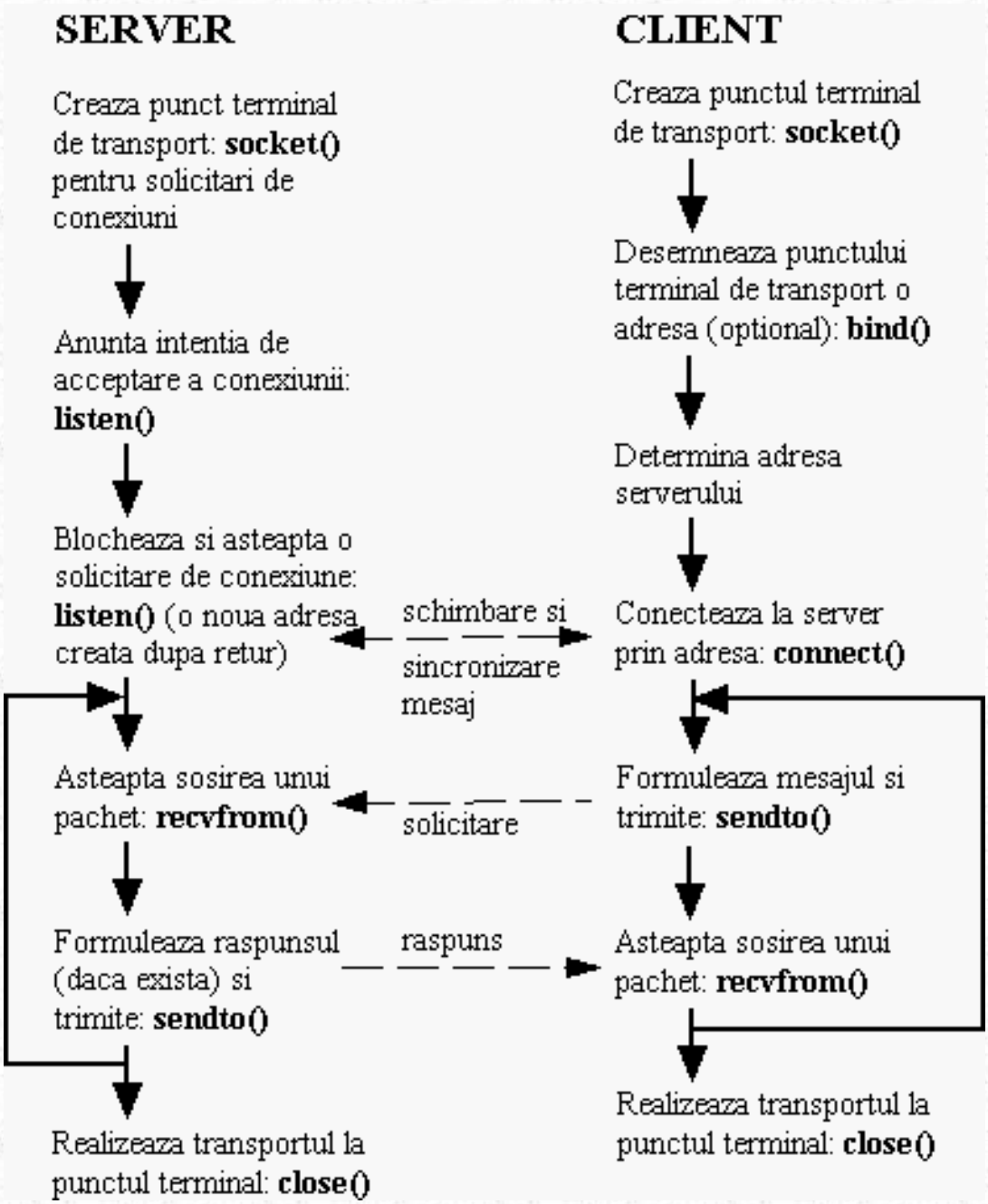
Exemplu: client fara conexiune

```
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <sys/socket.h>
4 #include <netinet/in.h>
5 #include <arpa/inet.h>
6 #include <errno.h>
7 #define MY_PORT_ID 6089
8 #define SERVER_PORT_ID 6090
9 #define SERV_HOST_ADDR
"128.119.40.186"
10
11 main()
12 {
13 int sockid, retcode;
14 struct sockaddr_in my_addr,
server_addr;
15 char msg[12];
16
17 printf("Client: creating socket\n");
18 if ( (sockid = socket(AF_INET,
SOCK_DGRAM, 0)) < 0)
19 { printf("Client: socket failed:
%d\n",errno); exit(0);}
20
21 printf("Client: binding my local
socket\n");
22 bzero((char *) &my_addr,
sizeof(my_addr));
23 my_addr.sin_family = AF_INET;
24 my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
25 my_addr.sin_port = htons(MY_PORT_ID);
26 if ( ( bind(sockid, (struct sockaddr *) &my_addr,
27 sizeof(my_addr)) < 0) )
28 { printf("Client: bind fail: %d\n",errno); exit(0); }
29
30 printf("Client: creating addr structure for server\n");
31 bzero((char *) &server_addr, sizeof(server_addr));
32 server_addr.sin_family = AF_INET;
33 server_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
34 server_addr.sin_port = htons(SERVER_PORT_ID);
35
36 printf("Client: initializing message and sending\n");
37 sprintf(msg, "Hello world");
38 retcode = sendto(sockid,msg,12,0,(struct sockaddr *) &server_addr,
39 sizeof(server_addr));
40 if (retcode <= -1)
41 {printf("client: sendto failed: %d\n",errno); exit(0);}
42
43 /* close socket */
44 close(sockid);
45 }
```

Exemplu: executarea comenzii trace

```
> cc udpserver.c; mv a.out udpserver
> cc udpclient.c; mv a.out udpclient
> udpserver &
  [1] 20837
> Server: creating socket
Server: binding my local socket
Server: starting blocking message read
  > udpclient
Client: creating socket
Client: binding my local socket
Client: creating addr structure for server
Client: initializing message and sending
 Server: return code from read is 11
Server: message is: Hello world
[1] Done udpserver
```

Serviciu cu conexiune orientata



Reguli pentru conexiunea client/server:

- clientul trebuie sa se conecteze in mod explicit la server inainte de a transmite sau primi date
- clientul nu va accepta connect() pana cand serverul nu accepta clientul
- serverul trebuie sa accepte in mod explicit clientul inainte de a transmite sau primi date
- serverul va astepta cu accept() pana la connect() a clientului.

In cazul serviciului cu conexiune orientata, serviciul de transport de baza este **sigur.**

Conexiunea client-to-server: connect()

Clientul foloseste `connect()` pentru a solicita conexiunea la server si comunicarea via socket

Protocolul de transport de baza (de ex. TCP) incepe protocolul de setare a conexiunii implementand regulile client/server

`connect()` se returneaza cand serverul accepta explicit conexiunea, sau in caz de pauza (nu exista niciun raspuns de la server).

Se foloseste de obicei cu protocoalele de transport sigure, dar si cu datagrame.

```
int connect ( int sockfd, struct sockaddr *toaddrptr, int addresslen)
```

- `sockfd`: variabila desemnata `socket()` returneaza valoare. Procesul accepta conexiuni care sosesc pe acest socket id.
- `*toaddrptr` este structura adresei lui sockaddr_in care pastreaza informatii despre adresa serverului. Este nevoie de alocarea dreptului necesqar pentru a tipari sockaddr.
- `addresslen` este dimensiunea structurii adresei.

Apelul de sistem listen()

Folosit de catre server cu conexiune orientata.

Permite SO/retelei sa afle daca serverul va accepta solicitarea de conexiune.

Nu blocheaza si asteapta solicitarea!

int listen (int sockfd, int maxwaiting)

sockfd: valoare de retur asignata variabil lui socket(). Procesul accepta conexiuni sosind pe acest socket id.

maxwaiting:numarul maxim de solicitari de conexiune care poate fi pus in asteptare pentru ca serverul sa dea un accept(). Valoarea tipica este 5.

Conexiunea server-to-client: accept()

Realizata de catre server, dupa listen().

Serverul va accept() solicitarea de conexiune via socketul specificat, si va returna *newly_created socket* pentru utilizarea in comunicarea retur pentru clientul cu accept().

Serverul are un socket pentru acceptarea solicitarilor de conexiuni care sosesc.

- creaza alti (noi) socketi pentru comunicarea cu clientii

Serverul nu poate accept() selectiv solicitarile de conexiune.

int accept (int sockfd, struct sockaddr *fromaddrptr, int *addresslen)

`sockfd` este valoarea de retur variabil asignata a socket()}.

`*fromaddrptr` este adresa structurii sockaddr_in continand informatii despre adresa socketului care transmite datele. Este o valoare de retur.

`addresslen` este dimensiunea structurii adresei. *Un argument cu valoare rezultata* (se ajusteaza inainte de apel, se reseteaza in timpul apelului).

```
struct sockaddr_in other_app_addr;
```

```
int sockid, newsockid, addrsize;
...
addrsize = sizesizeof(other_app_addr));
newsockid = accept(sockfd, (struct sockaddr *)
&other_app_addr, &addrsize);
/* newsockid to communicate with client,
sockid to accept more connections */
```

O aplicatie simpla cu evolutie in timp: 1

Client:

- conectare la server
- transmite clientului server timpul local
- citeste returul privind timpul local al serverului.

Server:

- primeste conexiuni de la clienti
- afiseaza timpul local al clientului
- transmite timpul local al serverului clientului.

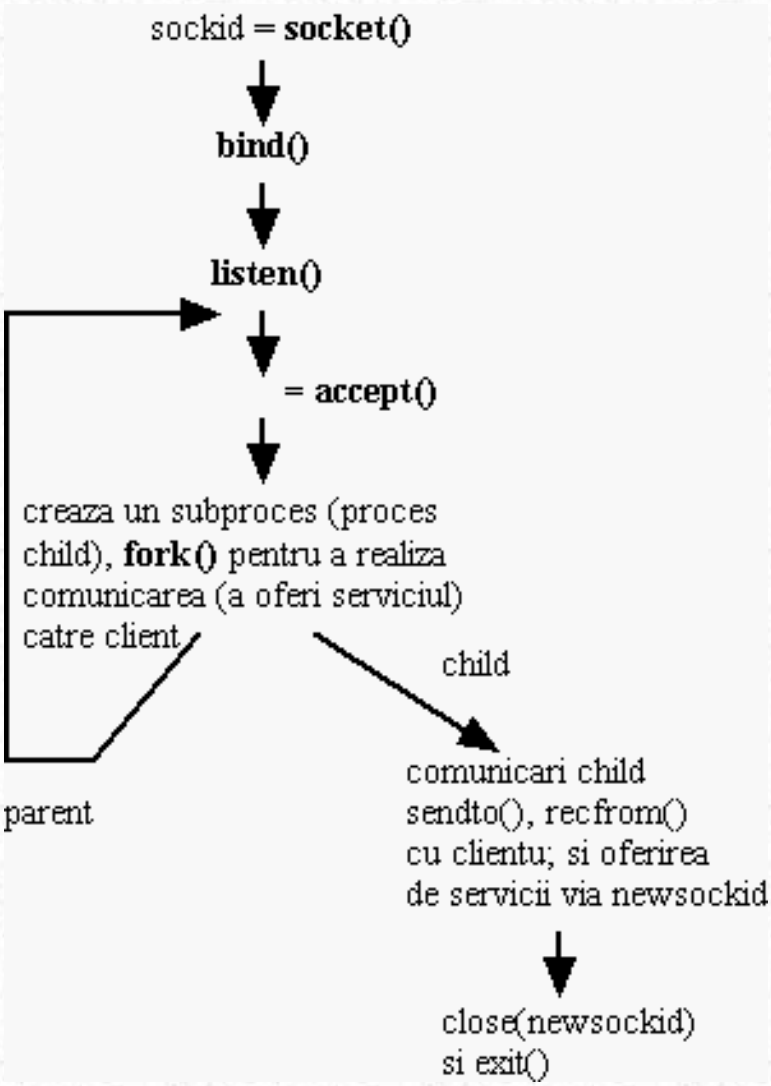
O aplicatie simpla cu evolutie in timp: codul client

```
1
2 #include <sys/types.h>
3 #include <sys/socket.h>
4 #include <netinet/in.h>
5 #include <arpa/inet.h>
6 #include <time.h>
7 #include <errno.h>
8 #define MY_PORT_ID 6090 /* a number >
5000 */
9 #define SERV_HOST_ADDR
"128.119.40.186" /* Jim's host machine */
10 main()
11 {
12 int sockid;
13 struct sockaddr_in ssock_addr;
14 struct timeval tp;
15 struct timezone tzp;
16
17 /* create a socket */
18 if ( (sockid = socket(AF_INET,
SOCK_STREAM, 0)) < 0)
19 { printf("error creating client socket,
error%d\n",errno); exit(0); }
20
21
22 /* name the socket using wildcards*/
23 bzero((char *) &ssock_addr, sizeof(ssock_addr));
24 ssock_addr.sin_family = AF_INET;
25 ssock_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
26 ssock_addr.sin_port = htons(MY_PORT_ID);
27
28 if (bind (sockid, (struct sockaddr *) &ssock_addr,
sizeof(ssock_addr)) < 0)
29 { printf("error connecting to server, error: %d\n",errno); exit(0); }
30 /* send time of day */
31 gettimeofday(&tp,&tzp);
32 /* convert from host octet order to network octet order */
33 printf("client: local time is %ld\n",tp.tv_sec);
34 tp.tv_sec = htonl(tp.tv_sec);
35 tp.tv_usec = htonl(tp.tv_usec);
36
37 /* send time of day to other side */
38 write(sockid, &tp, sizeof(tp));
39 /* get time of day back fro other side and display */
40 if ( (read(sockid, &tp, sizeof(tp))) < 0)
41 { printf("error reading new socket\n"); exit(0); }
42
43 /* convert from network octet order to host octet order */
44 tp.tv_sec = ntohl(tp.tv_sec);
45 tp.tv_usec = ntohl(tp.tv_usec);
46 printf("client: remote time is %ld\n",tp.tv_sec);
47 close(sockid);
```

O aplicatie simpla cu evolutie in timp: cod server

```
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <sys/socket.h>
4 #include <netinet/in.h>
5 #include <arpa/inet.h>
6 #include <time.h>
7 #include <errno.h>
8 #define MY_PORT_ID 6090 /* a number > 5000 */
9
10 main()
11 {
12 int sockid, newsockid, i,j;
13 struct sockaddr_in ssock_addr;
14 struct timeval tp;
15 struct timezone tzp;
16
17 /* create a socket */
18 if ( (sockid = socket (AF_INET, SOCK_STREAM, 0)) < 0)
19 { printf("error creating socket, error: %d\n",errno);
exit(0); }
20 /* do a man on errno to get error information */
21 /* name the socket using wildcards */
22 bzero((char *) &ssock_addr, sizeof(ssock_addr));
23 ssock_addr.sin_family = AF_INET;
24 ssock_addr.sin_addr.s_addr = htonl(INADDR_ANY);
> 25 ssock_addr.sin_port = htons(MY_PORT_ID);
26 /* bind the socket to port address */
27 if ( ( bind(sockid, (struct sockaddr *) &ssock_addr,
sizeof(ssock_addr)) < 0) )
28 { printf("error binding socket, error: %d\n",errno);
exit(0); }
29 /* start accepting connections */
30 if ( listen(sockid, 5) < 0)
31 { printf("error listening: %d\n",errno); exit(0); }
32
33 for (i=1; i<=50000 ;i++) {
34 /* accept a connection */
35 newsockid = accept(sockid, (struct sockaddr *)0,
(int *)0);
36 if (newsockid < 0)
37 { printf("error accepting socket, error:
%d\n",errno); exit(0); }
38 /* read remote time of day from socket */
39 if ( (read(newsockid, &tp, sizeof(tp))) < 0)
40 { printf("error reading new socket\n"); exit(0); }
41 /* convert from network octet order to host octet
order */
42 tp.tv_sec = ntohl(tp.tv_sec);
43 tp.tv_usec = ntohl(tp.tv_usec);
44 printf("server: remote time is %ld\n",tp.tv_sec);
45
46 /* get local time of day and send to client*/
47 for (j=0; j<1000000; j++)
48 ; /* delay */
49 gettimeofday(&tp,&tzp);
50 /* convert from host octet order to network octet
order */
51 printf("server: local time is %ld\n",tp.tv_sec);
52 tp.tv_sec = htonl(tp.tv_sec);
53 tp.tv_usec = htonl(tp.tv_usec);
54 write(newsockid, &tp, sizeof(tp));
55 close(newsockid);
56 }
57 close(sockid);
```

Structura tipica a serverului



Structura serverului: fragmentul de cod

```
int sockfd, newsockfd;
 if ( (sockfd = socket( ... )) < 0) /* create socket */
.....
if ( bind(sockfd,... ) < 0) /* bind socket */
.....
if ( listen(sockfd,5) < 0) /* announce we're ready */
.....
while (1==1) { /* loop forever */
  newsockfd = accept(sockfd, ...); /* wait for client */
  if ( (pid = fork()) == 0) {
     /* child code begins here */
     close(sockfd); /* child doesn't wait for client */
     ......
     /* child does work here, communicating
         with client using the newsockfd */
     .......
     exit(0); /* child dies here */
     }
```

```
/* parent continues execution below */
close(newsockfd); /* parent won't communicate with */
/* client - that's childsplay! */
} /* end of while */
```

Transmiterea si receptia datelor

Datele transmise/receptionate folosind apelurile i/o ale sistemului standard UNIX sau apelurile sistemului specific retelei

Apelul sistemului	Dispoyitiv	Buferare	Optiuni	Se specifica legatura?
read()/write()	oricare i/o	unica	N	N
readv()/writev()	oricare i/o	scatter/gather	N	N
recv()/send()	socket	unica	D	N
recvfrom()/sendto()	socket	unica	D	D
recvmsg()/sendmsg()	socket	scatter/gather	D	D

Transmisia via un socket: sendto()

```
int sendto (int sockfd, char
*buff, int bufflen, int flags,
            struct sockaddr *toaddrptr, int addresslen)
```

sockfd este valoarea de retur asignata variabil a socket()

*buff este un set de locatii de memorie consecutive pastrand mesajul pentru a fi transmis

bufflen este numarul de octets care trebuiesc transmisi.

Steagurile (flags) pot fi folosite pentru a specifica optiunile. Intotdeauna 0 pentru noi.

*toaddrptr este structura de adresa pentru sockaddr_in care pastreaza informatiile despre adresele de destinatie. Trebuie sa ai permisiunea pentru a putea scrie sockaddr.

addresslen easte dimensiunea structurii adresei.

Codul de retur OK NU implica date corect livrate, numai corect transmise

Citirea/scrierea socketilor: exemple

Exemplu: folosind sendto()
```
char buffer[50];
struct sockaddr_in other_app_addr;
int retcode

/* suppose we have done socket() and bind()
calls, filled in other_app_addr,and put
23 octets of data into buffer */

retcode = sendto(sockfd, buffer, 23, 0,
(struct sockaddr *) &other_app_addr,
sizeof(other_app_addr))
```

Exemplu: folosind recvfrom()
```
char buffer[50];
struct sockaddr_in other_app_addr;
int nread, addrlen;

/* suppose we have done socket(), bind() */
nread = recvfrom(sockfd, buffer, 23, 0,
(struct sockaddr *) &other_app_addr, &addrlen))
```

Citirea de la un socket: recvfrom()

```
int recvfrom (int sockfd, char *buff, int bufflen,
int flags, struct sockaddr *fromaddrptr,
int *addresslen)
```

sockfd este valoarea de retur asignata a socket()

*buff este un set de locatii de memorie consecutive in care sunt primite datele pentru a fi scrise

bufflen este numarul de octets de citit

Steagurile (flags) pot fi folosite pentru optiuni specifice. Intotdeauna sunt 0 pentru noi.

*fromaddrptr este structura de adresa pentru isockaddr_in continand informatii despre adresa socketului care transmite aceste date. Este o valoare returnata

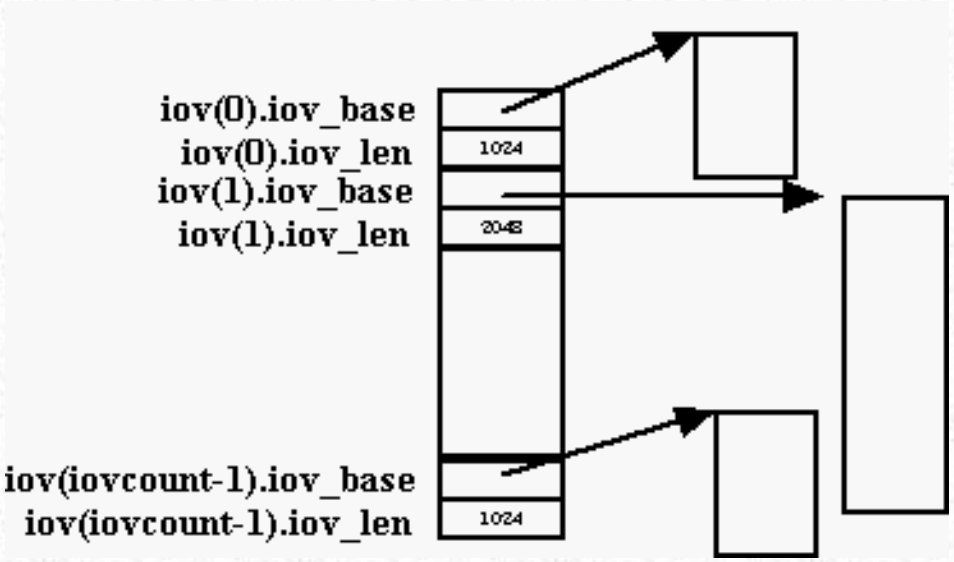
addresslen este dimensiunea structurii adresei. Este o valoare returnata.

recvfrom() returneaza octetii de numar primiti in prezent.

Scatter-ul citeste si gather-ul scrie

Folosite pentru a citi/scrie in/din multiple bufere ne-adiacente

```
writev(int sd, struct iovec iov[], int iovcount);
readv(int sd, struct iovec iov[], int iovcount);


struct iovec{
caddr_t iov_base; /* starting addr of this buffer */
int     iov_len;  /* num. octets in this buffer */
};
```



Rutine de ordonare a octetilor

Rutinele de ordonare a octetilor convertesc in/din numere intregi de 16 si 32 bit din/in "numere de ordine a octetilor in retea".

- computere diferite pot stoca octeti ai numerelor intregi in diferite moduri de ordonare in memorie
- reprezentarea interna a lui 2^{24}
- ordinea octetilor in retea este cu numerele mari la final
- cantitatile intregi (precum informatiile de adresare) trebuie sa fie explicit convertite in/din ordinea octetilor in retea (network octet order (nbo)).

Numerele mari la final (M6800, IBM370)			
MSB			LSB
00000001	00000000	00000000	00000000
x	x + 1	x + 2	x + 3
Numerele mici la final (80x86, VAX)			
00000000	00000000	00000000	00000001
x	x + 1	x + 2	x + 3

Numele functiei	Actiunea
htonl	converteste formatul gazda 32-bit in nbo
ntohl	converteste nbo in formatul gazda 32-bit
htons	converteste formatul gazda 16-bit in nbo
ntos	converteste nbo in formatul gazda 16-bit

Alte apeluri de sistem si rutine utile

close(sockfd) va realiza un socket.

Apelurile de sistem getsockopt() si setsockopt() se folosesc pentru a interoga/seta optiunile sistemului.

Apelul de sistem ioctl() se foloseste pentru a interoga/seta atributele socketului si, de asemenea, atributele interfetei dispozitivului de retea.

I/O asincrone

Socketul citeste ceea ce noi am vazut ca se blocheaza.

Aplicatia poate fi notificata de catre SO cand datele sunt gata pentru a fi citite folosind i/o asincrone:

- scrie o procedura usor de aplicat: apelat de SO cand datele sunt gata pentru a fi citite
- informeaza SO despre identitatea procesului si numele procedurii care va fi apelata
- permite i/o asincrone

I/O asincrone: exemplu

```
#include <signal.h>
#include <fnctl.h>
int datahere = 0;

main() {
/* create, bind, connect/accept, etc done here */
signal(SIGIO,read_proc);
/* inform OS: call read_proc when data arrives */
fnctl(sd, F_SETOWN, getpid())
/* inform OS: procedure is inside ``me'' */
fnctl(sd, F_SETFL,  FASYNCH)
/* enable asynchronous I/O */

while (1=2) {
.....
/ * do useful work, possibly checking datahere
flag.  Note there is NO EXPLICIT CALL to
read_proc() */
}
}

read_proc()  /* called by OS on data arrival */
{
datahere = 1;
/* possibly do other useful work */
}
```

Apelul de sistem select()

Aplicatia poate raspunde la i/o noi (de ex., date, solicitari de conectare) care apar pe mai multi socketi diferiti.

Nu intentioneaza sa blocheze nici macar un accept(), recvmsg()

Multiplexarea I/O folosind select():

- informeaza SO despre setul socketilor care ne intereseaza
- apeleaza select()
- select() ne va spune care socketi sunt gata pentru I/O
- creaza I/O (de ex., accept(), recvmsg()) pe socket corespunzator

Programele macro FD_ZERO, FD_SET, FD_CLEAR, FD_ISSET alimentate prin SO se folosesc de catre aplicatii pentru a declara si determina pregatirea socketilor.

```
int select(int maxsockets, fd_set *readytoread,
     fd_set *readytowrite, fd_set *readyexceptions,
               struct timeval *timeptr);
```

Apelul de sistem select(): exemplu simplu

```
#include <sys/types.h>
        main(){
    fd_set readset;
      int nready;

/* open, bind, connect/accept etc. sockets
            sd1, sd2, sd3 */
/* tell OS sockets we're interested in */
            FD_ZERO(&readset);
           FD_SET(sd1,&readset);
           FD_SET(sd2,&readset);
           FD_SET(sd3,&readset);

/* call select, returning when something's ready */
nready = select(64, &readset, NULL, NULL, NULL);
       /* read from appropriate socket */
          if (FD_ISSET(sd1, &readset))
            ..... /* do i/o from sd1 */
       else if (FD_ISSET(sd2, &readset))
            ..... /* do i/o from sd2 */
       else if (FD_ISSET(sd3, &readset))
            ..... /* do i/o from sd3 */
```

Crearea unui client si server ftp utilizand socketi

Stim acum suficient pentru a construi aplicatii client/server sofisticate:

- ftp
- telnet
- smtp
- http

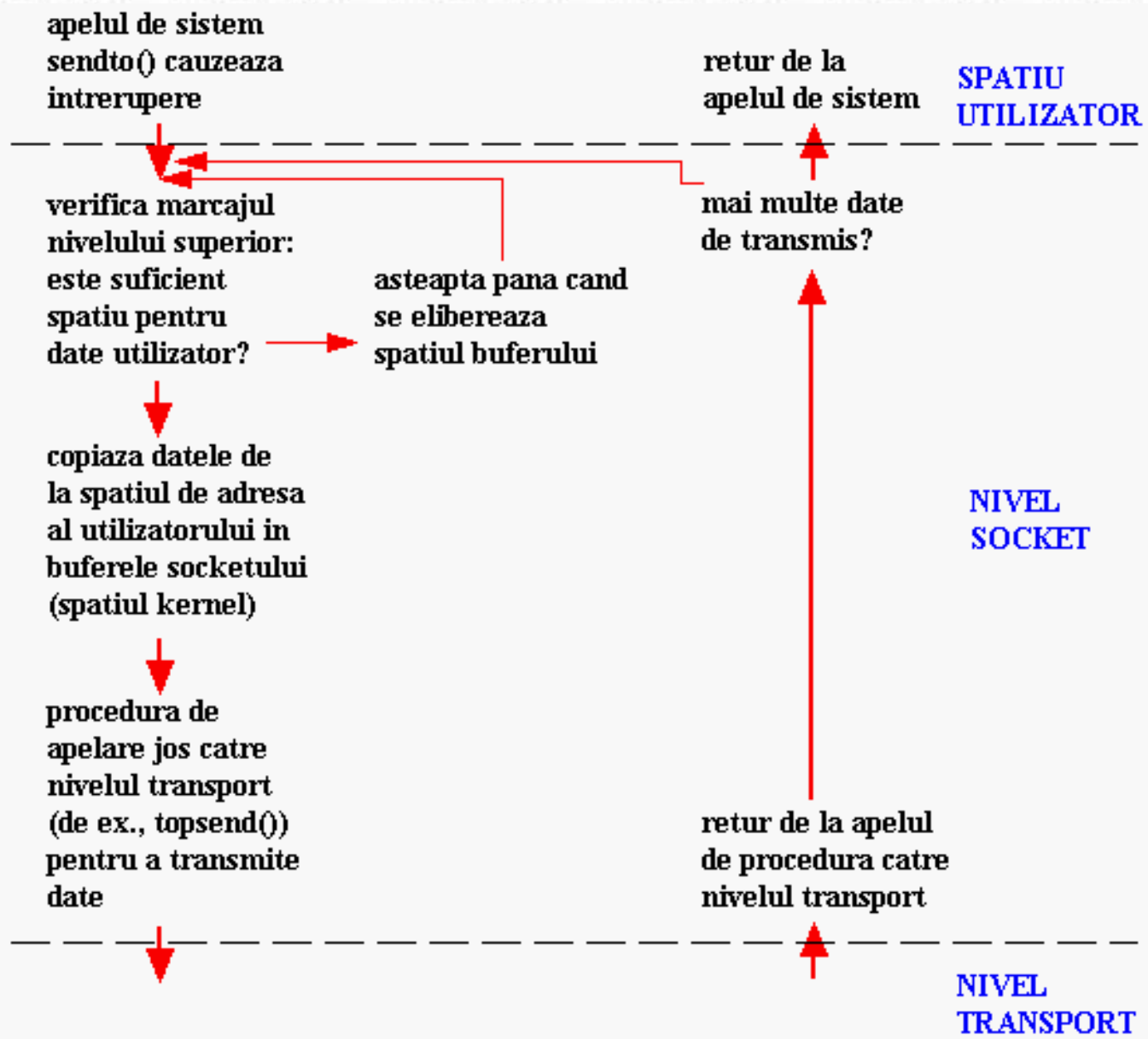
Implementare: structura de date a socketului

Iata o astfel de structura de date per socket in cadrul SO:

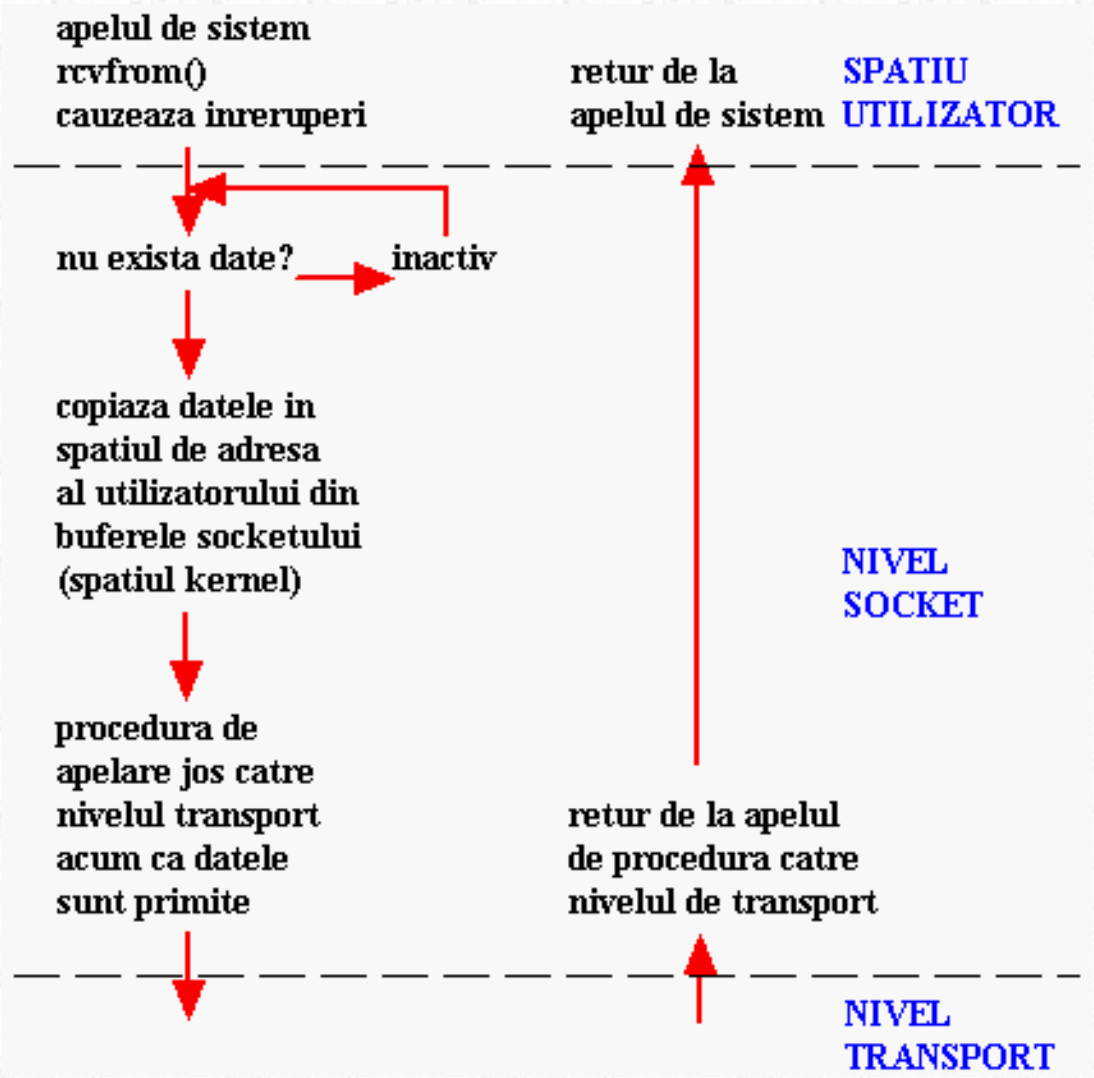
Campurile structurii socketului
tipul socketului
optiunile socketului (de la socket())
timpul de "intarziere"
starea socketului (ISCONNECTED, ISCONNECTING,...)
indicatorul pentru informatii privind protocolul de nivel inferior
indicatorul pentru socketul accept()
sir, numarul de conexiuni partial formate
numarul maxim de conexiuni in asteptare
sir, numarul de conexiuni sosite aflate in asteptare
evenimentul amanat daca nu exista nicio conexiune care sa soseasca
flagurile de eroare
informatii de semnalare
indicator de depasire a limitei de banda

bufer de transmisie	campuri/caracteristici
	numarul de octeti in bufer
	marcator al nivelului de sus
	marcator al nivelului de jos
	indicator catre zona buferelor (mbufs)
	transmisia/receptia
	flaguri
bufer de receptie	aceleasi campuri ca la cel de transmisie

Implementare: actiunile SO asupra sendto()



Implementare: actiunile SO asupra rcvfrom()



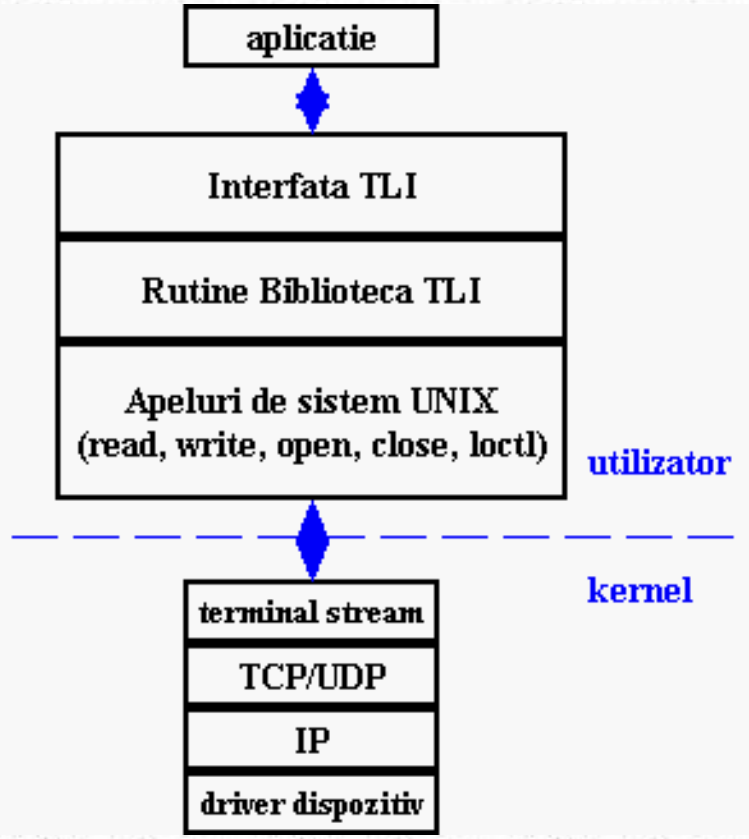
UNIX SVR4: Interfata TLI

Interfata Nivelului de Transport (TLI) pentru Sistemul Unix V, Release 4 (svr4)

Ca socketi:

- independent de transport
- doua modele: orientat pe conexiuni, datagrame
- "terminalul de transport endpoint" se aseamana cu un socket

TLI este implementata ca rutine de bibiloteca C la nivel utilizator, mai degraba decat apeluri de sistem.



Comparatie pentru unele TLI versus socket

Functia TLI	Apelul BSD cel mai apropiat	Comentarii
t_open()	socket()	
t_bind()	bind(), listen()	
t_listen()	accept()	asteapta solicitarea dar nu o accepta. Returneaza informatia la client
t_snddis()		refuza clientul fara accept
t_accept()	accept()	TLI nu va crea un nou terminal. BSD accept()=t_list+t_open+t_accept
t_connect()	connect()	
t_send()	send()	
t_recv()	recv()	
t_sendudata()	sendto()	
t_recvudata()	rcvfrom()	
t_look()		ia evenimentul curent de la conexiunea transport
t_getstate()		ia starea curena de la conexiunea transport

Socketi Windows

Se bazeaza pe socketii BSD:

- BSD: standardul de facto pentru retelistica TCP/IP'
- suporta modelul stream(TCP)/datagram(UDP)

- IPA este aceeasi

Cateva diferente/incompatibilitati:

- extensii pentru programarea asincrona
- coduri de retur pentru eroarea de diferenta: -1 nu este codul de retur al erorii!
- identificatorul socketului este diferit de identificatorul fisierului
- read(), write(), close() nu trebuie utilizati
- folositi in schimb echivalentii specifici socketului.

Socketii Windows; apeluri de sistem

Specificatiile pentru socketii Windows includ urmatoarele rutine de socket tip Berkeley:

accept() *	O conexiune la intrare este recunoscuta si asociata cu un socket creat imediat. Socketul original este returnat in starea de ascultare.
bind()	Asigneaza un nume local unui socket nenumit.
closesocket() *	Inlatura un socket din tabela de referinta a procesului. Numai blocurile daca SO_LINGER este ajustat.
connect() *	Initiaza o conexiune pe socketul specificat.
getpeername()	Restabileste numele perechii conectate la socketul specificat.
getsockname()	Restabileste numele curent pentru socketul specificat.
getsockopt()	Restabileste optiunile asociate cu socketul specificat.
htonl()	Converteste o cantitate de 32 bit din ordinea de octete a gazdei in cea a retelei.
htons()	Converteste o cantitate de 16 bit din ordinea de bit a gazdei in cea a retelei.
inet_addr()	Converteste un sir de caractere reprezentand un numar in notatia Internet "." standard intr-o valoare de adresa pe Internet.
inet_ntoa()	Converteste o valoare de adresa pe Internet intr-un sir ASCII in notatia ".", de ex. "a.b.c.d".
ioctlsocket()	Ofera controlul socketilor.
listen()	Asculta conexiunile de intrare pe un socket specificat.
ntohl()	Converteste o cantitate de 32 bit din ordinea bzte de retea in cea de gazda.
ntohs()	Converteste o cantitate de 16 bit din ordinea bzte de retea in cea de gazda
recv() *	Primeste date de la un socket conectat.
recvfrom() *	Primeste date de la un socket conectat sau neconectat.
select() *	Realizeaza multiplexarea sincrona I/O.
send() *	Transmite date la un socket conectat.
sendto() *	Transmite date la un socket conectat sau neconectat.
setsockopt()	Stocheaza optiunile asociate cu socketul specificat.
shutdown()	Inchide partial o conexiune full-duplex.
socket()	Creaza un terminal pentru comunicatie si retirneaza un socket.

* = Rutina poate bloca daca actioneaza asupra unui socket de blocare.acting on a blocking socket.

Socketi Windows: functii pentru baza de date

gethostbyaddr() *	Restabileste numele si adresa corespunzatoare unei adrese de retea.

gethostbyname() *	Restabileste numele si adresa corespunzatoare unui nume de gazda.
gethostname()	Restabileste numele gazdei locale.
getprotobyname() *	Restabileste numele si numarul de protocol corespunzator unui nume de protocol.
getprotobynumber() *	Restabileste numele si numarul de protocol corespunzator unui numar de protocol.
getservbyname() *	Restabileste numele de serviciu si portul corespunzator unui nume de serviciu.
getservbyport() *	Restabileste numele de serviciu si portul corespunzator unui port.

*** =** Rutina poate bloca in anumite circumstante.

Socketi Windows: asincronie

Suportul asincron permite programatorului sa specifice rutina care trebuie apelata cand se petrec evenimente care au legatura cu socketul:

- socketul gata pentru citire/scriere
- datele care depasesc limita de banda gata pentru a fi citite
- socketul gata pentru a accepta conexiunea
- conexiune inchisa:

Aceste *extensii* sunt de ajutor cu programarea concurenta, multifir

WSAAsyncGetHostByAddr()	Un set de functii care ofera versiuni asincrone ale functiilor Berkeley standard **getXbyY()**. De exemplu, functia **WSAAsyncGetHostByName()** da un mesaj asincron bazat pe implementarea functiei Berkeley standard **gethostbyname()**.
WSAAsyncGetHostByName()	
WSAAsyncGetProtoByName()	
WSAAsyncGetProtoByNumber()	
WSAAsyncGetServByName()	
WSAAsyncGetServByPort()	
WSAAsyncSelect()	Executa versiunea asincrona a lui **select()**
WSACancelAsyncRequest()	Anuleaza o solicitare a unei functii **WSAAsyncGetXByY().**
WSACancelBlockingCall()	Anuleaza o "blocare" a apelului IPA.
WSACleanup()	Iesire din substratul DLL al socketilor Windows.
WSAGetLastError()	Obtine detalii despre ultima eroare IPA a socketilor Windows.
WSAIsBlocking()	Afla daca substratul DLL al socketilor Windows blocheaza deja un apel existent pentru acest fir.
WSASetBlockingHook()	"Agata" (hook) metoda de blocare folosita de substratul implementare al socketilor Windows.
WSASetLastError()	Ajusteaza eroarea care trebuie returnata de urmatoarea **WSAGetLastError()**
WSAStartup()	Initializeaza substratul DLL al socketilor Windows.
WSAUnhookBlockingHook()	Restaureaza functia de blocare originala.

Programarea retelei in Java

Abstractizarea IPA a retelei: socket

Clasele de servicii de transport sunt aceleasi cu socketii:

- datagrama: UDP
- orientata pe conexiune: TCP

O interfata de nivel putin mai inalt decat in cazul socketilor UNIX.

- sunt de asemenea disponibile apeluri de sistem de nivel scazut (cei pentru UNIX)

Unele consideratii specifice Java:

- apletii se pot conecta la servere numai din locul lor de origine (pentru securitate)
- cosul JAVA colecteaza obiectele nefolosite.

Referinte: Java in a Nutshell, D. Flanagan, O'Reily and Associates, 1996

Transmiterea unei datagrame in Java

Creaza un pachet DatagramPacket, specificand date, lungimea, adresa IP si portul destinatarului.

Invoca metoda (procedura) send() a socketului DatagramSocket

```
// This example is from the book _Java in a Nutshell_ by David Flanagan.
// Written by David Flanagan. Copyright (c) 1996 O'Reilly & Associates.
// You may study, use, modify, and distribute this example for any purpose.
// This example is provided WITHOUT WARRANTY either expressed or implied.
import java.io.*;
import java.net.*;

// This class sends the specified text as a datagram to port 6010 of the
// specified host.
public class UDPSend {
static final int port = 6010;
public static void main(String args[]) throws Exception {
if (args.length != 2) {
System.out.println("Usage: java UDPSend ");
System.exit(0);
}

// Get the internet address of the specified host
InetAddress address = InetAddress.getByName(args[0]);
// Convert the message to an array of octetes
int msglen = args[1].length();
octete[] message = new octete[msglen];
args[1].getoctetes(0, msglen, message, 0);
// Initilize the packet with data and address
DatagramPacket packet = new DatagramPacket(message, msglen,
address, port);
// Create a socket, and send the packet through it.
DatagramSocket socket = new DatagramSocket();
socket.send(packet);
}
}
```

Receptia unei datagrame in Java

Creaza un pachet DatagramPacket cu un bufer pentru a receptiona pachetul.

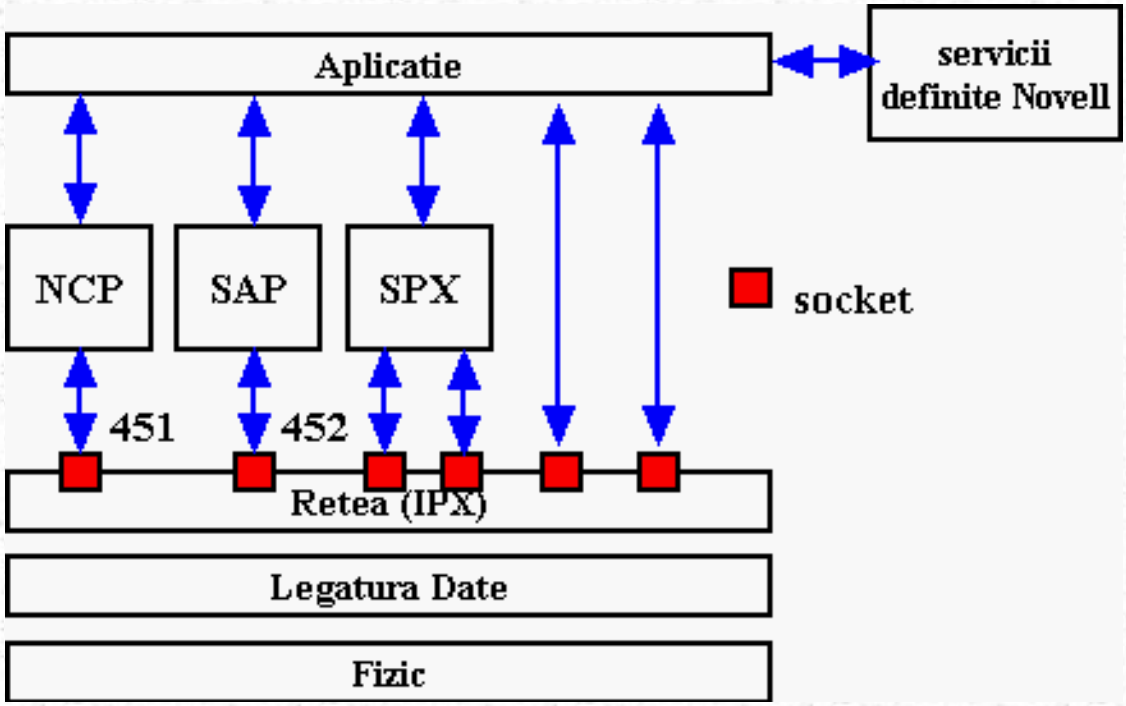
Creaza un socket DatagramSocket care va "astepta"pachetul.

Invoca metoda (procedura) receive() a socketului DatagramSocket

```
// This example is from the book _Java in a Nutshell_ by David Flanagan.
// Written by David Flanagan.  Copyright (c) 1996 O'Reilly & Associates.
// You may study, use, modify, and distribute this example for any purpose.
// This example is provided WITHOUT WARRANTY either expressed or implied.

import java.io.*;
import java.net.*;

// This program waits to receive datagrams sent to port 6010.
// When it receives one, it displays the sending host and port,
// and prints the contents of the datagram as a string.
public class UDPReceive {
static final int port = 6010;
public static void main(String args[]) throws Exception
{
octete[] buffer = new octete[1024];
String s;
// Create a packet with an empty buffer to receive data
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
// Create a socket to listen on the port.
DatagramSocket socket = new DatagramSocket(port);

for(;;) {
// Wait to receive a datagram
socket.receive(packet);
// Convert the contents to a string
s = new String(buffer, 0, 0, packet.getLength());
```

```
            // And display them
        System.out.println("UDPReceive: received from " +
                packet.getAddress().getHostName() + ":" +
                    packet.getPort() + ": " + s);
                }
            }
        }
```

Retele Novell: IPA Netware



Adopta paradigma client-server.

Transportul de date: datagrama este nesigura (IPX), orientarea pe conexiune este sigura (SPX)]

SAP: Protocolul de Avertizare a Serviciului (Service Advertising Protocol) folosit pentru servicii de avertizare/solicitare.

NCP: Protocolul Principal de Retea (Netware Core Protocol) folosit pentru interactii client server definite Netware.

Servicii definite Netware de nivel superior:

- servicii de director
- suport pentru tranzactii
- controlul resurselor
- servicii de inchidere.

Protocolul SAP Novell

Permite clientilor/serverelor sa localizeze serviciile de avertizare.

Ruterele si portile mentin tabela de servicii cunoscute.

Serviciu periodic de difuzare a informatiilor.

Diferit fata de modelul Internet: reteaua (ruterele) interne Novell au informatii la nivel de aplicatie.

SAP difuzeaza pachetele IPX. Model de format:

operatie	tip serviciu	nume server	adresa retea	adresa gazda	adresa socket	salturi

Tipurile de servicii sunt definite prin Novell.

Operatii posibile:

tip serviciu	semnificatie
1	solicita numele/adresa tuturor serverelor din tipul specificat
2	raspuns la solicitarea de tip 1, periodic difuzat de server si ruter
3	la fel ca la 1, dar pentru serviciul cel mai apropiat
4	raaspuns la solicitarea de tip 3

Entitatile serviciului pentru aplicatia OSI

Aplicatiile pot apela (comunica cu) entitati existente predefinite care ofera un serviciu.

Elemente de siguranta ale serviciului de transfer:

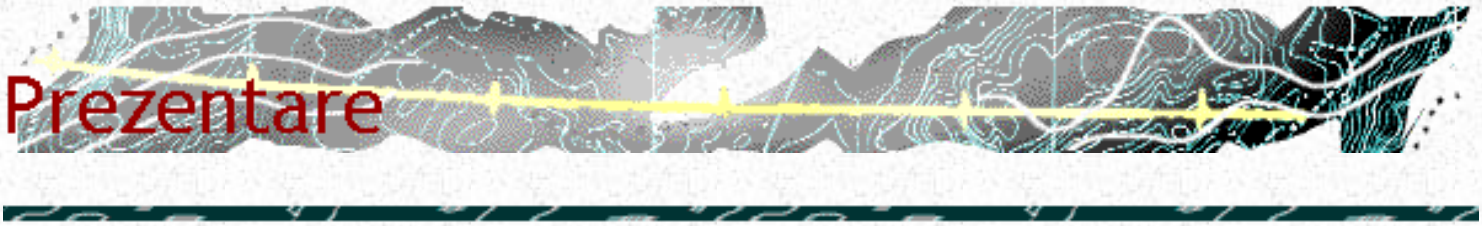
- *punct de control si recuperare*: transfer sigur de date fata de vulnerabilitatile conexiunii de retea
- *comunicare cu doua moduri de alternanta:* confirma transferul si receptioneaza inainte de a transmite urmatorul mesaj.

Obligativitatea, concurenta si recuperarea ofera o actiune la nivel atomic:

- ajustarea schimburilor de mesaje si prelucrarea tuturor garantiilor pentru a realiza completitudinea sau altceva daca nu se realizeaza nici o actiune
- nicio instabilitate nu trebuie sa deregleze reteaua

Back Top Next

Retele, protocoale si servicii asigurate prin retelele de calculatoare



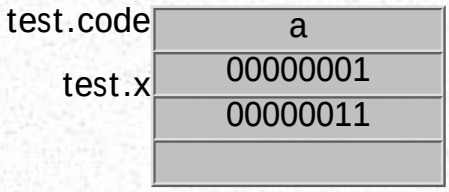
4. Servicii de prezentare

Motivatie

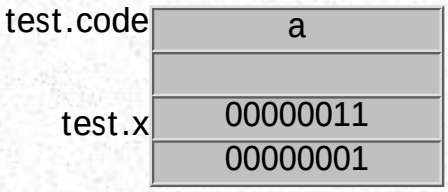
Esenta problemei:

- avem de-a face cu *semnificatia* informatiei, nu cu *reprezentarea*
- diferite calculatoare, SO, compilere au conventii diferite pentru datele de reprezentare
  - arhitectura: numere mari la sfarsit, fata de numere mici la sfarsit
  - format de punct flotant
  - dimensiunea tipului de date: 16, 32, 64 bit
  - dimensiuni si layout diferite ale structurilor de date

```
struct{
  char code;
  int x;
  } test;
test. x = 259;
test.code = 'a';
```

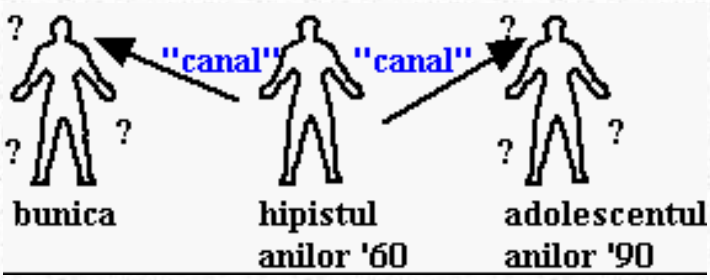


masina X

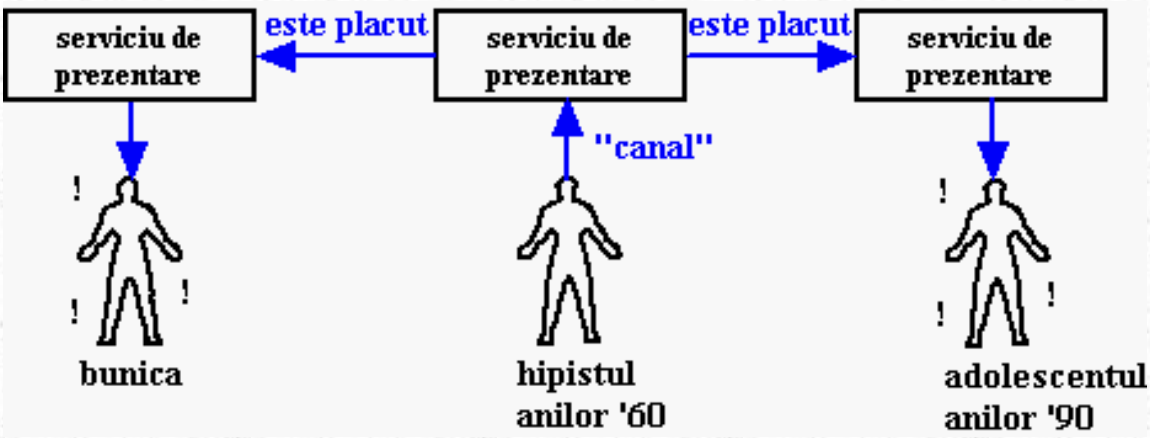


masina Y

Rezolavrea problemei de reprezentare



- expeditorul codeaza in formatul destinatarului
- destinatarul decodeaza din formatul expeditorului
- metoda independenta pentru masina + SO + limbaj, de descriere a structurii de date
  - gazda traduce in/din limbajul de descriere universal din/in formatul propriu



ASN.1: Notatia 1 pentru sintaxa abstracta

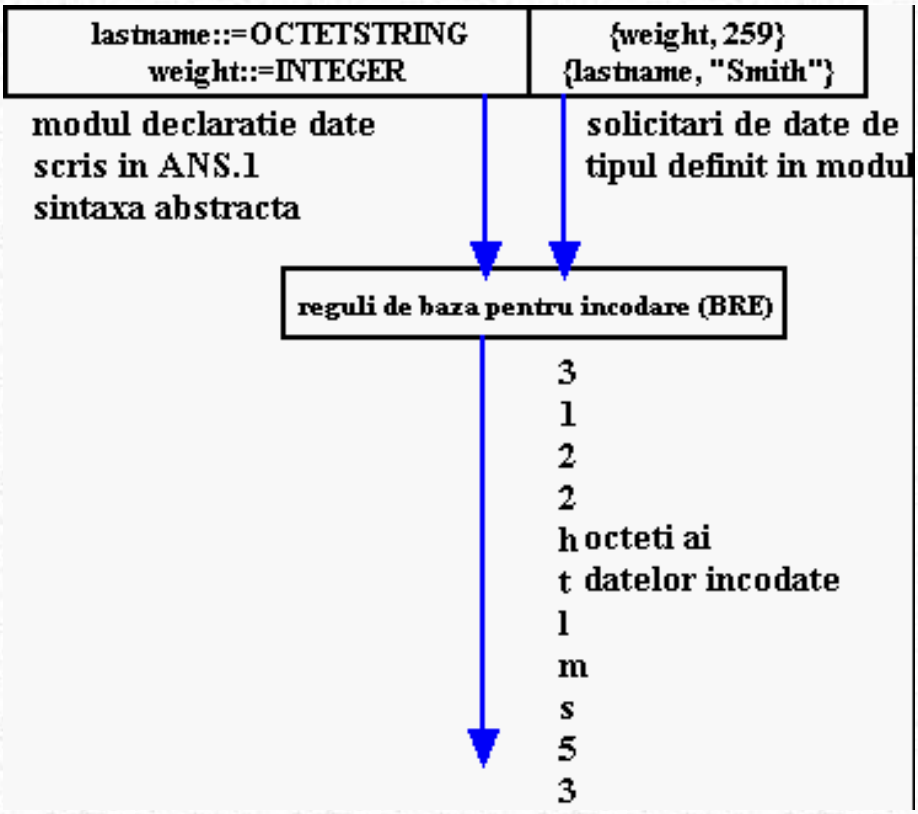
Standardul ISO

Sintaxa abstracta: "limbaj" pentru descrierea structurii de date

- este limbaj pentru descrierea datelor, nu limbaj de programare
- defineste tipuri universale de date
- permite tipuri de date definite pentru utilizator

Reguli de baza pentru incodare:

- converteste specificatia de sintaxa abstracta a structurii de date in serii de octeti (pentru transmisie)



ASN.1: Tipuri universale

Tipuri predefinite cu valori date ale tagurilor:

Tagul	Tip	Comanda
1	BOOLEAN	valoarea este adevarata sau falsa
2	INTREG	poate fi arbitrar de mare
3	sIR DE BIT	lista unui sau mai multor biti
4	sIR DE OCTET	lista unuia sau mai multor octeti
5	NUL	nici o valoare
6	IDENTIFICATOR DE OBIECT	se refera la un "obiect", de ex., numar de protocol
9	REAL	punct flotant

Exemplu: gandeste-te la ::= ca definind tipuri noi de date in termeni de tipuri universale de date

```
Married ::= BOOLEAN

SSN ::= INTEGER

Lname ::= OCTETSTRING

Salary ::= REAL

IPAddress ::= OCTETSTRING (SIZE 4)
```

Sintaxa ASN.1: constructori

ASN.1 defineste tipul de constructor pentru constructia tipurilor mai complexe de date din tipuri de date "mai simple":

Tag	Tip	Comentariu

16	SECVENta	lista ordonata, fiecare element este un tip ASN.1
17	SET	la fel ca la secventa, dar neordonat
11	ALEGERE	un tip luat din listele specificate

Exemplu de tip de date construite:

```
studentRecord ::= SEQUENCE {
Lname OCTETSTRING,
Fname OCTETSTRING,
Mname OCTETSTRING,
Married BOOLEAN DEFAULT FALSE,
SSN INTEGER
}
```

Sintaxa ASN.1: realizarea tagurilor si codarea

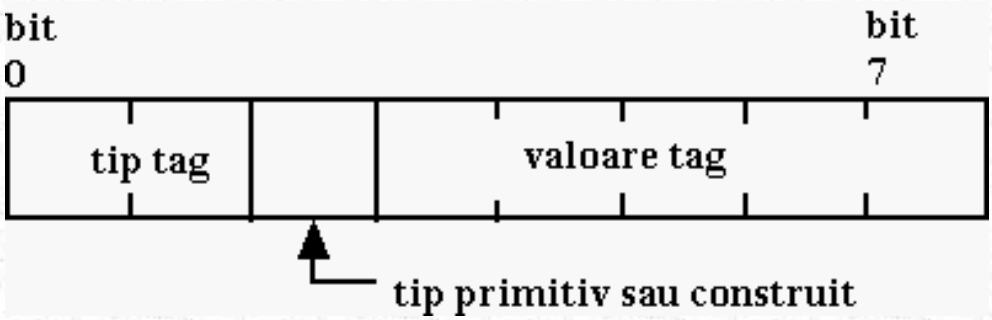
Datele ASN.*1* sunt ***autoidentificabile.***

- BER: Basic Encoding Rules (Reguli pentru incodarea de Baza) (PER: Packed Encoding Rules (Reguli pentru incodarea impachetata))
  - fiecare valoare transmisa este incodata folosind incodarea tag/lungime/valoare (TLV):

Tag	Lungime	Valoare

Tagul de 8-bit

- tipul tagului, 2 biti. (00 este UNIVERSAL)
- primitiva versus construita, 1 bit
- valoarea tagului (5 biti)



Lungimea de 8-bit: lungimea datelor incodate.

Valoarea: datele insasi incodate

- intregii sunt incodati in al 2-lea complement, de lungime arbitrara
- boolean, este incodat in un octet, 0 este FALS
- sir de octeti: transmite valorile octetului
- real: sunt posibile mai multe incodari.

ASN.1 Exemplu de incodare:

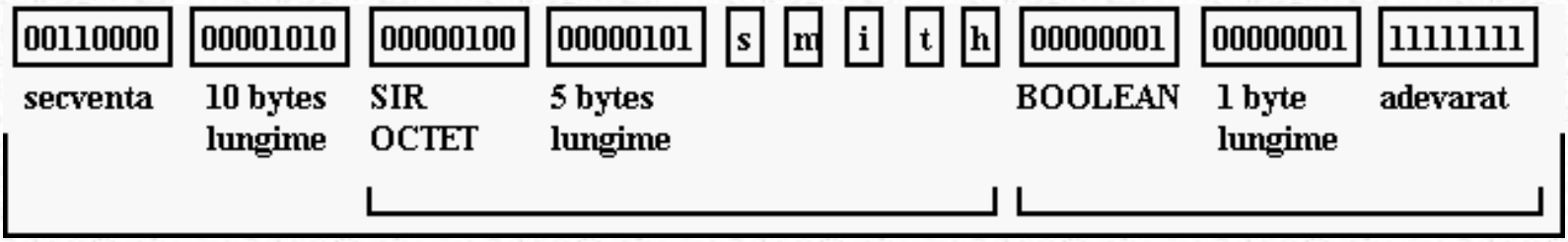
Definitia ASN.1:

Attendee ::= SEQUENCE {

name OCTET STRING,

paid BOOLEAN }

Data {"Smith",T} trebuie incodata:

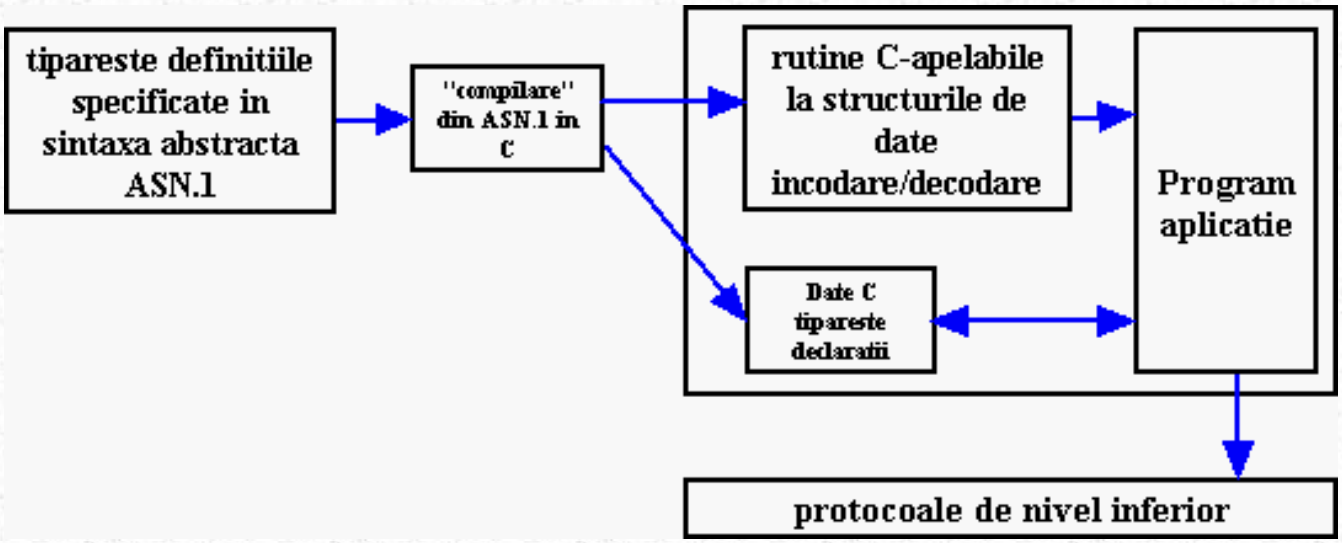


De notat structura de cuib a TLV in exemplul de mai sus.

ASN.1: Cum se foloseste?

"Compilerele" ASN.1 iau modulul de sintaxa abstracta a lui ASN.1 si produc

- definitii de tipuri de date C (de ex., typedef) pe care utilizatorul poate sa le includa pentru a crea structuri de date continand aceste tipuri
- bioblioteca de rutine C - apelabile (de ex., una pentru fiecare tip de date) pentru a incoda/decoda fiecare typedef in/din incodarea TLV.



Servicii de Prezentare: intentia de inchidere

Consumuri cu procesarea prezentarii:

- pana la 90% timp de procesare pe stiva ethernet/IP/TCP/prezentare
- costul de incodare a distributiei este de 5-20 ori mai mare decat copierea acesteia

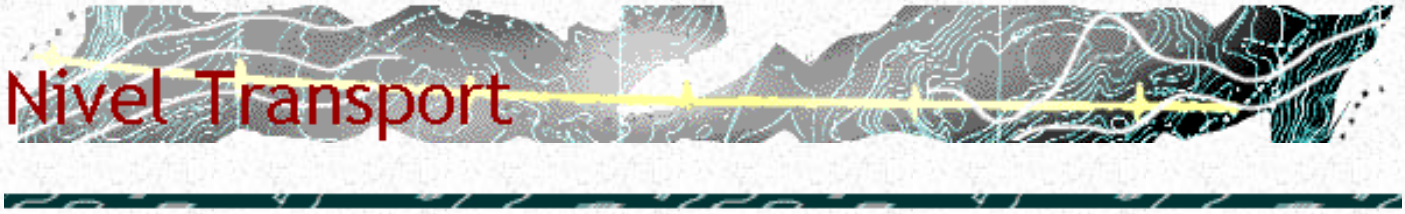
Alt "nivel" popular de prezentare este xdr al lui SUN (reprezenare de date externe)

- este similar ca si concept cu ASN.1

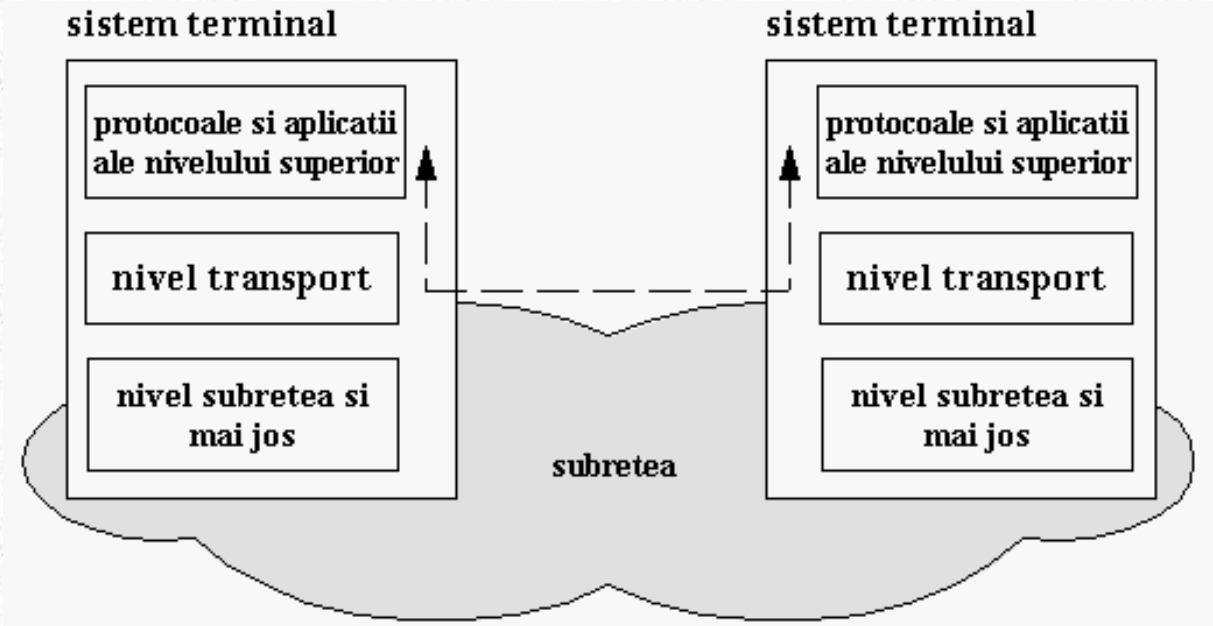
Referinte:

- John Larmouth's book "Understanding OSI" : chapter 8: ASN.1 , role of ASN.1 in next generation http
- Neufeld and Y. Yang, "An ASN.1 to C compiler," *IEEE Trans. Software Engineering*, Oct. 1990
- C. Huitema and A. Doghri, "Defining Faster Transfer Syntaxes for the OSI Presentation Protocol," *ACM Computer Communication Rev*. Oct. 1989

Back | Top | Next



5. Nivelul Transport



- introducere
- probleme fundamentale in retele
- comunicarea sigura prin un canal nesigur
- controlul congestiei si al traficului
- instalarea/dezinstalarea unei conexiuni
- multiplexarea si adresarea
- fragmentarea si reasamblarea
- calitatea serviciului
- probleme de implementare
- studii de cazuri: TCP, UDP, ATM, XTP

Aspecte interesante

Multe probleme fundamentale importante de retea apar aici:

- cum sa comunici in siguranta printr-un canal nesigur
- de ce/cum/cand trebuie restransa o transmisie: controlul traficului si al congestiei
- care sunt regulile de comunicare (si oprire) in cazul unor mesaje pierdute, intarziate sau reordonate
- cum sa lucrezi cu cantitati mari de date: fragmentarea si reasamblarea
- cum sa garantezi performanta in cazul unor resurse partajate statistic.

Oferta de servicii

Livrarea datelor intre doua gazde-terminal

Aspecte ale serviciului de transport:

- ***detectia erorii si recuperarea:*** erori (date pierdute sau corupte) detectate la destinatar. Corectarea erorilor detectate
- ***timing:*** timpul intre date de la expeditor prezervat in cazul livrarii la destinatar
- ***fragmentarea:*** prezervarea limitelor unitatii de date (de ex., "mesajul")

Modele comune pentru serviciul de transport:

- ***fara conexiune:*** datagrame, fara garantii, detectia optionala a erorilor, nu sunt recuperate datele eronate, fara timing
- ***orientate pe conexiune:*** recuperarea datelor eronate, fara timing
- ***asemanator circuitelor:*** prezervarea timingului, nu sunt recuperate datele eronate, detectia optionala a erorilor.

Modele de servicii Internet, OSI, ATM

arhitectura retea	serviciu	protocoale
Internet	orientat pe conexiune	TCP
	fara conexiune	UDP
OSI	orientat pe conexiune	TP0, TP1, TP2, TP3, TP4
	fara conexiune	CLTP
ATM	asemanator circuitelor	AAL1
	orientat pe conexiune	AAL3/4, AAL5, asigurat
	fara conexiune	AAL3/4, AAL4, neasigurat

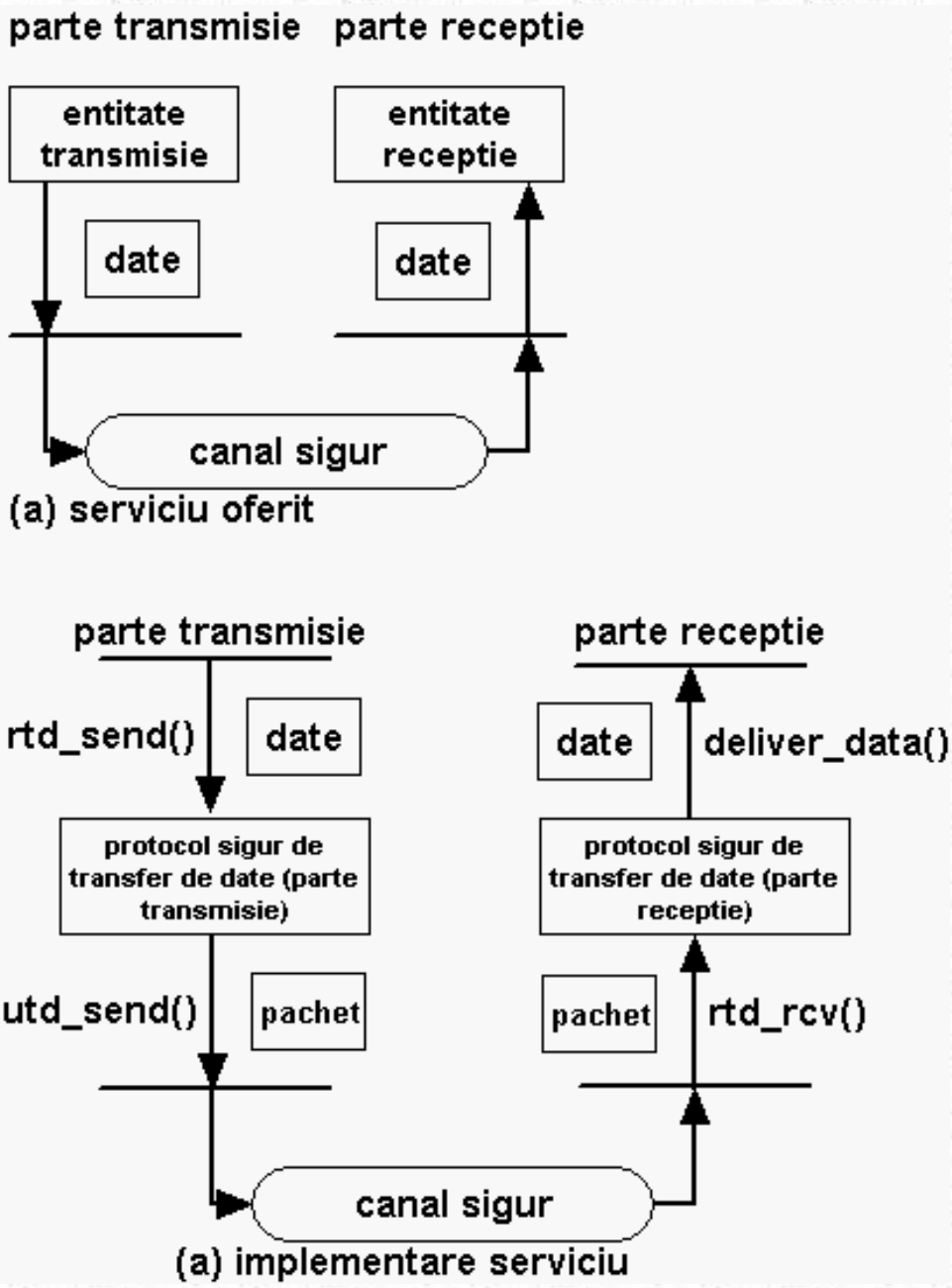
Nota: protocoale multiple pentru acelasi serviciu:

- ATM:
- OSI:

Comunicatii sigure printr-un canal nesigur

Scop: proiectarea unui protocol al transferului de date astfel incat:

- sa existe livrari sigure de date intre aplicatiile/protocoalele nivelului superior
- sa se utilizeze un nivel de retea care este "nesigur"



Interactia cu nivelele superior si inferior

Nivelul superior al expeditorului: nivelul de transport invocat mai sus prin apelarea la **`rdt_send(data)`**

- **`rdt:`** transfer sigur de date
- **`data:`** de livrat la nivelul superior al destinatarului

Nivelul superior al destinatarului: nivelul de transport livreaza **`date`** catre nivelul superior prin apelarea **`deliver_data(data)`**

Nivelul inferior al expeditorului: apelarea la **`udt_send(packet)`** va trece **`pachetul`** in nivelul inferior

- **`udt:`** transfer nesigur de date

Nivelul inferior al destinatarului: livreaza **`pachetul`** catre nivelul de transport prin apelarea la **`rdt_rcv(packet)`**

Note:

- **`data`** este unitatea de date care traverseaza limitele

- **packet** este unitatea de date care traverseaza limita inferioara
- **packet** = **data** in cazul catorva campuri suplimentare.

Transfer sigur de date: aspecte de mediu

Presupuneri privind serviciul pentru nivelul retea:

- mediul de baza (retea) care conecteaza expeditorul si destinatarul poate avea multe legaturi, rutere, retele!

Un prim set de presupuneri despre mediu:

- nu exista pierderi, coruperi, sau reordonari

O prima incercare pentru un protocol (rdt1.0)

```
rdt_send(data)
{
make_packet(packet,data);
udt_send(packet);
}

rdt_rcv(packet)
{
extract(packet,data);
deliver_data(data);
}
```

Cum se specifica un protocol?

Cum se descrie/specifica un protocol?

- in engleza
- limbaj de programare sau pseudocod
- metode grafice: modelele Petri net si masini cu stare finita.

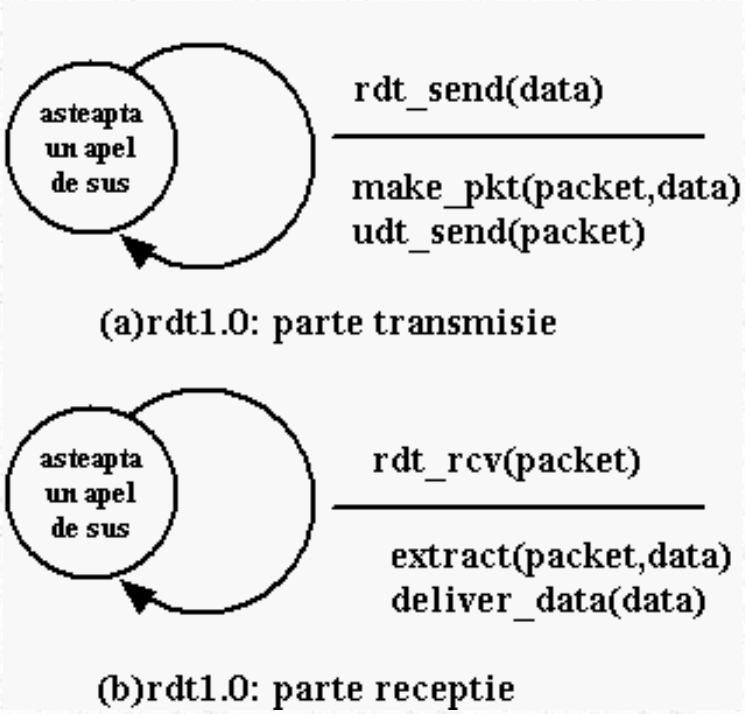
Masina cu stare finita (finite state machine (FSM)) consta din:

- ***set de stari*** pentru fiecare entitate de protocol
  - fiecare entitate are *propriul* sau set de stari
  - starea "inregistreaza" intreaga istorie trecuta relevanta a entitatii
  - raspunsul entitatii la fiecare "eveniment" din stare trebuie sa fie unic definit
- ***set de arce etichetate directate*** intre stari
  - reprezinta schimbarile in stari
  - etichetarea arcului:

evenimentul sau actiunea care determina tranzitia

actiunea luata la tranzitie

FSM pentru rdt1.0



Un al doilea set de presupuneri despre mediu

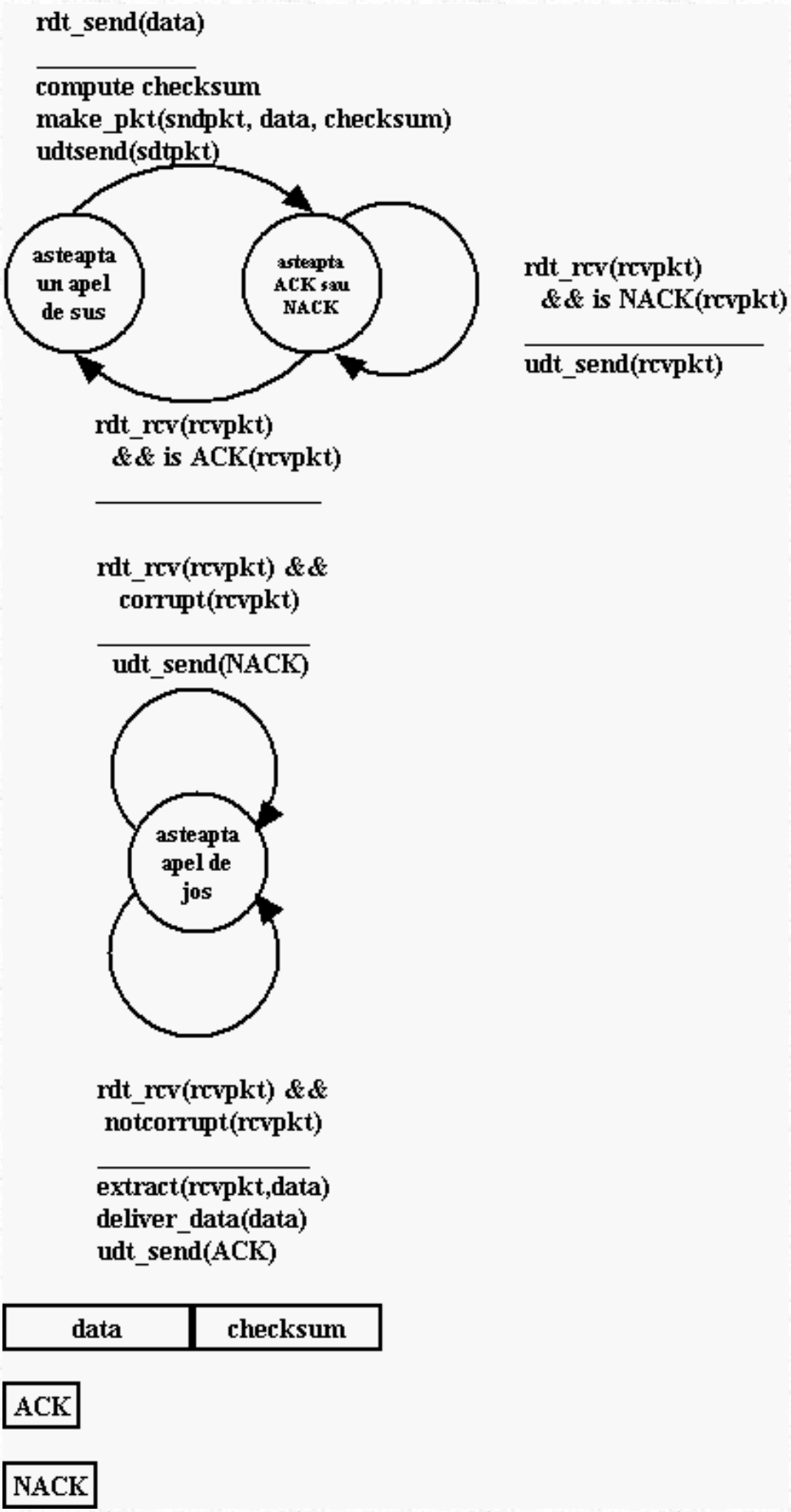
Pachetele trimise in retea pot fi corupte dar nu pierdute

- orice parte a pachetului poate fi corupta

`corrupt(P), notcorrupt(P)` returneaza T daca pachetul este (nu este) corupt

in acest caz rdt1.0 nu mai este bun pentru noile presupuneri despre mediu, fiind necesar un nou mecanism de protocol (rdt2.0)

Protocolul rdt2.0

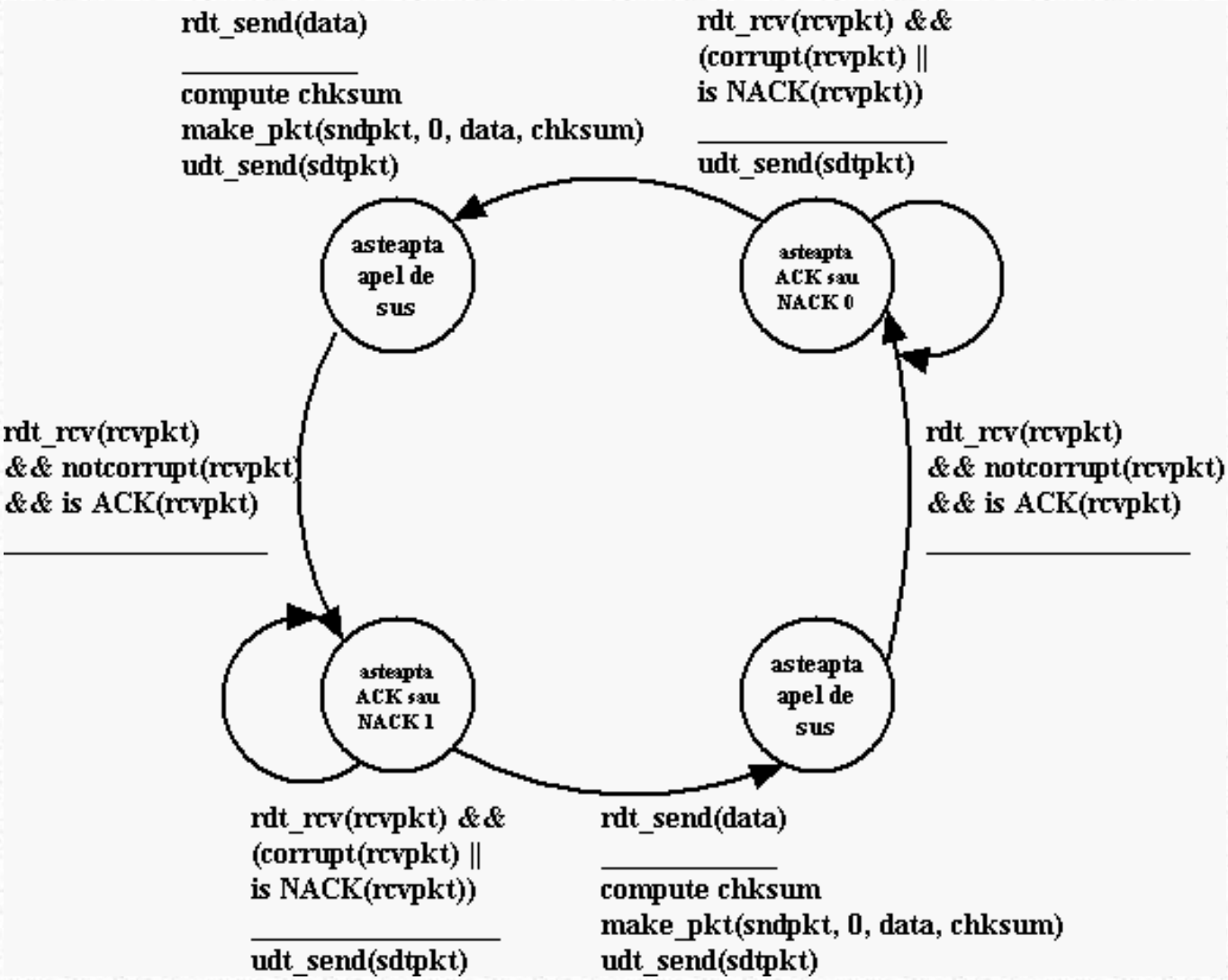


intrucat rdt2.0 nu merge intotdeauna cu presupunerile considerate, a aparut un nou protocol.

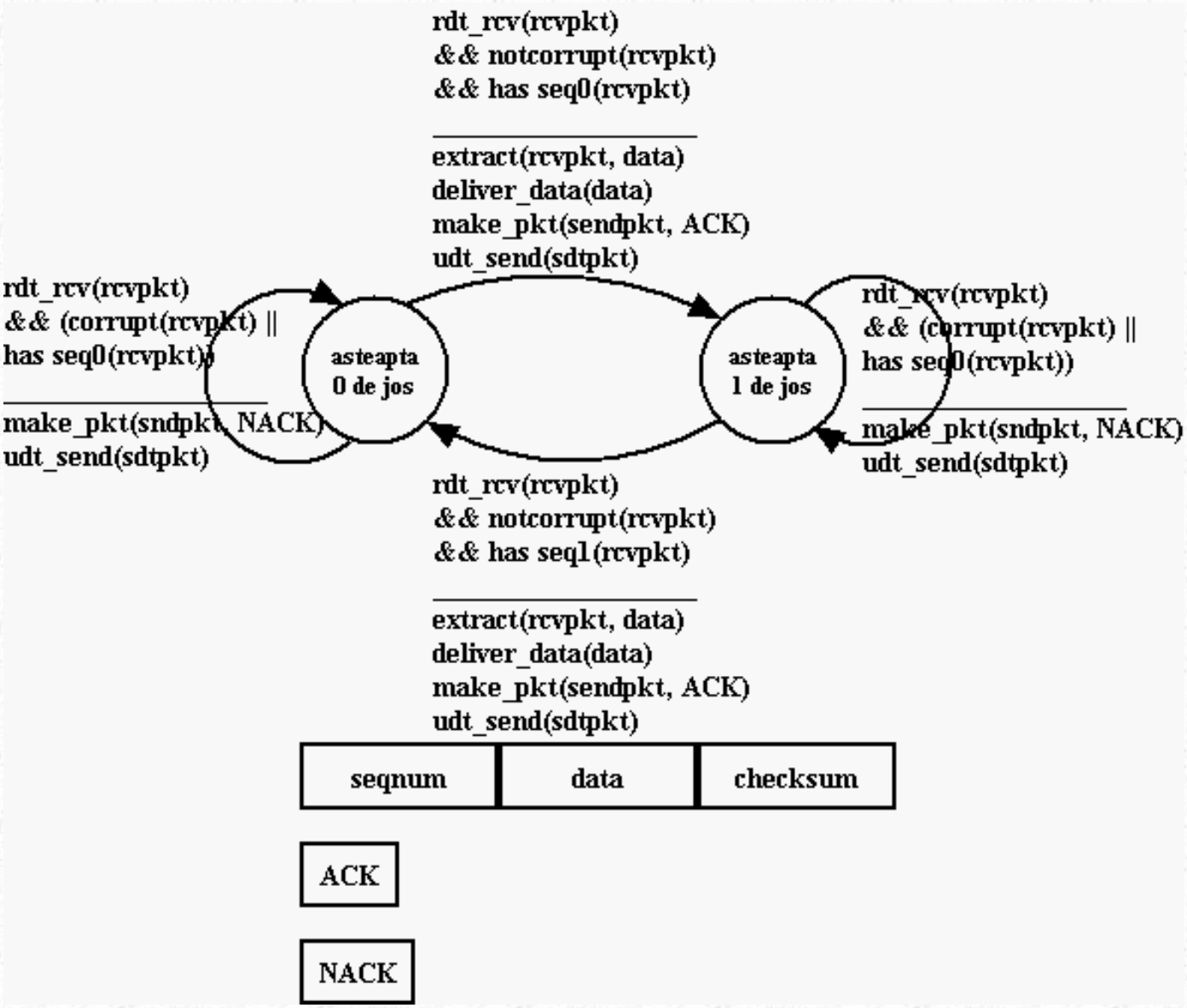
Protocolul rdt2.1

Corecteaza unele probleme ale protocolului rdt2.0.

Protocolul rdt2.1: expeditor



Protocolul rdt2.1: destinatar

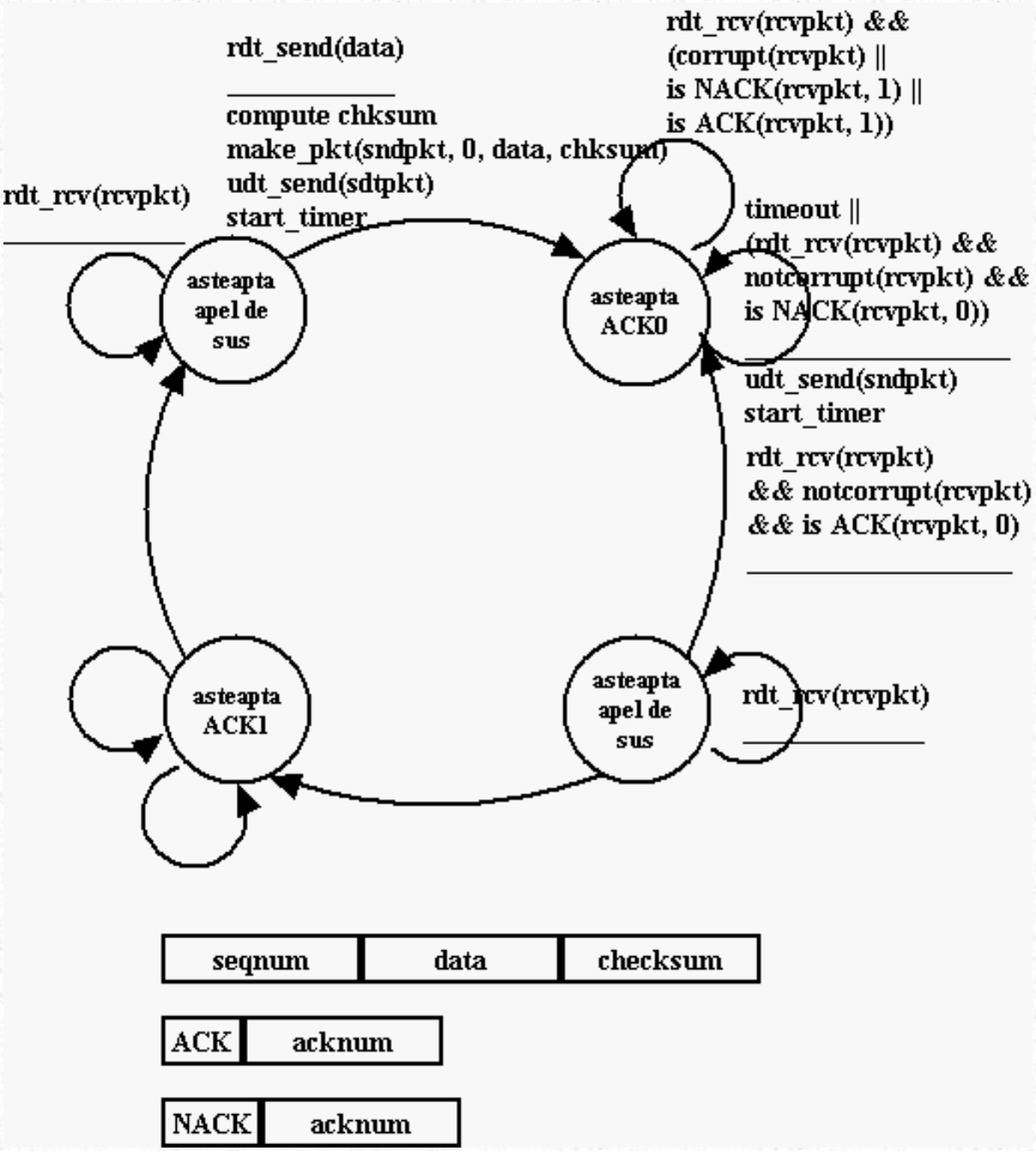


Un al treilea set de presupuneri despre mediu

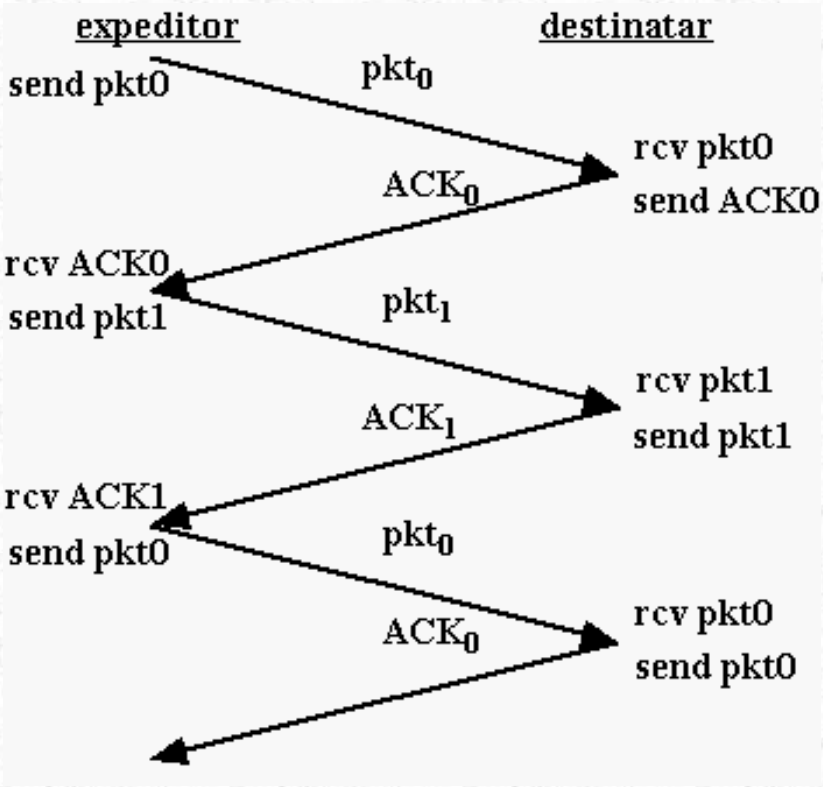
Pachetele trimise catre nivelul retea pot fi pierdute (si de asemenea corupte)

Sunt necesare noi mecanisme de protocol (rdt3.0):

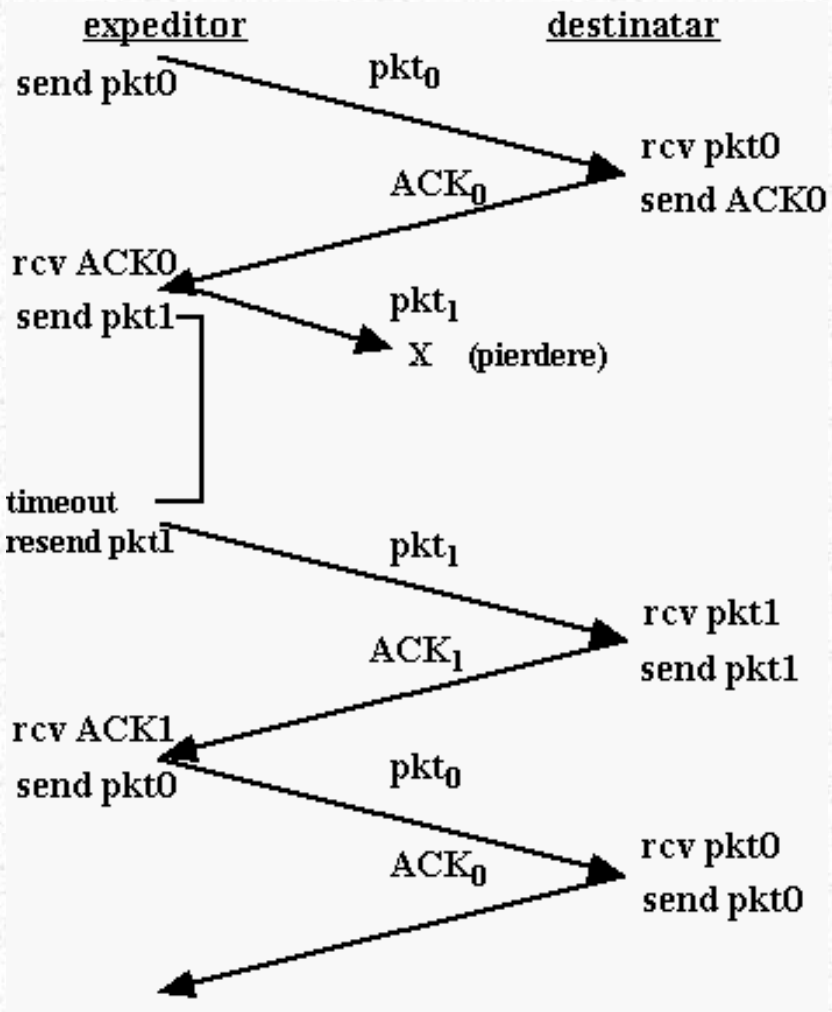
Protocolul rdt3.0: expeditor



Operatii ale rdt3.0: fara erori



Operatii ale rdt3.0: pachete pierdute



Protocolul de bit alternant

Protocolul rdt3.0 este cunoscut ca protocolul de Bit Alternant (Alternating Bit (AB))

Opreste si asteapta ARQ

- ARQ: repetare/solicitare automata

Comunicare asemanatoare cu cea semi-duplex

Utilizarea

- timere
- numere secventiale
- biti de detectie a erorilor

Necesara pentru a oferi comunicare sigura in prezenta unor pachete pierdute sau corupte.

Protocoale de recuperare a datelor eronate tip pipeline

intarzieri cu un larg spectru: propagarea intarzie lungimea in functie de timpul de transmisie al pachetului

De ex.: pentru legatura de 1 Gbit/sec si pachet de 1Koctet sunt necesare 8 microsecunde pentru a transmite prin fir.

- t_trans = (8Kbit/pachet)/(10**9 bit/sec) - 8 microsecunde

Utilizarea canalului: fractiune din timpul de expediere (canal la expeditor) este ocupat cu transmisia

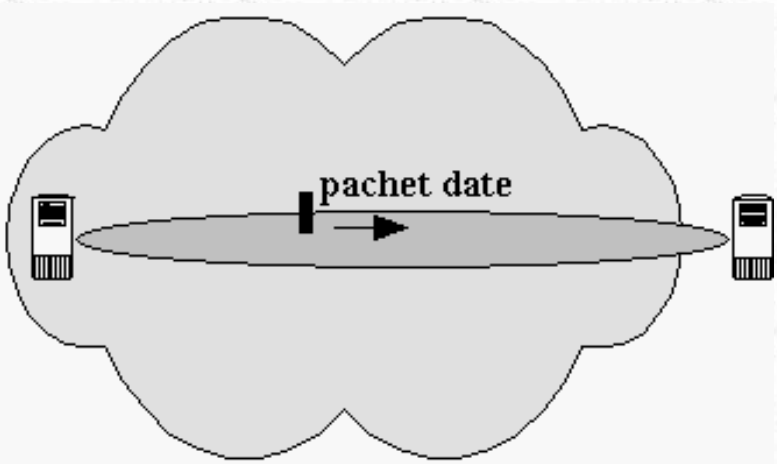
- U_expeditor = 0.008msec/30.016msec = 0.000266
- expeditorul este ocupat numai 0,02% din timp!
- Viteza de transmisie a expeditorului este de 266Kbit/sec chiar cu o legatura de 1G.

Protocolul (nu capacitatea canalului) restrange performanta.

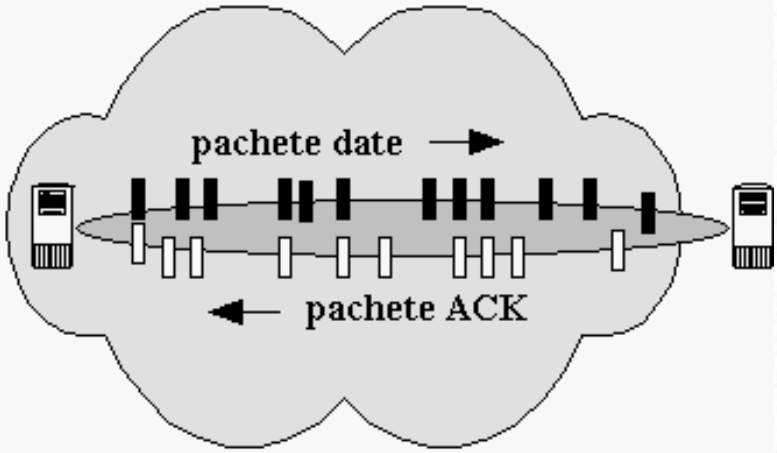
Protocoale tip pipeline de recuperare a erorilor

Solutia la transmisia lenta cu intarziere mare in propagare: pipeline.

Permite tranzitarea intre expeditor si destinatar a pachetelor recunoscute multiplu.



(a) un protocol opreste-si-asteapta in operare



(b) un protocol pipeline in operare

Go-Back-N ARQ

Pachete transmise continuu (cand este posibil) fara a astepta pentru recunoastere, pachete nerecunoscute.

O asciere temporara diferita din punct de vedere logic pentru expeditor cu fiecare pachet nerecunoscut: extensie a protocolului AB.

Destinatar:

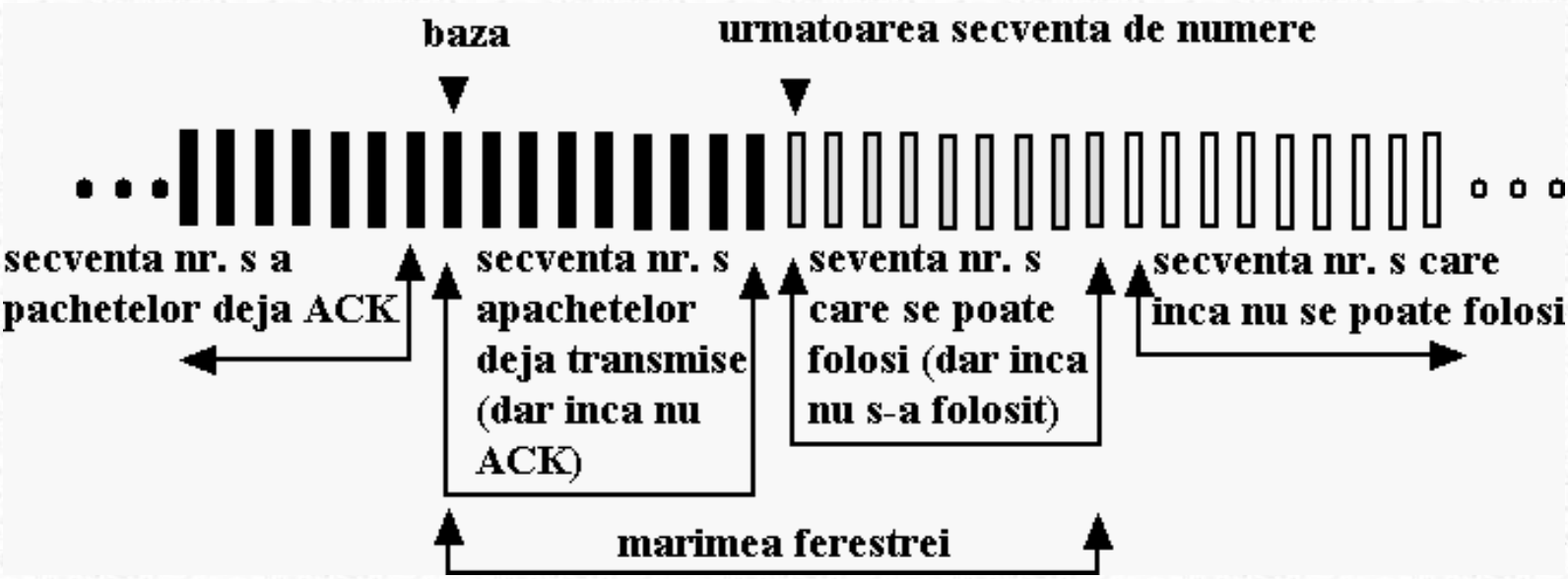
- Pachetul recunoscut, daca este primit corect si in ordine, trece catre nivelul superior
- Pachet nerecunoscut sau ignorat, corupt sau care nu este in ordine

Expeditor:

- Daca pachetul n primit este nerecunoscut sau a trecut timpul, incepe retransmisia de la n din nou
- Recunoasterea cumulativa: recunoasterea lui n presupune implicit recunoasterea tuturor pana la n.

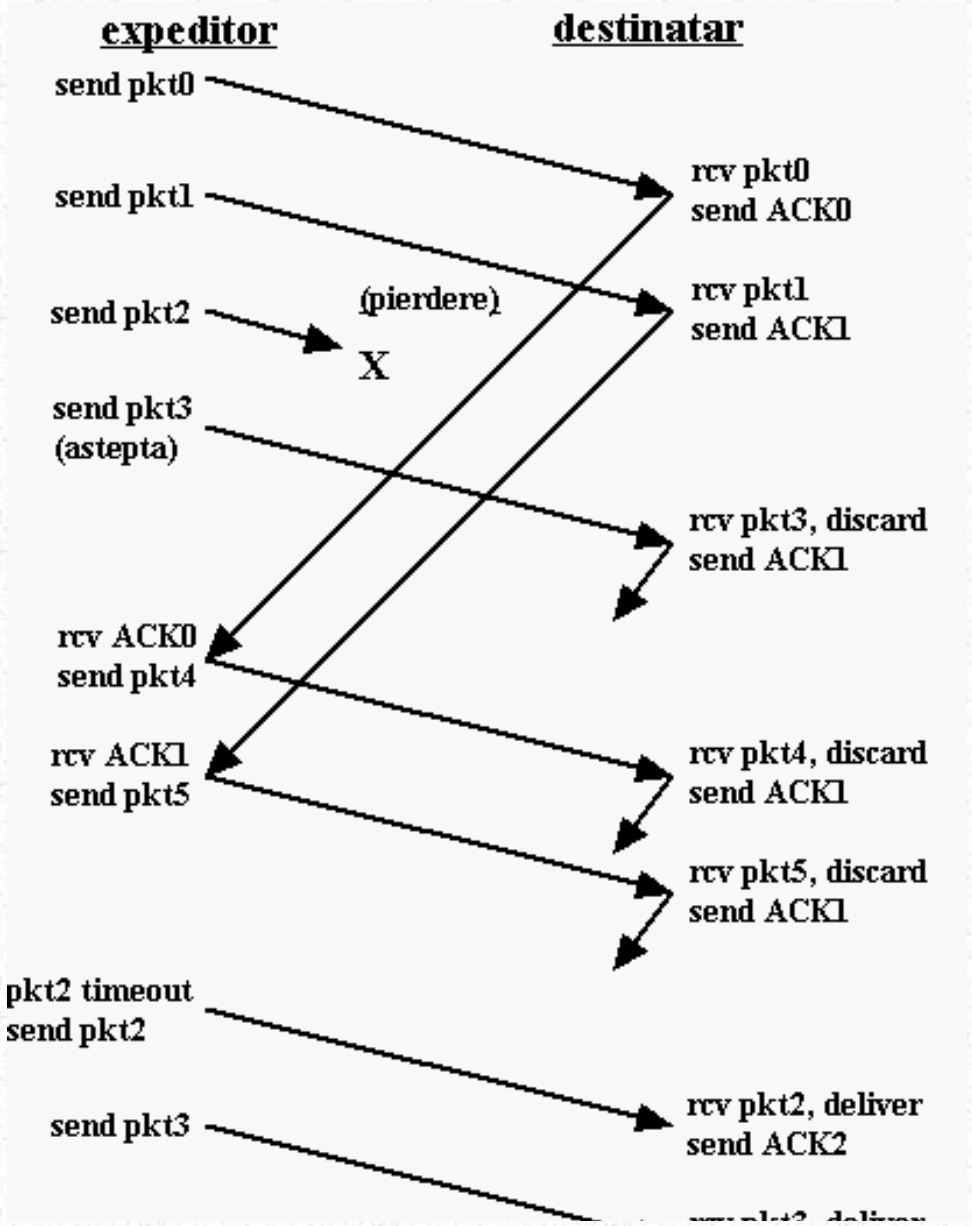
Niciun transport receptionat nu este buferat prin inlaturare. Resursele sunt salvate la destinatar. Se elimina astfel posibilitatea livrarii simultane a unor mari cantitati de pachete catre nivelele superioare.

Sunt necesare o buferare si o procesare a protocolului cat se poate de simple, atat la expeditor cat si la destinatar.



Este necesara o optimizare intre complexitatea buferarii/procesarii si largimea de banda.

Go Back N: Exemplu



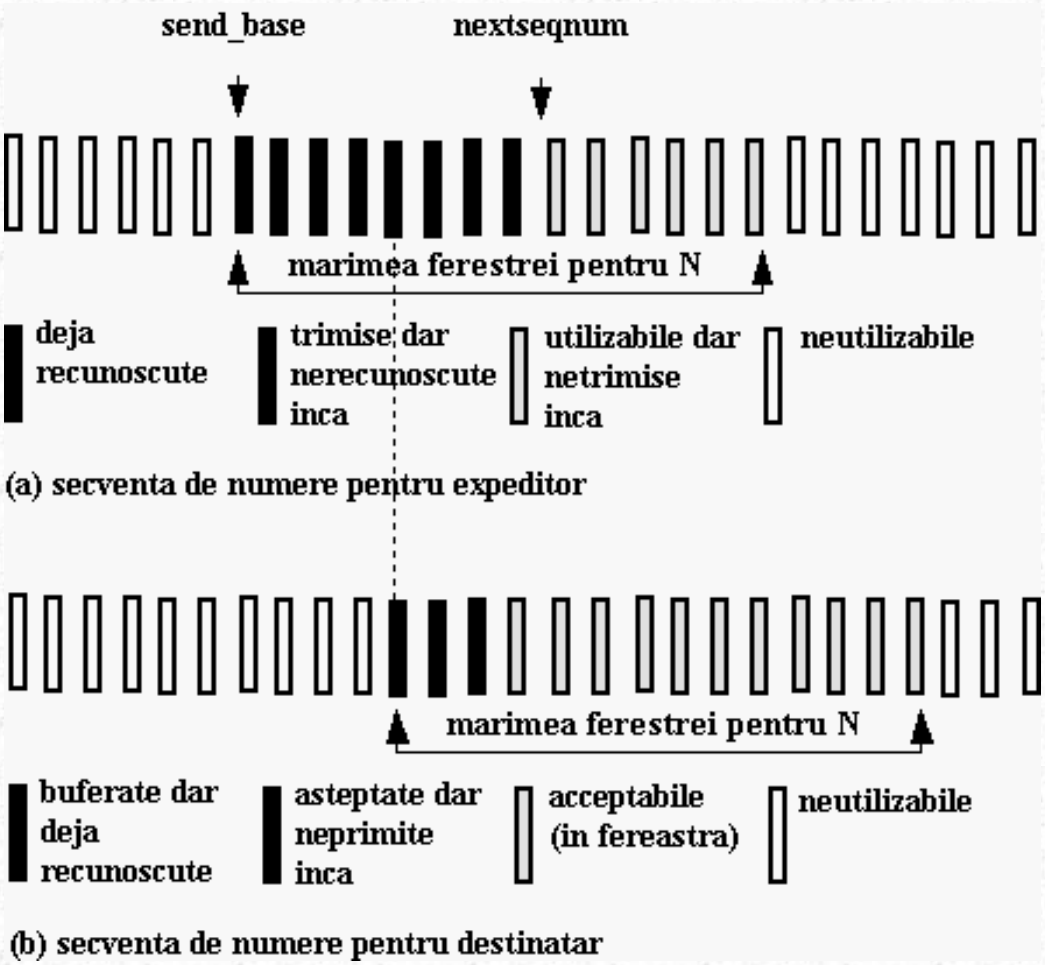
ARQ cu repetare selectiva

La fel ca in Go back-N:

- pachetele sunt transmise cand este posibil, pana la limita
- perioada asociata cu fiecare pachet nerecunoscut
- destinatarul nu recunoaste sau ignora pachetele corupte.

Fata de Go-Back-N

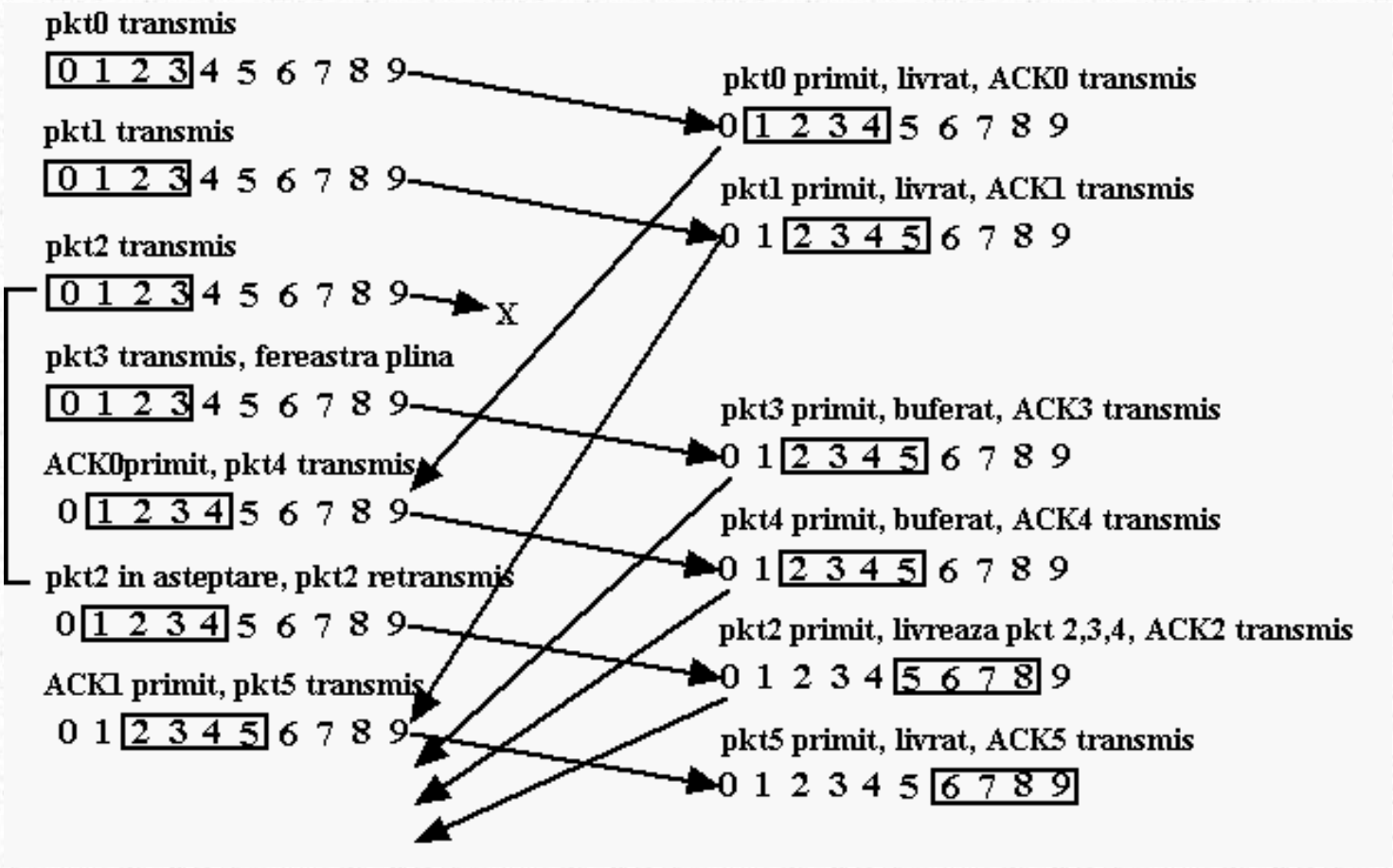
- pachetele care nu sunt in ordine dar sunt corecte sunt recunoscute
- destinatarul: bufereaza pachetele care nu sunt in ordine
- expeditorul: in cazul asteptarii sau la nerecunoasterea pachetului n, doar retransmite n



Nota:

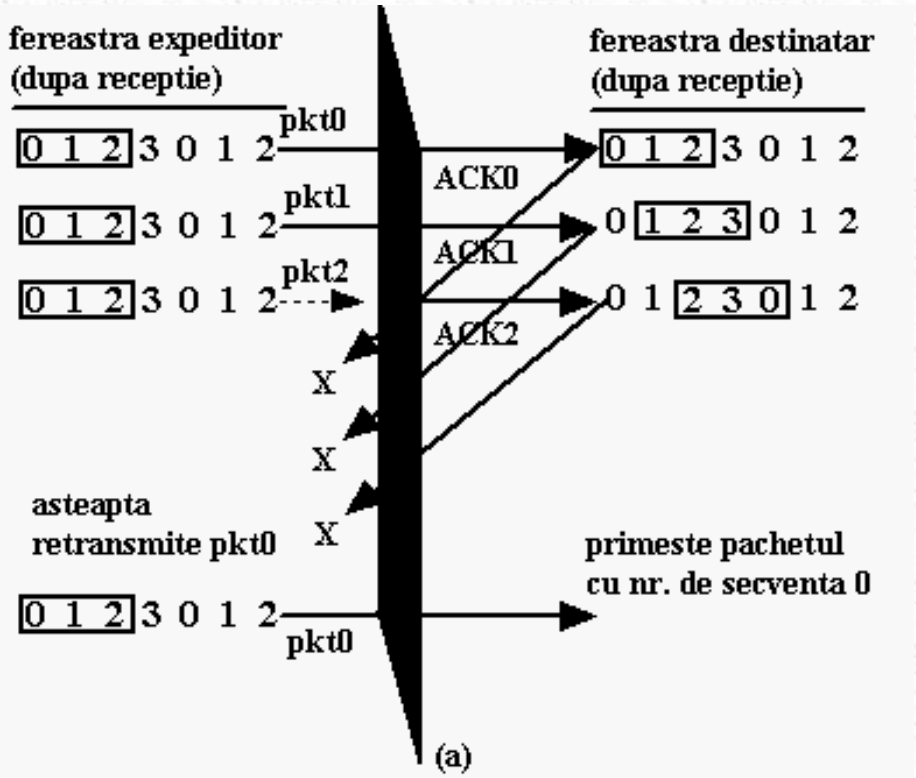
- mai multe pachete buferate la destinatar decat in cazul Go back-N
- un management de buferare mai complicat de ambele parti
- pentru largimea de banda nu este necesar sa se retransmita corect pachetele receptionate.

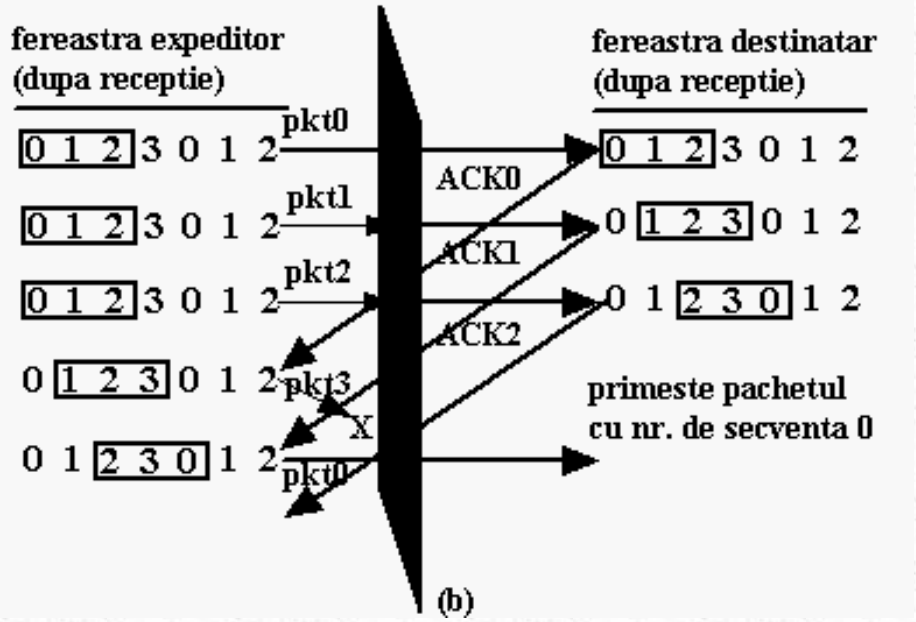
ARQ cu repetare selectiva: exemplu



Cat de mare poate fi o fereastra?

Sa presupunem ca dimensiunea spatiului de numere secventiale este N





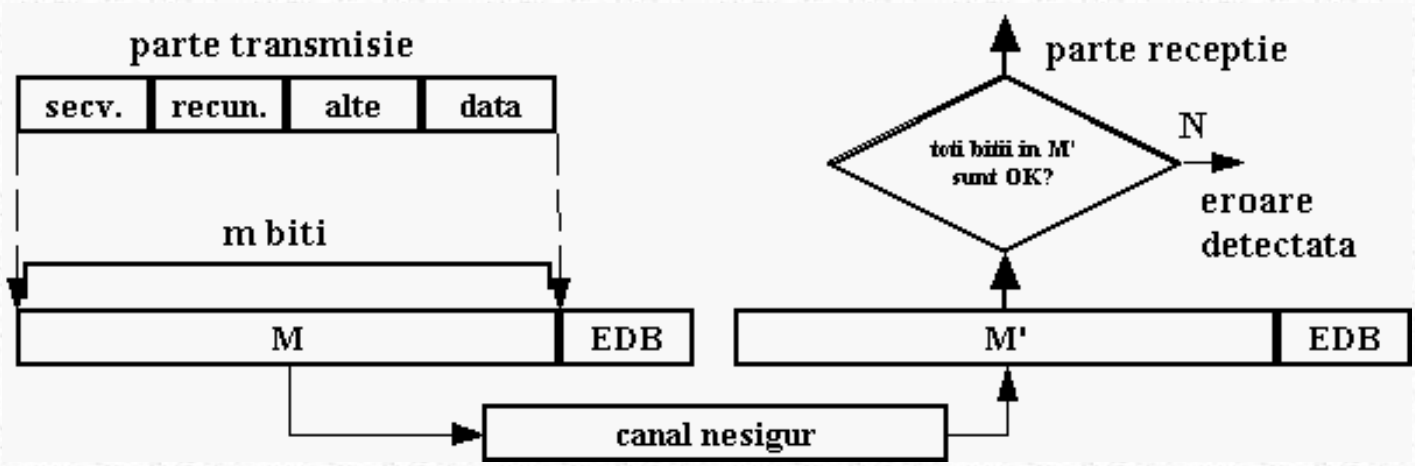
Problema fundamentala: expeditorul si destinatarul

- nu au informatii sincronizate
- nu cunosc exact aceleasi informatii.

Erori de detectie: checksum

Trebuiesc detectate erorile: bitii in pachet pot fi deranjati in timpul transportului sau al stocarii in zone intermediare.

Solutie: se adauga biti suplimentari in pachete care ne vor permite sa detectam (si posibil sa corectam) erorile de bit.



Exemplu simplu: paritatea

- Fiind dat pachetul cu n-1 bit, adaugam bitul al n-lea, alegand valoarea astfel incat numarul total de biti 1 din pachet (inclusiv al n-lea bit) este par (paritate egala).

Exemplu de pachet:

secventa	recunoastere	data	bit de detectie a erorii (paritate)
0111	0001	10101011	0

La destinatar:

- numarar # 1din packet, daca este impar, atunci este eroare!

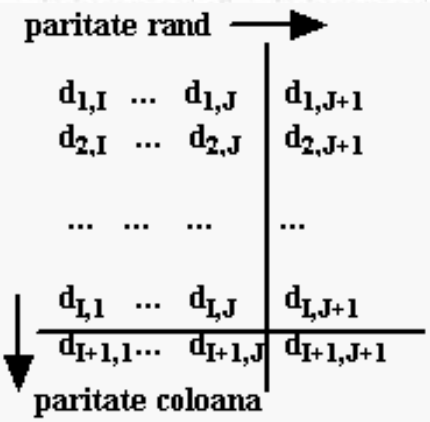
Nota:

- Exista multe coduri cu posibiltati de detectie a erorilor mult mai puternice
- Headerul insusi al pachetului este adesea vericat separat prin checksum (verificarea sumei)
- Verificarea sumei se face de asemenea si la nivelul legaturilor de date
- Suportul hardware pentru verificarea sumei la nivelul transport: SGI

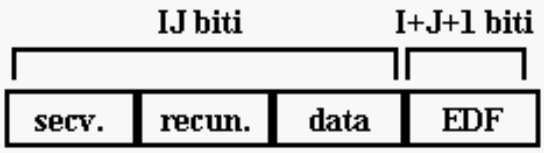
Corectia eroarei de directionare (Forward Error Correction): FEC

Protocoalele ARQ opereaza prin detectarea erorilor si retransmitere

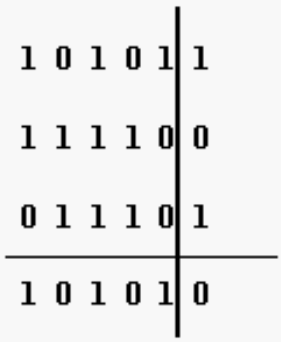
- Este necesara o intarziere in bucla a retransmisiei pentru a fi recuperata
- Poate fi prea mare pentru aplicatii in timp real complexe, de mare viteza
- FEC: ideea de baza este sa se retransmita date suficient de redundante astfel incat sa permita destinatarului sa le recupereze chiar din erori! (nu este necesara o transmisie a expeditorului)



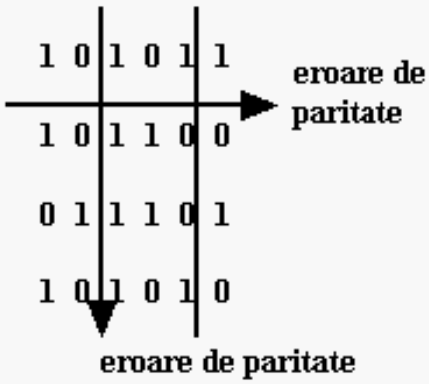
(a) paritate bidimensionala



(b) format pachet



(c) exemplu: fara eroare



(d) exemplu: eroare de un bit

Transferul datelor. Studiu de caz: TCP

Go-Back-N ARQ

- numarul secvential de 32 bit indica numarul de octet in trafic
- transfera un flux octet, blocuri utilizator cu dimensiunea nefixata
- transfer de date full duplex (bidirectional)
- transmite datele de nivel superior atunci cand acestea "doresc" (RFC793), incercand sa acumuleze 512 octets de date

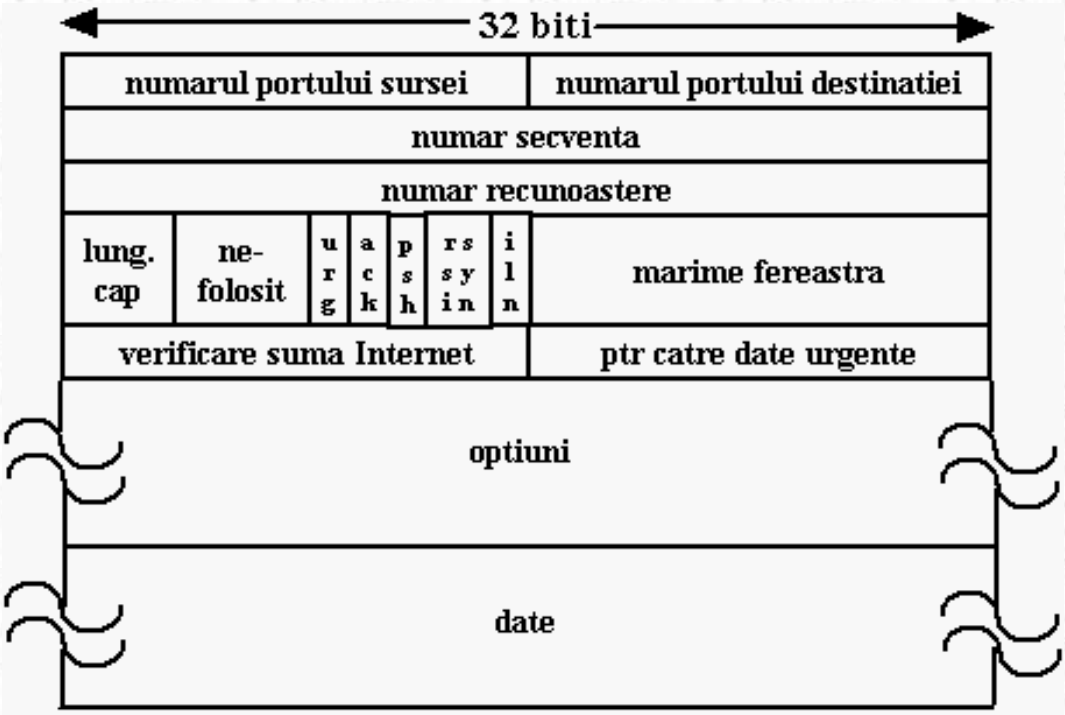
Recunoastere cumulativa: recunoasterea lui n implica recunoasterea tuturor octetilor pana la n

- recunoasterea pentru datele primite de la A la B influenteaza pachetul de date de la B la A

Checksum pentru Internet: adauga date, ia complementul 1

- acopera atat datele cat si headerul

Formatul de pachet TCP



Transferul de date: XTP

XTP: Xpress transfer protocol (protocol de transfer Xpress)

Desemnat pentru viteze mari, retele de inalta performanta.

Numere cu secventa 32 bit cu tranzitie la numere cu secventa 64 bit

Campul prioritar 32 bit pentru diferite date prioritare

Transfer de date sigure sau nesigure selectabile pentru utilizator

Go-Back-N ARQ dar destinatarul poate de asemenea indica *dimensiunea* pachetelor primite

- expeditorul doar retransmite lipsurile

Checksum:

- forma de paritate bidimensionala
- headerul si datele verificate separat
- verificarea datelor poate fi dezactivata
- verificarea la final (transmite date in timp ce se calculeaza checksum

Controlul traficului si al congestiei

Uneori expeditorul nu trebuie sa transmita un pachet gata:

- destinatarul nu este pregatit (de ex., buferele sunt pline)
- reactie la congestie
  - multe pachete nerecunoscute pot insemna intarzieri mari intre capete, retea congestionata
  - reteaua insasi poate oferi expeditorului indicatii asupra congestiei
- eliminarea congestiei:
  - expeditorul transmite uniform, pentru a evita supraincarcarile temporare ale retelei

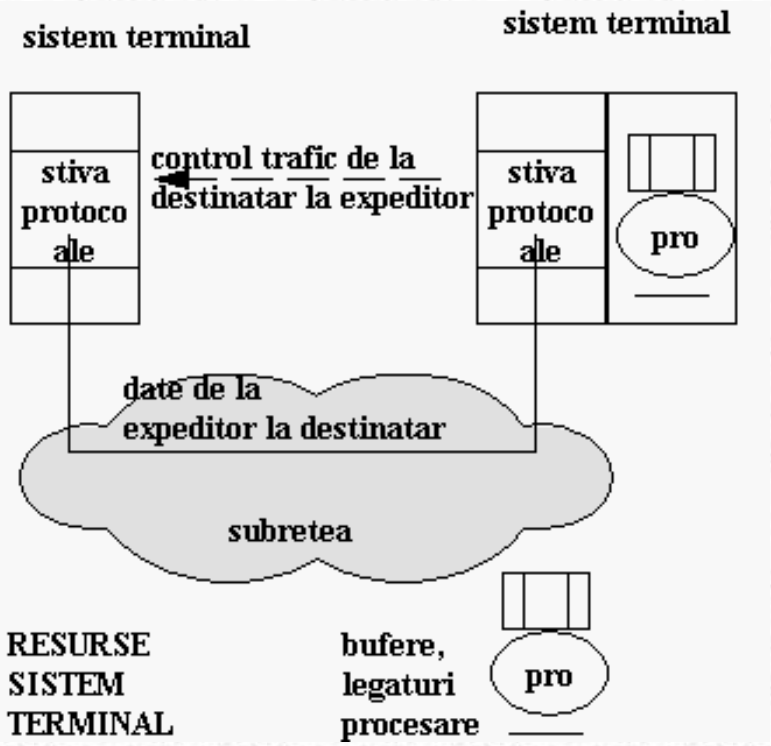
Controlul traficului: fituirea vitezei si a resurselor intre expeditor si destinatar

- expeditorul nu trebuie sa il suprasolicite pe destinatar

Controlul congestiei: actiune luata ca raspuns la intarzierea si (in consecinta) congestia in retea.

- sufocarea expeditorului este numai una din solutii

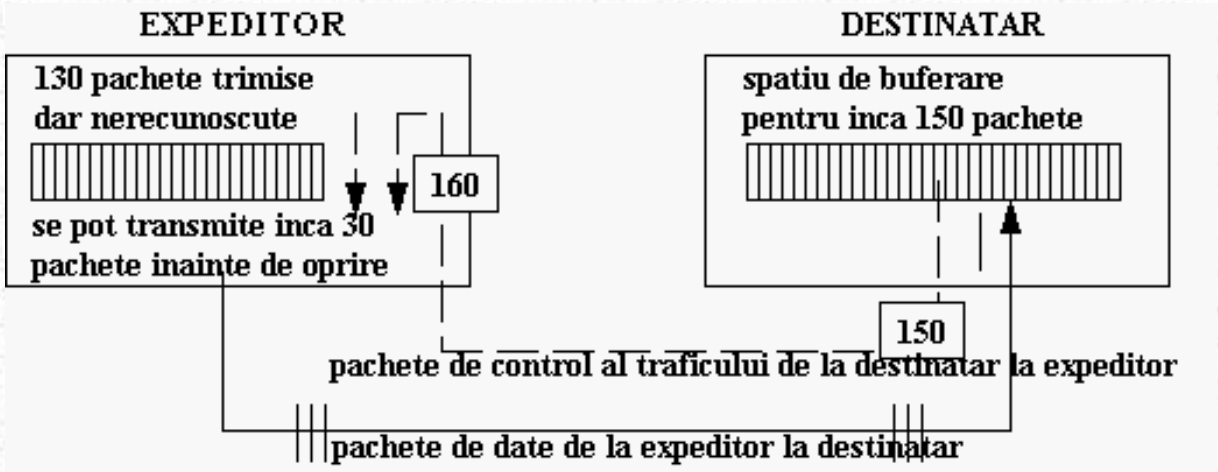
Scenariul controlului traficului



Doua situatii privind controlul traficului

Controlul explicit al traficului

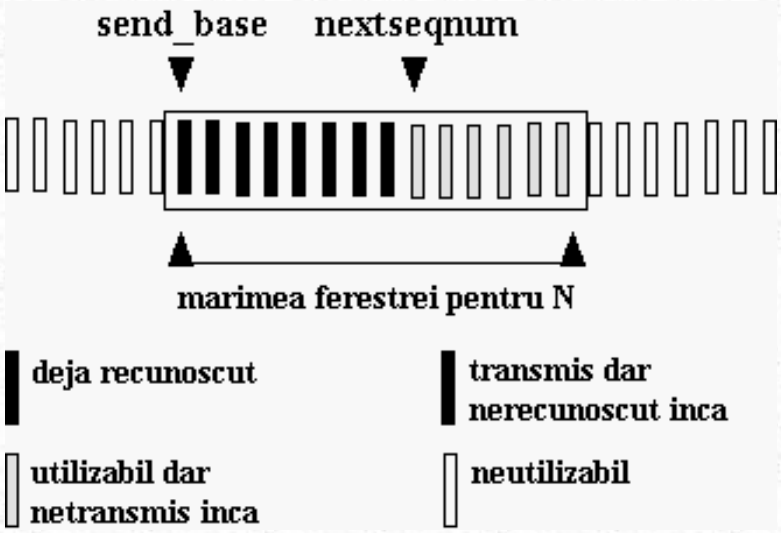
- Destinatarul spune expeditorului cat de mult sa transmita



Abstractizare utila: expeditorul mentine ***fereastra de trecere*** peste numerele din secventa, indicand ce poate fi transmis

- utilizat in TCP si TP4

- fereastra de control al congestiei (considerata opusa traficului) poate restrictiona in continuare expeditorul

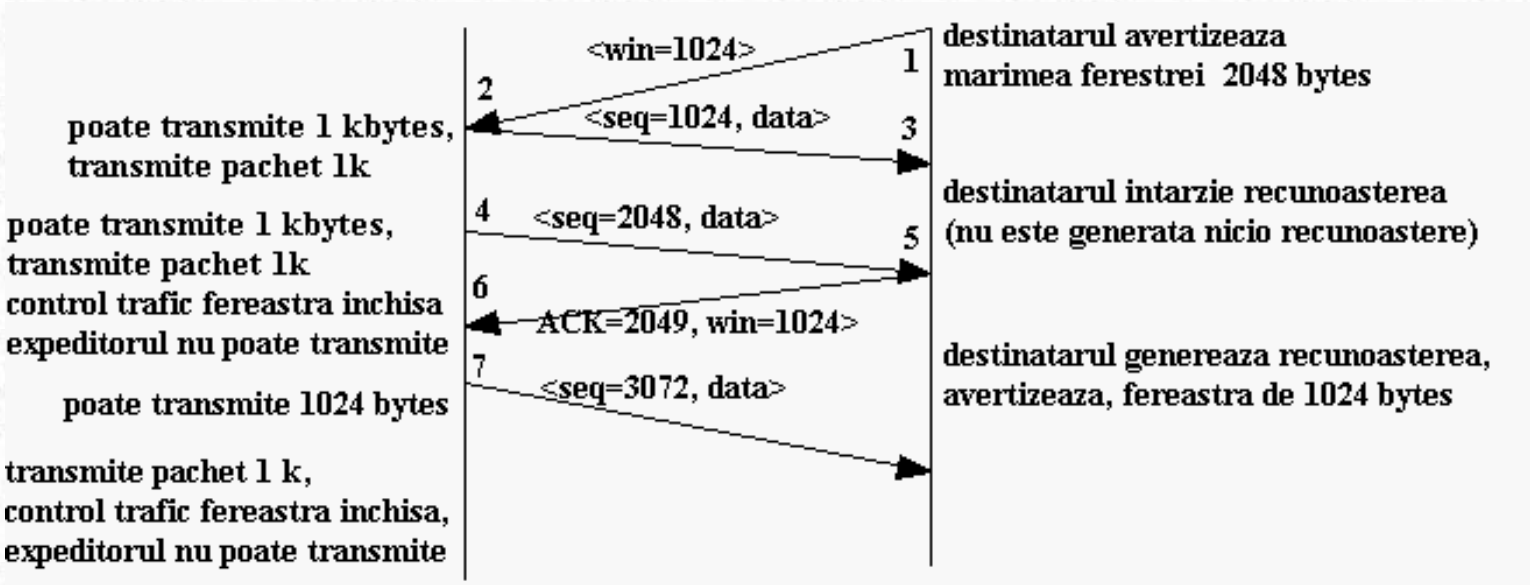


-

Controlul traficului in TCP

Destinatarul avertizeaza in mod explicit despre spatiul de buferare disponibil pentru expeditor

- "fereastra de avertizare" de 16-bit specifica numarul de octeti (incepand cu ACKnum) pe care expeditorul trebuie sa il primeasca
- dimensiunea maxima a ferestrei este de 64K
- in prezent se cauta optiuni pentru ferestre mai mari



A doua situatie de control al traficului: controlul ***implicit***

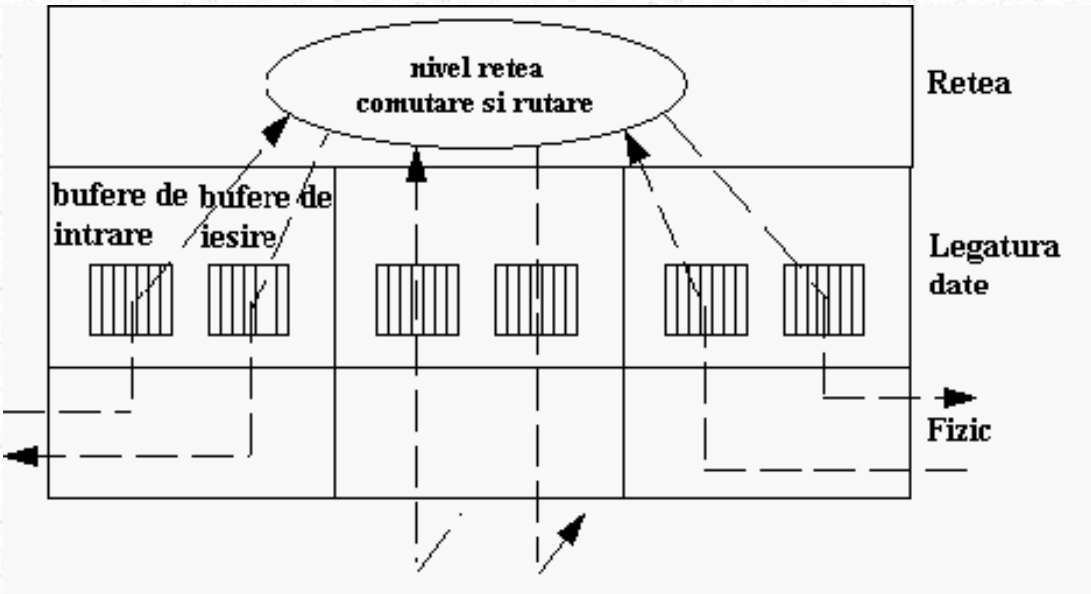
- recunoasterile intarziate (indiferent de motiv) incetinesc expeditorul

- traseul virtual al IBM:
  - initial transmite fereastra de N pachete
  - recunoasterea primului pachet in aceasta fereastra permite expeditorului sa transmita inca N pachete
  - fereastra arbitrara
  - numarul maxim de pachete nerecunoscute reduce traficul de pachete recunoscute

Controlul congestiei

Solicitarea temporara pentru partajarea resurselor (legaturi, procese, bufere) in ***nivelul retea si cele inferioare*** poate depasi oferta:

- pachetele sunt buferate pana cand resursele devin disponibile
- in cazul buferelor pline, se pierde pachetul (se renunta la el)
- multe buferari implica intarzieri excesive.



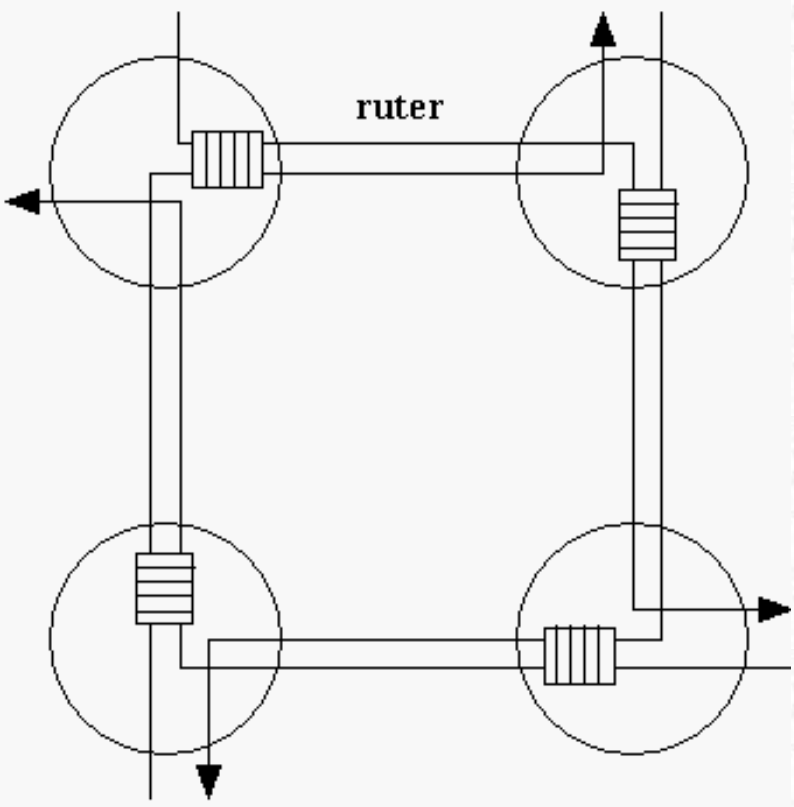
Controlul congestiei: efecte de retransmisie

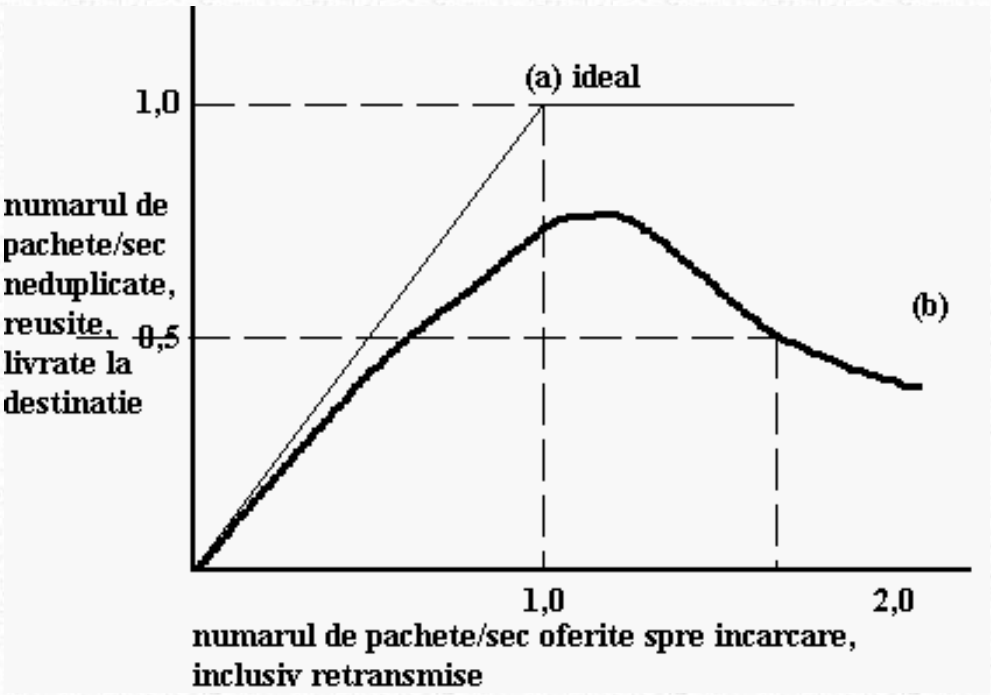
Cazul ideal:

- fiecare pachet livrat cu succes pana cand subreteaua atinge capacitatea necesara
- daca este sub capacitatea necesara, se livreaza pachetele adaptate la actuala capacitate

in cazul pierderilor sau a intarzierilor intre capete, retransmisia poate determina o inrautatire a situatiei.

- injecteaza mai mult (nu mai putin) trafic in retea





in mod real:

- Cand oferta de incarcare creste, se pierd mai multe pachete, determinand mai multe retransmisii, rezultand un trafic mai mare, si deci mai multe pierderi
  - in cazul (b), fiecare pachet original este transmis de patru ori in medie
  - scaderea vitezei de transmisie (de ex., o valoare de asteptare mai mare) ***creste*** transferul de date total.

Trei situatii de baza pentru controlul traficului

Controlul congestiei intre capete

- in cazul congestiei observate la expeditor (de ex., intarzieri) destinatarul va controla expditorul
- Control in bucla inchisa

Controlul congestiei indicate de retea

- Nivelul retea ofera expeditorului feedbackul necesar

Controlul pe baza vitezei

- Comportarea expeditorului este fixata (limitata) in timp
- Control in bucla deschisa

Controlul congestiei intre capete: controlul congestiei pe baza de fereastra

Entitatea de transport transmisa mentine fereastra peste spatiul numerelor din secventa

- se poate trimite un pachet daca numarul de secventa al pachetului se incadreaza in fereastra
- fereastra pentru congestie este diferita de cea destinata controlului traficului.

in asteptare:

- pierderi asumate
- scade marimea ferestrei pentru congestie
- creste valoarea timpului de asteptare (in cazul in care pavhetul a fost intarziat).

La receptia recunoasterii, creste marimea ferestrei.

Controlul congestiei intre terminale: TCP

Foloseste controlul congestiei pe baza de fereastra.

Sunt folosite doua variabile:

- *cwnd: dimensiunea ferestrei de congestie*
- *ssthresh: prag pentru incetinirea vitezei de crestere.*

Startul lent TCP + eliminarea congestiei:

- considera marimea segmentului de 4K
- marimea ferestrei TCP = min(marimea ferestrei de control al traficului, marimea ferestrei de control al congestiei)

```
initialize: cwnd=1
     ssthresh=16
loop: if (ACK received and cwnd <= ssthresh)
         cwnd = cwnd+1
      else if (ACK received and  cwnd > ssthresh)
         cwnd = cwnd + 1/ssthresh
      else if packet timeout
         ssthresh = cwnd/2    /* new thresh half current win */
         cwnd = 1             /* new window size back to 1 */
forever
```

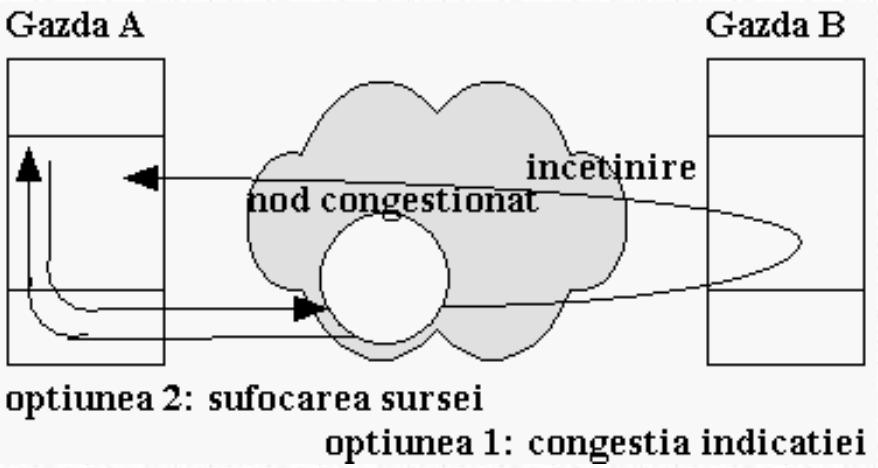
Controlul congestiei indicate de retea

Control pe baza de fereastra strict intre terminale

- Nivelul retea nu este implicat, dar congestia se realizeaza in nivelul retea!

Un caz de indicare-retea: reteaua "marcheaza" pachetele care trec prin nodul congestionat.

- Destinatarul vede marcajul indicand congestia si spune expeditorului sa incetineasca
- Steag in cazul aparitiei congestiei pentru ISO CLNP, CWI (schimba indicatorul ferestrei in cazul traseului virtual SNA al IBM).



Al doilea caz de indicare-retea: dupa detectarea congestiei, ruterele congestionate transmit inapoi mesaje explicite catre sursa de trafic pentru a o incetini.

- Text: pachete sufocate
- Sursa este "linistita" prin ICMP (Internet control message protocol - Protocolul de mesaj pentru controlul Internetului)
- in cazul traseului virtual SNA: VR-RWI bit

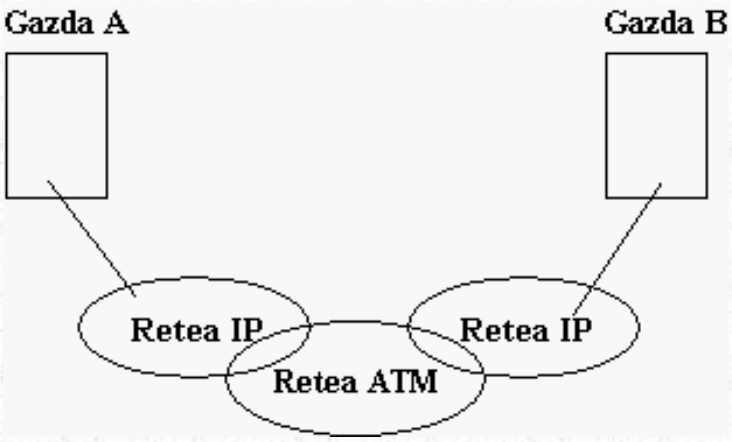
Controlul congestiei indicate in retea: dificultati

Controlul initiat de destinatar poate avea un timp mare de feedback in retelele de mare viteza

- Expeditorul poate avea 1000 pachete transmise (dar nerecunoscute) inainte de a primi indicatia de congestie de la destinatar
- Traseul este deja plin.

Amandoua cazri necesita cuplarea nivelelor retea si transport.

- Este util in cazul retelelor omogene
- Apar probleme in mediul inter-retele, cu nivele de retea diferite.



Controlul congestiei pe baza de viteza

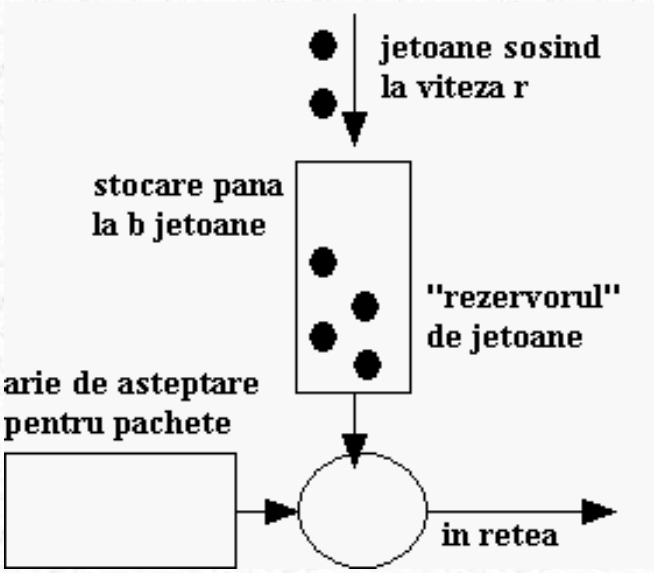
Controlul congestiei este deosebit de dificil in retelele de mare viteza. Mii de pachete "pe fir" se propaga in tzoate

directiile. Cand apare congestia, este prea tarziu pentru a reationa. Congestia se poate elimina prin regularizarea traficului de pachete in retea

- un trafic mai lin va elimina distrugerea pachetelor sosite la acelasi nod de la mai multi expeditori si care cauzeaza congestia
- pachetele care nu se incadreaza se distrug la capatul retelei inainte de a intra in retea.

Controlul congestiei pe baza de viteza: rezervorul cu scurgere

Scop: regularizarea vitezei la care expeditorul poate injecta pachete in retea



Un pachet trebuie sa se potriveasca cu si sa inlocuiasca un jeton inainte de a intra in retea.

Jetoanele adaugate la "rezervor" la viteza r: controleaza viteza pe termen lung a intrarii pachetului in retea.

Cel mult b jetoane se pot acumula in rezervor. Marimea b a rezervorului controleaza sosirile.

Numarul maxim de pachete care intra in retea in t unitati de timp este b+rt.

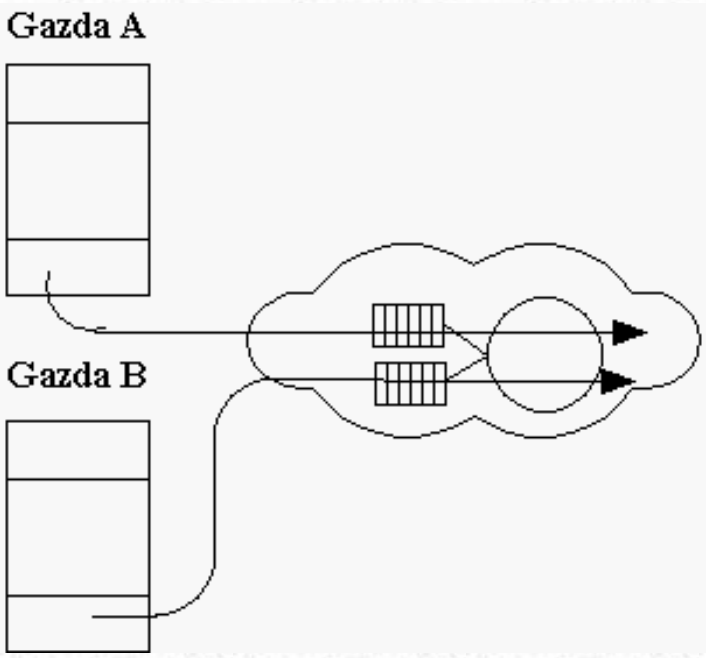
XTP foloseste viteze si parametri de distrugere analogi lui r,b.

Controlul congestiei prin prealocarea buferului

Lipsa buferarii in retea este cauza fundamentala a congestiei. Aceasta se elimina prin alocarea buferelor unei conexiuni terminal-terminal. Daca sunt insuficiente bufere pentru o noua conexiune, se blocheaza (ca in comutarea de circuite).

Legaturile buferelor destinate conexiunii sunt inca partajate.

Protectia se realizeaza prin prevenirea unei comportari gresite a conexiunii.



Nivelul legatura de date este implicat in controlul congestiei nivelului transport.

Controlul congestiei in serviciul ATM ABR

Serviciul ATM ABR (available Bit Rate - viteza de bit disponibila):

- permite expeditorului sa transmita la viteze de pana la o celula de varf (peak cell rate PCR))
- garanteaza o viteza de cel putin o celula (minimum cell rate (MCR)) atunci cand este nevoie
- viteza expeditorului poate fluctua intre 0 si MCR, in functie de nevoile expeditorului si congestiunea in retea.

Controlul congestiei in serviciul ABR:

- Combina aspecte ale controalelor bazate pe viteza si pe indicatii in retea.

Controlul congestiei ATM ABR: EFCI

EFCI: explicit forward congestion indication (indicatii explicite despre directia congestiei)

Se bazeaza pe feedbackul negativ (indicatii despre aspecte negative) la expeditor.

Nodul congestionat (lungimea cozii > pragul) marcheaza bitul EFCI in celula expeditor-la-destinatar.

Destinatarul vede setul EFCI si notigica expeditorul.

Expeditorul reduce viteza celulei:

- ACR: allowed cell rate (viteza permisa a celulei)
- ACR = max(ACR *descreste multiplicativ, MCR)

Expeditorul creste viteza celulei daca nu exista niciun feedback negativ intr-un anumit interval de timp:

- ACR = min(ACR+ creste aditiv, PCR)

Controlul congestiei ATM ABR: viteze explicite

Expeditorul declara fiecare a N-a celula ca celula "RM"

- RM: resource management (management resurse)
- inregistreaza PCR, viteza permisa a apelului (allowed_call_rate) in celula RM
- Campul ER in celula RM: utilizat de comutatoare pentru a ajusta viteza sursei.

Comutarea pe calea expeditor-la-destinatars daca exista congestie

- Determina noi viteze pentru acea sursa (se considera PCR, ACR)
- ajusteaza campul ER pentru a indica noi viteze numai daca acestea sunt mai mici decat valoarea curenta a lui ER.

Managementul conexiunii: paradigme ale conexiunii

Orientat pe conexiune

- Instaleaza/dezinstaleaza in mod explicit conexiuni
- Numar de secventa initial, dimensiunea ferestrei de control al traficului
- Schimba date in contextul conexiunii
- de ex., TCP, ISOI TP4

Serviciu fara conexiune

- Datagrama pura
- Transmisie unica nesigura
- de ex., UDP (RFC 768), ISP CLTP (ISO 8072)
- Orientata pe tranzactii
- Solicitare unica de la expeditor, un singur raspuns de la destinatar
- Protocol VMTP.

Managementul conexiunii: probleme fundamentale

Sursa problemelor:

- Reteaua poate produce intarziere, reordonarea pierde pachete.
- Asteptarea/retransmisia introduce duplicate in date, recunoastere, conexiune, inchide pachetele.

La sosirea pachetelor:

- Noi solicitari/livrari de conexiuni de la "clientul in viata" sau de la un altul mai vechi
- Protocoalele de transport trebuie sa creeze/mentina/distruge suficiente informatii despre "stare" pentru a raspunde la intrebari
- Conexiune explicita stabilita/dezactivata cui serviciul orientat pe conexiune.

Managementul conexiunii: alegerea unui unic identificator

Problema: sa se aleaga un identificator (de ex., numar) a.i. niciun alt pachet asociat cu prezenta gazda in retea sa nu aiba acelasi identificator.

- Gazda este unica la nivel global, la fel ca adresele concatenate si identificatorul unic
- presupunere: cunoastem timpul de viata a pachetului in retea (T)

Abordare: mentinerea starii

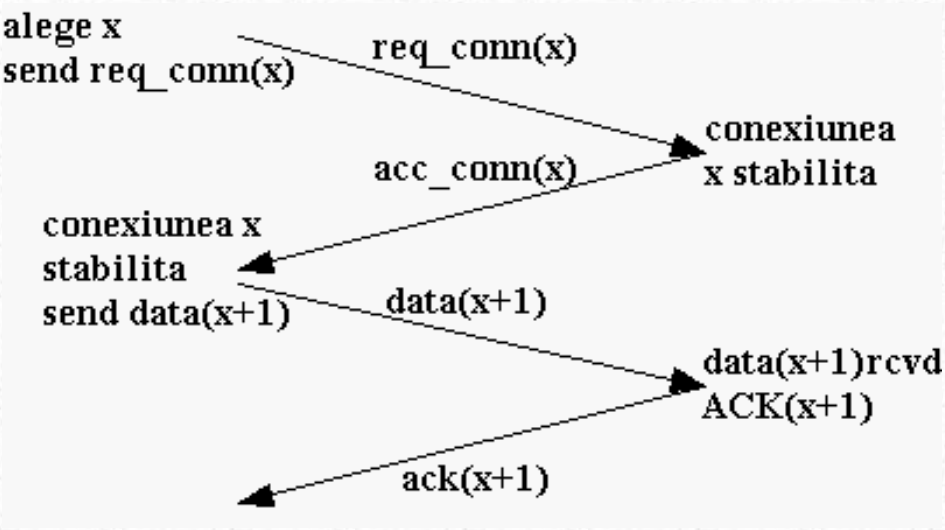
- Se pastreaza lista tuturor valorilor folosite in ultima perioada 2T
- Daca lista este pierduta, se asteapta 2T

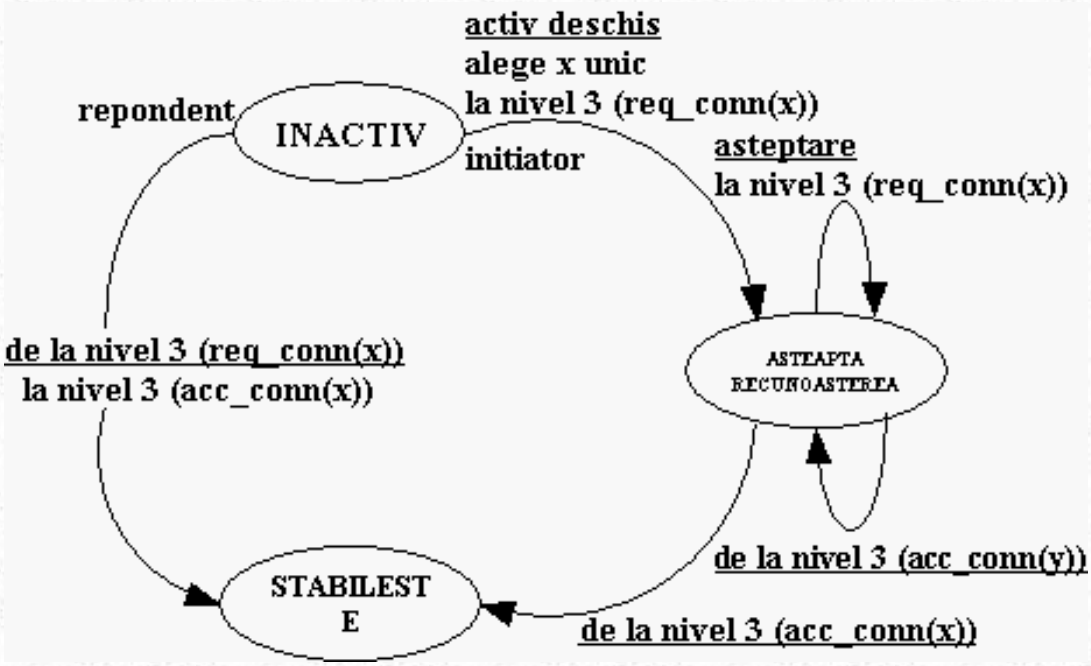
Stabilirea conexiunii: legatura pe doua cai

Initiatorul transmite mesajul req_conn(x) catre cealalta parte (repondent).

X este identificator unic.

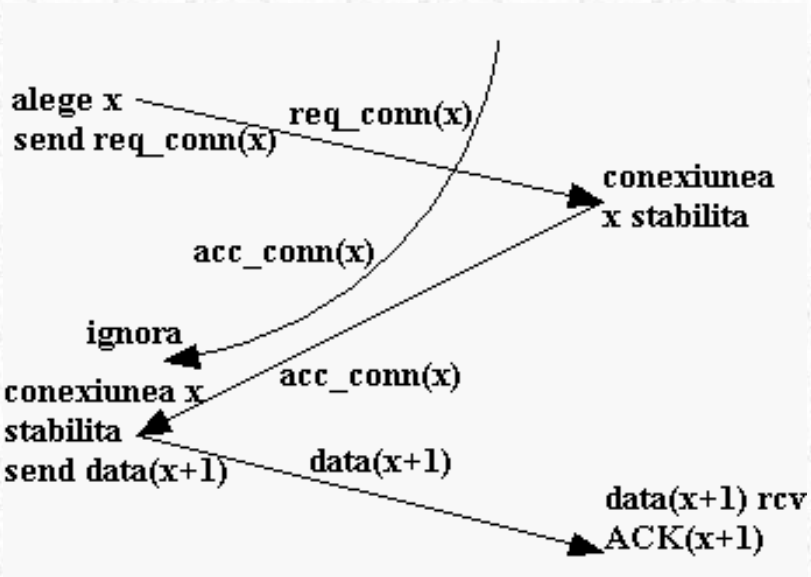
Repondentii accepta conexiunea via raaspunsul acc_conn(x).



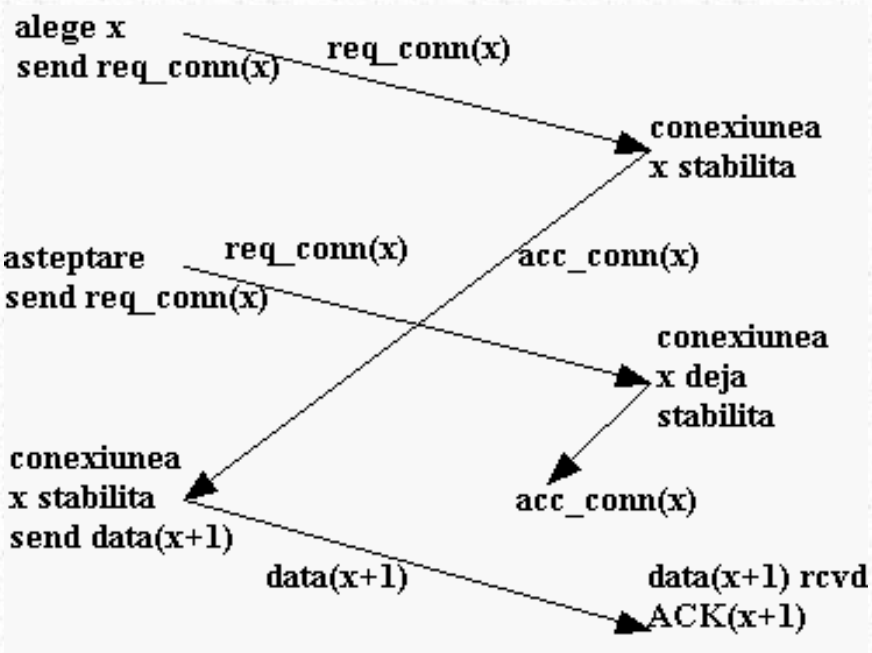


Legatura pe doua cai: mesaje vechi

acc_conn(y) recunoscut ca vechi!

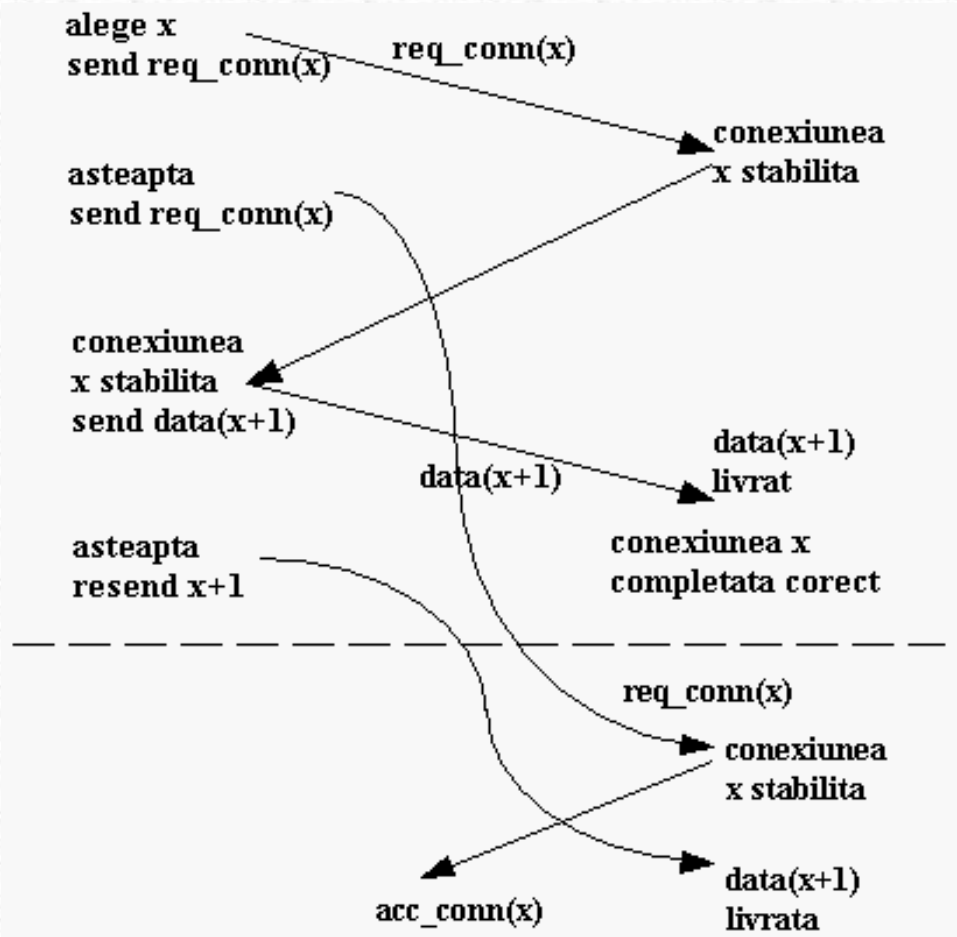


Legatura pe doua cai: manuirea duplicatelor



Legatura pe doua cai: scenariul esuarii

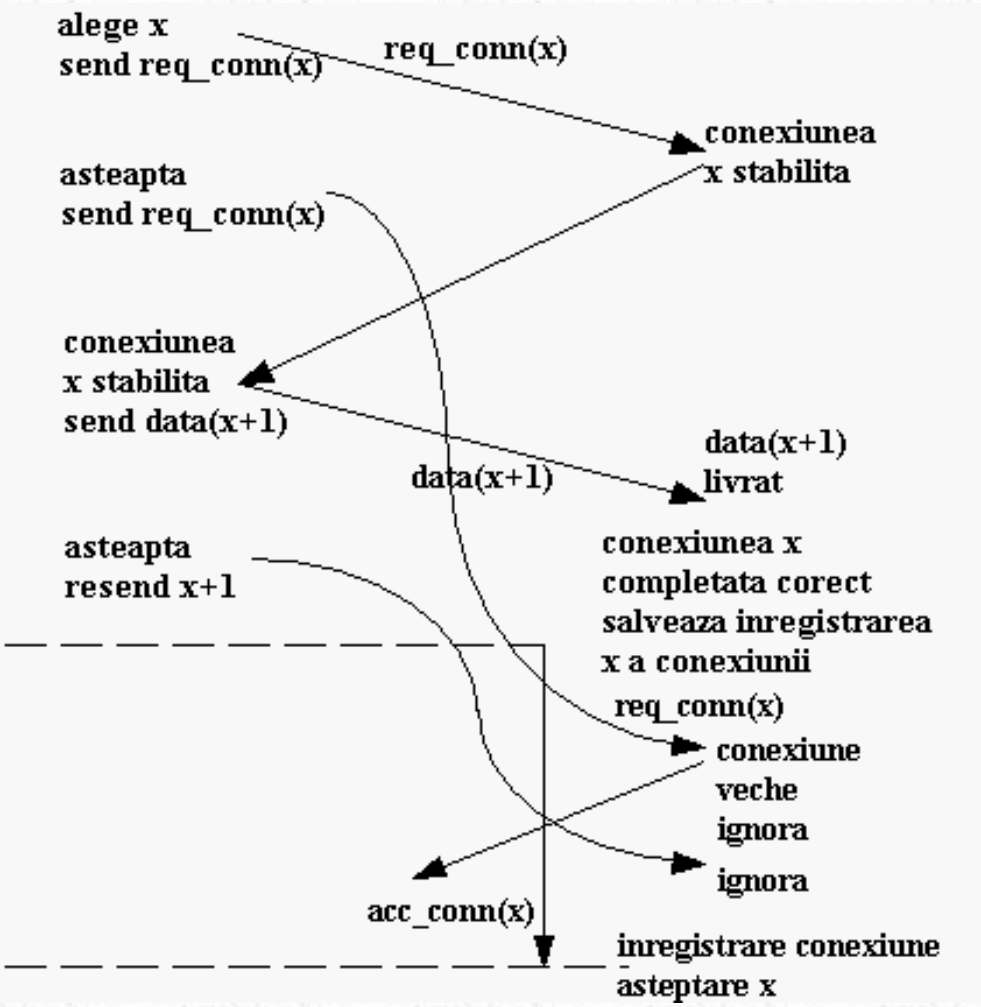
Destinatarul spune daca req_conn(x) este sau nu un duplicat.



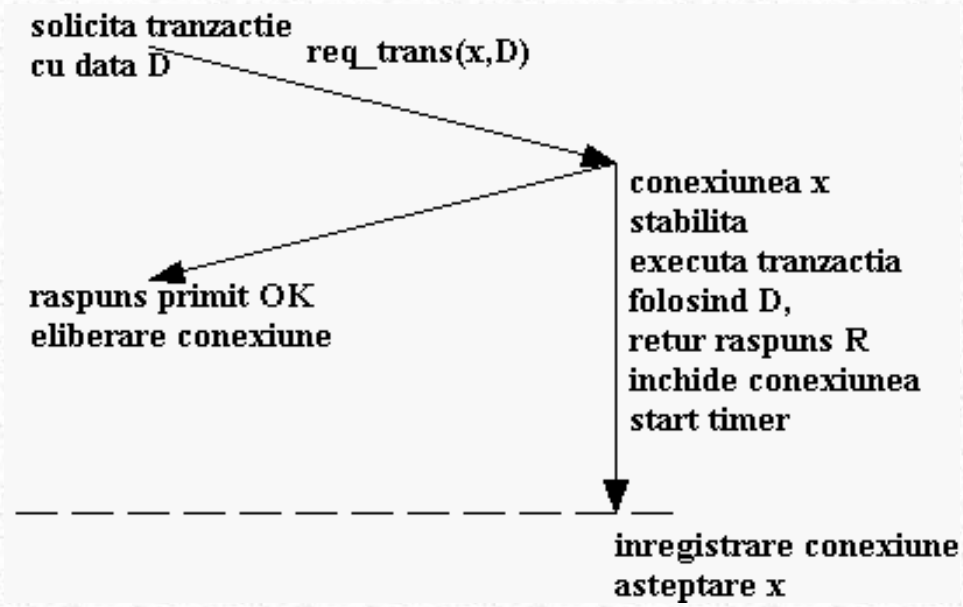
Legatura pe doua cai cu timere

Destinatarul nu va sterge inregistrarea conexiunii pentru x pana cand este sigur ca nu mai exista req_conn(x) in retea.

- Se pastreaza inregistrarea timp de T dupa inchiderea conexiunii.



Legatura pe doua cai: tranzactiile



Solicitare de deschidere a conexiunii, datele trec, inchiderea conexiunii se realizeaza cu un pachet.

- Este necesara doar o intarziere de o runda pentru "tranzactii"
- Destinatar: la receptie, se executa operatia pe date, se returneaza raspunsul, se inchide conexiunea.

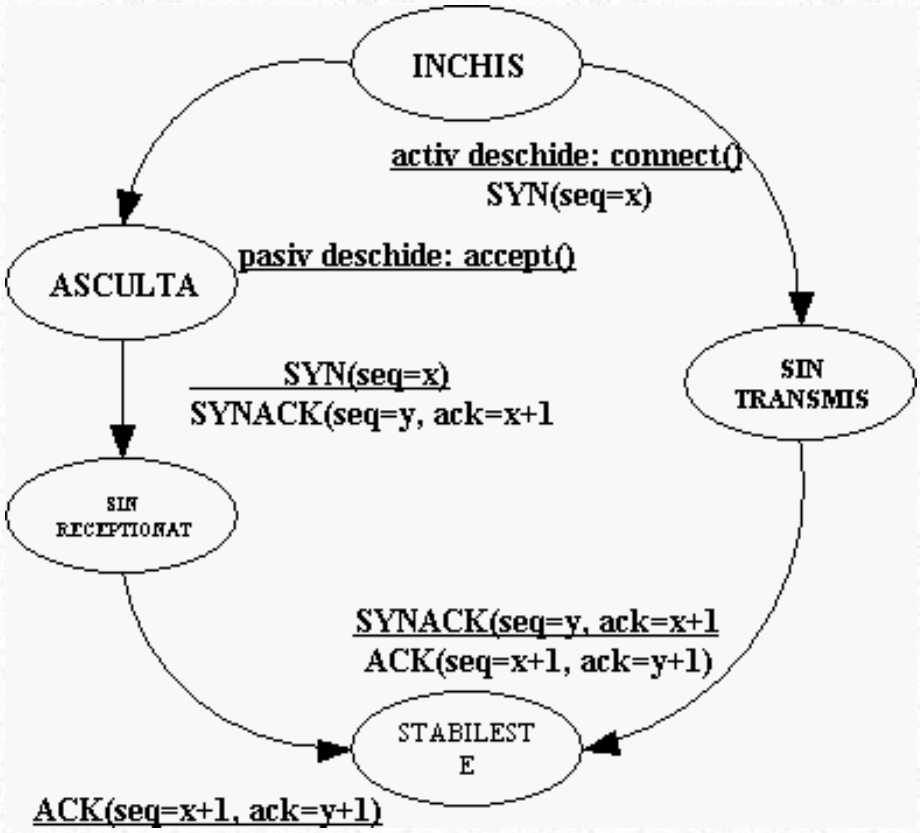
Legatura pe trei cai

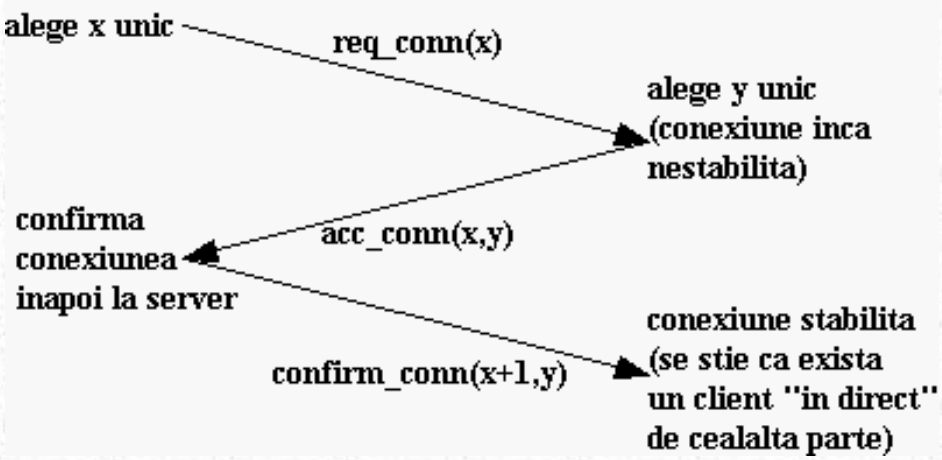
Legatura pe doua cai:

- Expeditorul alege un unic identificator, x
- Permite expeditorului sa detecteze vechile raspunsuri de la destinatar (destinatarul trebuia sa replice cu x)
- Permite destinatarului (cu timerele pe valoarea x) sa detecteze mesajele vechi ale expeditorului.

Legatura pe trei cai

- Se lasa de asemenea destinatarul sa aleaga propriul lui identificator, z, si se solicita expeditorului sa replice folosind z
- permite destinatarului sa detecteze mesajele vechi ale expeditorului fara a folosi timere
- Necesita trei cai de schimb a mesajelor pentru a seta conexiunea.

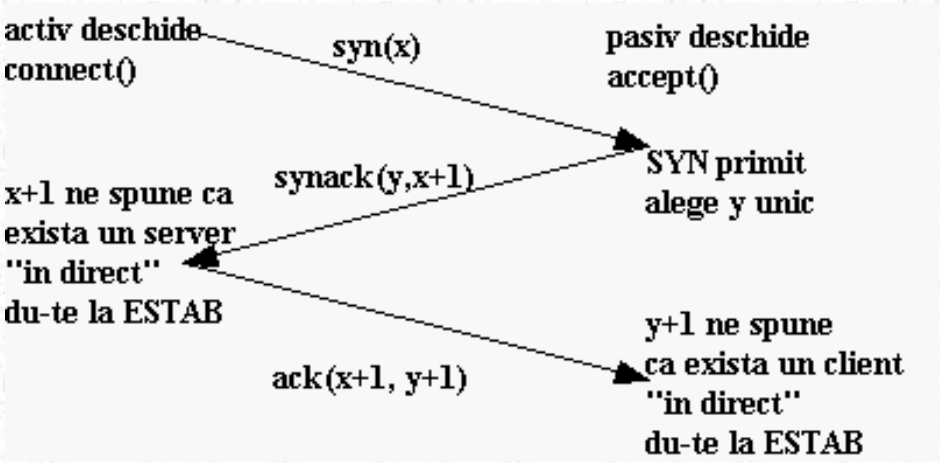




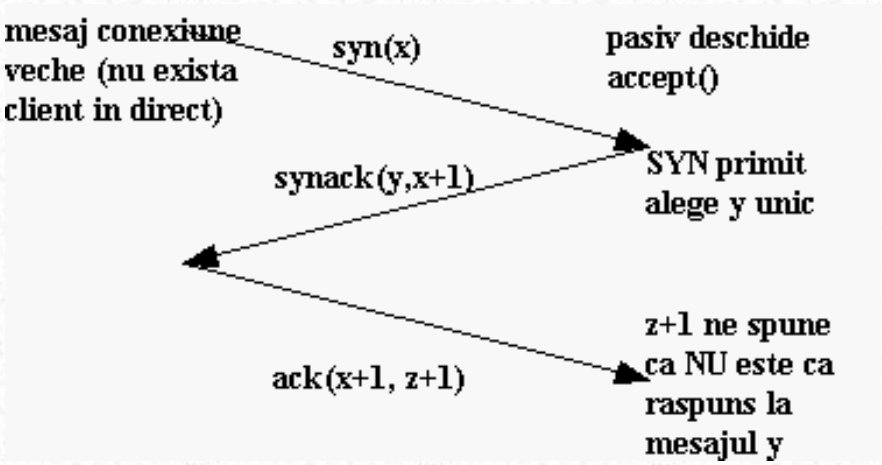
Legatura pe trei cai:

- Folosita in TCP, TP4, DECnet

- Bitii de header in TCP seimpacheteaza pentru SZN, ACK
- Sunt necesare bucle suplimentare pentru cazul lipsei timerelor.



Scenariul legaturii in TCP



inchiderea unei conexiuni

Doua abordari pentru inchiderea conexiunii

- Renuntare: trimite mesajul close(), inchide conexiunea, sterge informatiile de stare
- Pentru a nu se pierde mesajul close)=, inainte de a sterge informatiile despre stare asteapta recunoasterea lui close()

inchiderea lina a TCP:

- Initiatorul transmite mesajul FIN(x) celeilalte parti (repondentului) pana la ACK(x+1)
- Daca ACK(x+1) nu este primit, repondentul nu trebuie sa trebuie sa se opreasca, pentru ca trebuie sa retransmita ACK(x+1) mai tarziu

Timere pentru protocoale de transport

Exista mai multe timere pentru nivelul transport.

- Timer pentru asteptare-retransmitere
- Timer pentru inchiderea implicita a conexiunii: legatura pe doua cai
- Asteptarea pentru stabilirea conexiunii: renunta daca nu exista un raspuns la SZN in 75 sec.
- Timer de recunoastere intarziata - incearca sa transmita datele de la destinatar la expeditor pentru recunoastere, asteapta 200 ms inainte de a genera recunoastere autonoma.

Timeri aditionali:

- Timer pentru controlul traficului (timer pentru persistenta TCP) daca destinatarul are setata freastra la 0 si nu a primit date sau solicitari recente.
- Timer de pastrare in viata: daca nu exista nicio activitate o anumita perioada, genereaza pachetul "I'm OK are you OK packet" – 2 ore in TCP

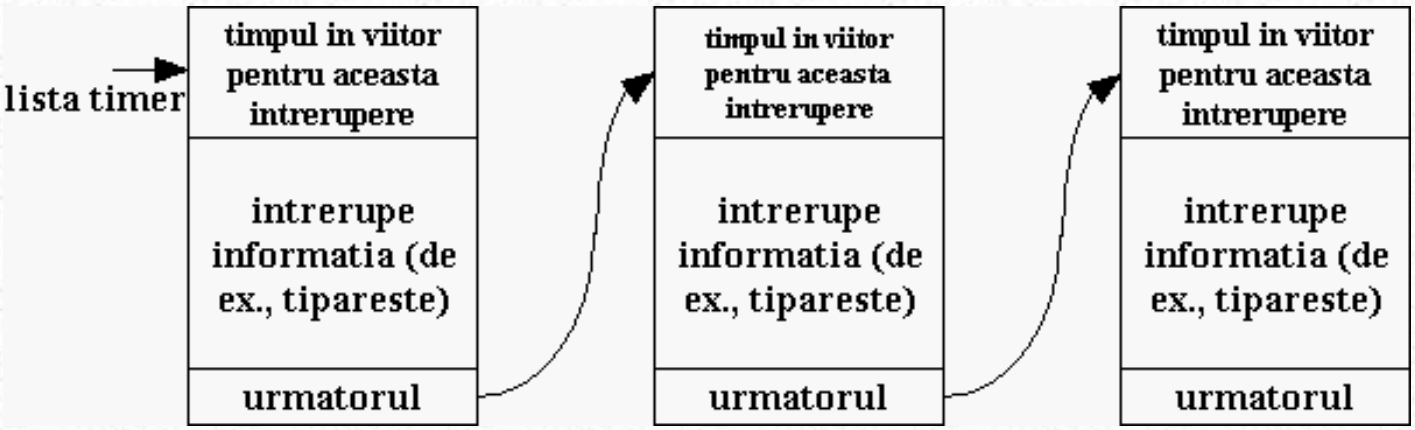
Timere: implementare

Fizic: un simplu timer de numarare inversa

- Initializat, pornit si oprit prin software
- Va genera intreruperi dupa ce numaratoarea ajunge la zero
- Codul de protocol (posibilitatea de intrerupere a timerului) initiat ca raspuns la intrerupere.

Logic: pot functiona mai multe timere

- Toate valorile viitoare de asteptare ale timerelor inregistrate in structurile de date
- Timerul hardware numara invers pana la cel mai apropiat timp de intrerupere
- La intrerupere:
  - Executa activitatea solicitata
  - Consulta structura de date, incarca timerul fizic pentru urmatorul timp de intrerupere cel mai apropiat, porneste timerul
  - Retur de la intrerupere.



intarzierile in bucla estimate

Valoarea timerului de retransmisie se bazeaza pe estimarea intarzierii in bucla.

intarzierea cap-cap este variabila in WAN (rezulta congestia).

Timpul in bucla estimat (estimating round trip time (RTT)): mediere exponentiala

- timpul de inregistrare de la pachet se transmite la receptia ACK
- se proceseaza noul RTT estimat folosind noi intarzieri in bucla masurate (M) si estimari vechi:
  - RTT ß a*RTT + (1-a)M
  - A in intervalul [0,1], RTT se schimba lent cand a se apropie de 1, si rapid cand a se apropie de 0
- Complicatie: problema retransmisiei

Timere: valoarea timerului de retransmisie

Timerul de retransmisie trebuie sa fie in functie de RTT (asa cum s-a estimat mai sus)

Valoarea de asteptare = b * RTT

Specificatiile TCP originale recomanda b=2

La asteptarea pachetului

- Creste valoarea timerului: pierderea intarzierii asumate datorita congestiei (mai degraba decat coruperii)
- Dublarea valorii de asteptare este comuna (pana la un anumit prag)

Timer de retransmitere TCP: algoritmul lui Jacobson

Algoritmul original pentru timerul TCP a fost inlocuit la sfarsitul anilor 1980.

Noi abordari:

Se ajusteaza timerul ca functie de RTT si rezulta o masura a deviatiei (D):

D = aD + (1-a)|RTT-M|

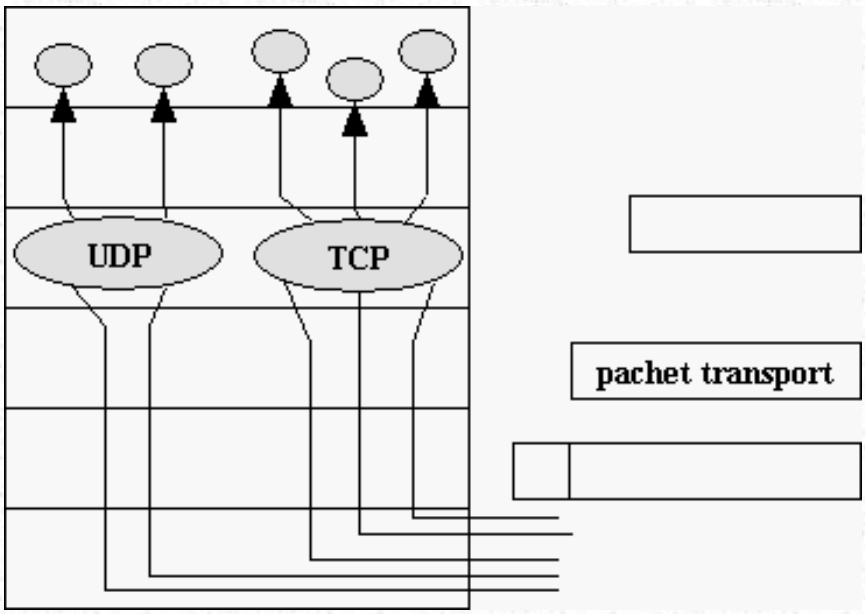
Valoarea de asteptare = RTT + 4*D

Multiplexarea si adresarea

Protocolul nivelului de transport administreaza adesea multiple conexiuni de nivel superior.

- Pachetul la nivelul N trebuie sa contina informatii despre protocolul de nivel N+1de trecere a "datelor".
- Protocolul TCP va administra simultan multiple conexiuni (socketi deschisi)
- Trebuie sa poata despacheta (demultiplexa) pachetele care sosesc pentru a corecta conexiunea de nivel superior (de ex., socket).
- TCP foloseste informatii despre adresa IP-port local si la distanta pentru demultiplexare

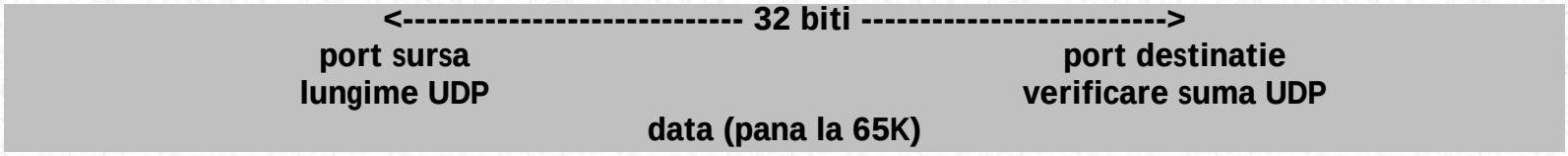
Nivelul retea poate necesita o demultiplexare a unui sau mai multor protocoale de transport posibile (de ex., UDP sau TCP)



Studiu de caz pentru nivelul transport pentru Internet: TCP si UDP

UDP: servciu de datagrama

- orientat pe mesaj, un timp transmite apoi prezerva limitele mesajului
- nesigur
- fara retransmisie
- detectie de eroare folosind verificarea sumei pe Internet peste pachetul UDP si 3 campuri de headere IP (lungimea adresei si IP)
- fara controlul traficului sau al congestiei
- fara ordonare



Implementarea UDP: codul sursa

Implementarea Linux a UDP partea de transmisie

```
/* udp_send called from above, as result of sendto system call */
static int udp_send(struct sock *sk, struct sockaddr_in *sin,
         unsigned char *from, int len, int rt)
{
        struct sk_buff *skb;
        struct device *dev;
        struct udphdr *uh;
        unsigned char *buff;
        unsigned long saddr;
        int size, tmp;
        int ttl;

        /* Allocate an sk_buff copy of the packet. */
        size = sk->prot->max_header + len;
        skb = sock_alloc_send_skb(sk, size, 0, &tmp);
        if (skb == NULL)
                return tmp;

        skb->sk       = NULL;   /*
to avoid changing sk->saddr */
        skb->free     = 1;
        skb->localroute = sk->localroute|(rt&MSG_DONTROUTE);

        /*Now build the IP and MAC header. */

        buff = skb->data;
        saddr = sk->saddr;
        dev = NULL;
        ttl = sk->ip_ttl;
```

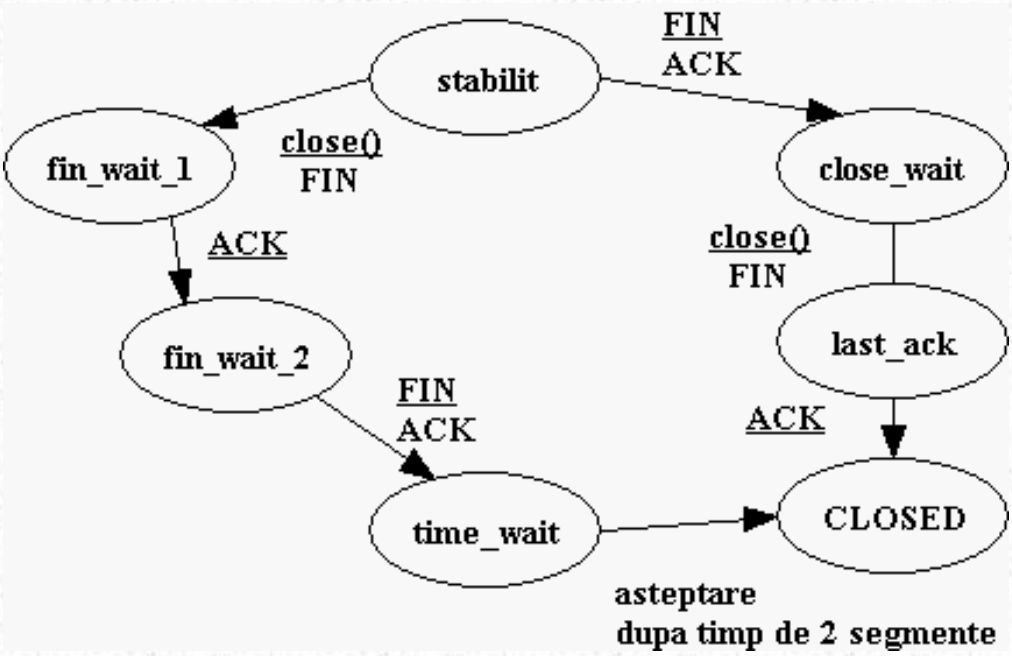
```
        tmp = sk->prot->build_header(skb,
saddr, sin->sin_addr.s_addr,
                        &dev, IPPROTO_UDP, sk->opt,
skb->mem_len,sk->ip_tos,ttl);
        skb->sk=sk;     /* So memory is freed correctly */
        /*Unable to put a header on the packet. */

        if (tmp < 0 )
        {
                sk->prot->wfree(sk, skb->
mem_addr, skb->mem_len);
                return(tmp);
        }

        buff += tmp;
        saddr = skb->saddr; /*dev->pa_addr;*/
        skb->len = tmp + sizeof(struct udphdr) +
len;    /* len + UDP + IP + MAC */
        skb->dev = dev;

        /*Fill in the UDP header.  */

        uh = (struct udphdr *) buff;
        uh->len = htons(len + sizeof(struct udphdr));
        uh->source = sk->dummy_th.source;
        uh->dest = sin->sin_port;
        buff = (unsigned char *) (uh + 1);

        /*Copy the user data.  */
        memcpy_fromfs(buff, from, len);

        /*Set up the UDP checksum.  */
        udp_send_check(uh, saddr, sin->sin_addr.s_addr,
 skb->len - tmp, sk);

        /* Send the datagram to the interface.  */
             udp_statistics.UdpOutDatagrams++;
        sk->prot->queue_xmit(sk, dev, skb, 1);
        return(len);
}
```

TCP: apel setare si inchidere

Legatura pe trei cai pentru setarea conexiunii.

inchiderea conexiunii:

- Fiecare parte din TCP trebuie sa determine inchiderea alteia



TCP: transfer de date

Orientat pe flux (flux de octete fara limitari de mesaj)

Verificarea sumei pe Internet ca in UDP

Cand trebuie sa se transmita in TCP?

- RFC spune "oricand"
- fereastra de control al traficului si al congestiei restrictioneaza transmiterea
- operatiile interactive: ecoul caracterului in telnet
  - header de 40 octete + un octete pentru incarcare
  - algoritmul lui Nagle: cu un singur caracter la intrare, se bufereaza caracterele pana cand se obtine recunoasterea pentru ultimul batch de caractere transmis
  - retea rapida necongestionata: nu intereseaza suprahead-ul (overhead), pe cand in reteaua lenta se bufereaza caracterele pentru a salva suprahead-ul.

La destinatar:

- recunoastere cumulativa
- RFC nu spune cand sa se faca recunoasterea
- pentru pachetele care nu sunt in ordine, TCP nu specifica actiunea
  - de obicei se bufereaza si se recunoaste ultimul pachet in ordine
- recunoasterea poate sa nu fie generata imediat
  - se spera in returnarea datelor in directie inversa
  - se spera ca la sosirea celui de al doilea pachet, sa fie recunoscute amandoua pe loc

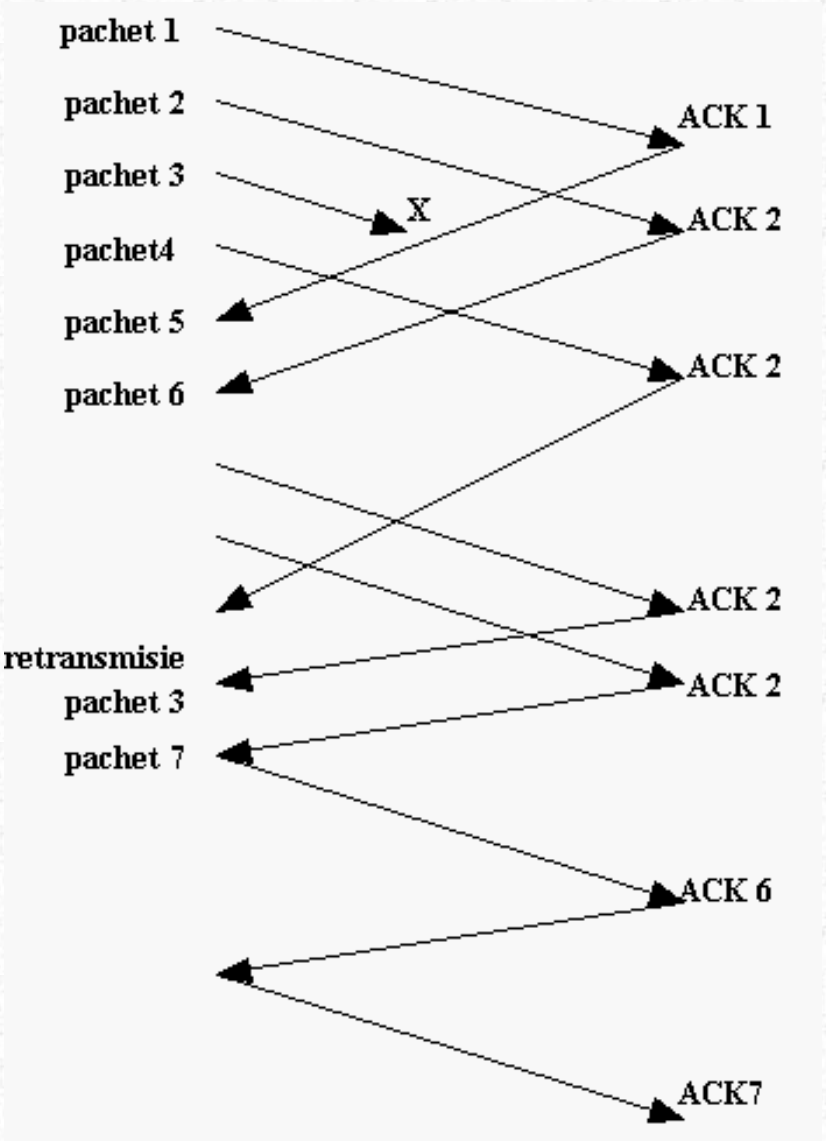
TCP: retransmiterea rapida cu recunoastere cumulativa

Retransmitere rapida dupa receptia a trei recunoasteri (N) in timp ce se asteapta pentru recunoasterea (N+1)

- Se retransmite N fara asteptare (N a fost ca si pierdut)

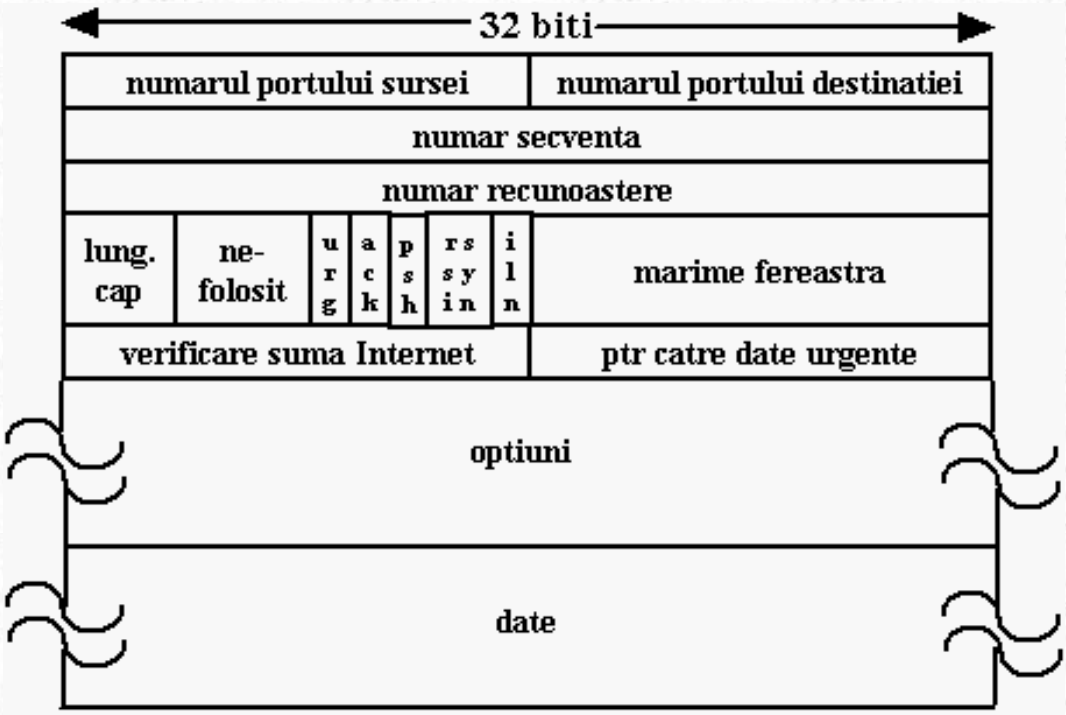
Exemplu:

- retransmiterea rapida a lui 3
- recunoasterea cumulativa a lui 6



Formatul pachetului TCP

Despre multe din campurile de pachete am vorbit deja.



Campuri despre care nu s-a vorbit:

- lungimea: lungimea headerului (ariabila, datorita optiunilor)
- URG: daca se seteaza, pointerul urgent indica datele "urgente"
- PSH: TCP trebuie sa impinga aceste date catre destinatar cat mai curand posibil. Se foloseste sau se poate folosi adesea.
- optiuni: pot fi folosite pentru a controla tactul, sau informatii despre marimea maxima a segmentului.

Ferestre si timere TCP

Estimarea RTT se bazeaza pe intarzierile masurate ale recunoasterii.

Timere de retransmitere

- algoritm vechi: 2*RTT
- algoritm nou: RTT +4*factorul de variatie

Controlul traficului cu ajutorul ferestrei:

- destinatarul avertizeaza asupra spatiului buferului la expeditor

Controlul congestiei cu ajutorul ferestrei:

- start lent
- creste mult mai incet

Aspecte terminal-terminal pentru ATM

Nivelul de adaptare ATM (ATM adaptation layer (AAL)) permite functionarea terminal-sistem in retelele ATM

- diferite AAL ofera servicii diferite nivelelor superioare

- functionalitate comuna:
  - fragmentarea si reasamblarea: spargerea unitatilor de date la nivel de aplicatie in fragmente de 48 octete pentru nivelul de retea ATM
  - detecteaza erorile, dar nu le corecteaza

AAL1:

- pentru timp real, aplicatiile la viteze de bit constant precum audio necomprimat, video
- prezerva timingul in cazul terminal-terminal

AAL2: nefunctional

AAL3/4 si AAL5

- transportul de date pentru datele sensibile la eroare si care nu sunt in timp real
- mod mesaj (prezerva limitarile mesajului) si mod flux.

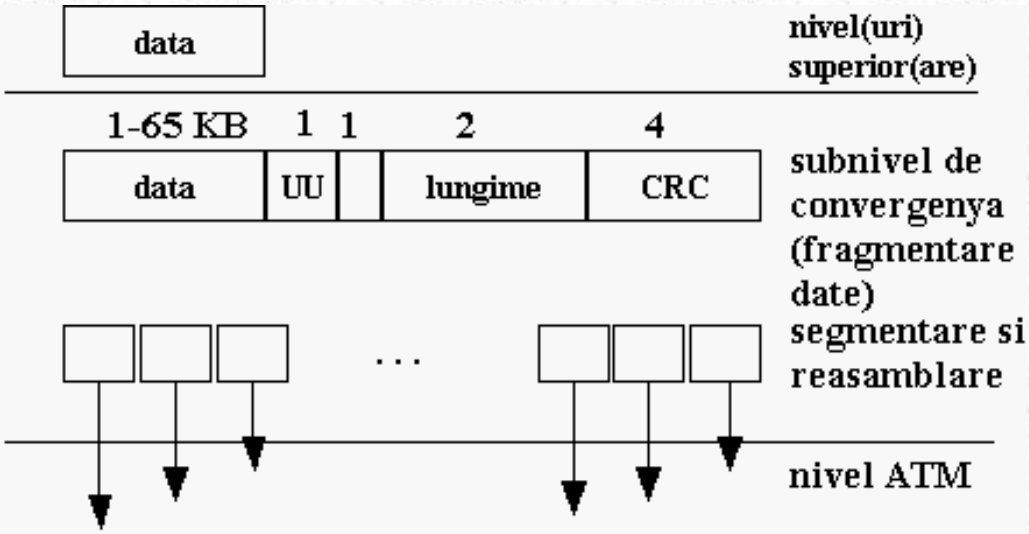
AAL5 Studiu de caz

Seamana mult cu UDP

Va detecta erori dar nu le va retransmite

Formatul pachetului:

- *data:* pana la 65K octetes per mesaj AAL5
- *UU:* camp utilizator (nivel superior) nefolosit de AAL. utilizari posibile sunt numerele secventiale si/sau multiplexarea
- lungimea: lungimea datelor
- CRC: verificarea redundantei ciclice (cyclic redundancy check): detectie de erori mai puternica decat verificarea sumei pe Internet (checksum). Erorile detectate (semnalate catre nivelul superior) dar necorectate.



Nivel transport: perspectiva

Cerintele noilor aplicatii impun noi mecanisme pentru nivelul de transport.

Se cauta alaturarea unui set specific de functionalitati cu un protocol monolitic cum este TCP, pentru a permite protocoale flexibile in cazul blocurilor de constructie mai mici.

Inter-retele: cand se traverseaza mai multe retele, cel mai de jos nivel de serviciu este minimal. Este

necesara functionalitatea nivelelor de transport bogate.

Back Top Next

E-mail: engineering@teleactivities.net

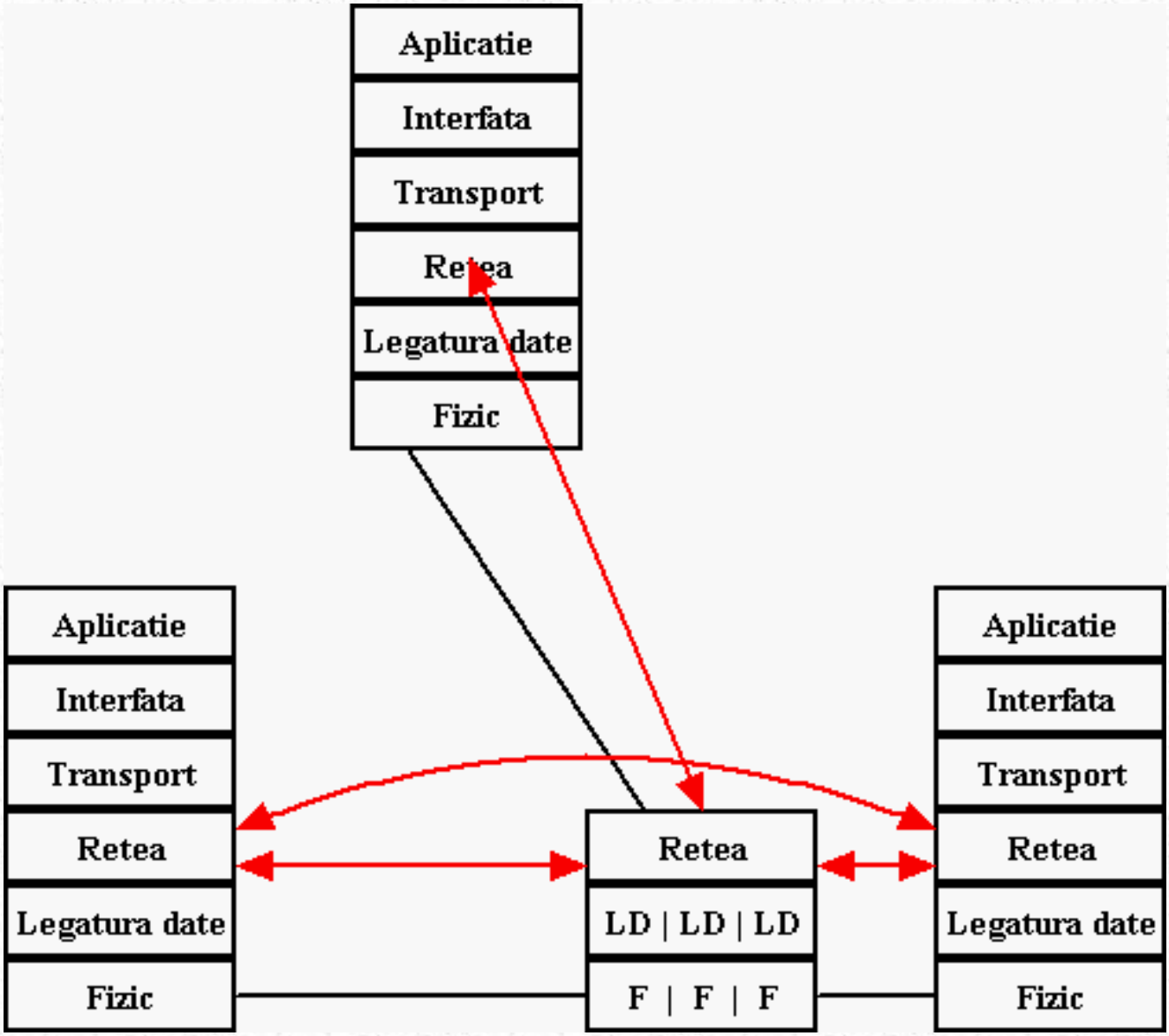
Retele, protocoale si servicii asigurate prin retelele de calculatoare

6. Nivelul retea

Introducere

Nivel retea: aspecte *generale*:

- *nivel transport:* intre doua gazde
- *nivel legatura date:* intre doua gazde conectate fizic, rutere
- *nivel retea:* implica fiecare ruter in part, gazda, poarta din retea

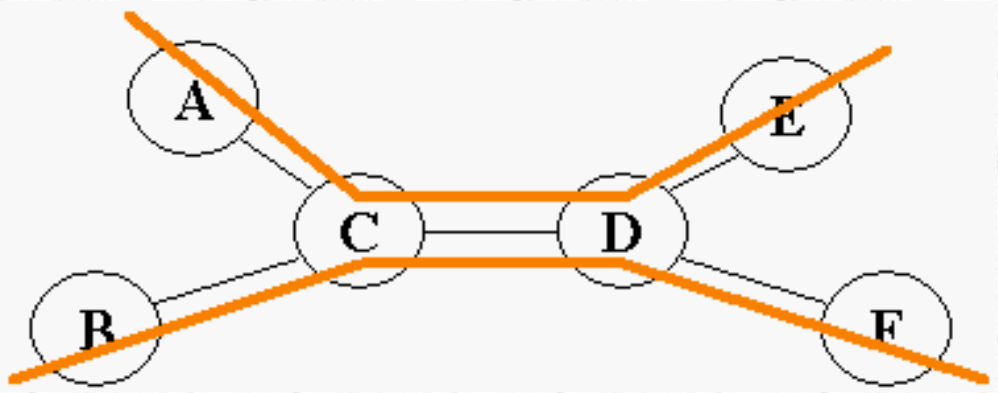


Serviciu nivel retea: circuit virtual

Virtual, pare un circuit, dar nu este.

in general asociat cu serviciu orientat pe conexiune.e

Toate pachetele din conexiune urmaresc aceeasi ruta.



La timpul de stabilire a conexiunii:

- pachetul de setare a conexiunii trece de la expeditor la destinatar
- tabelele de rutare sunt actualizate in nodurile intermediare pentru a reflecta noile VC
- esential: starea la ruter per conexiune
- se potrivesc bine cu garantiile SC: rezerva resurse si/sau accepta/refuza apeluri bazate pe resurse la acest ruter

Analogie: reteaua telefonica.

Serviciu nivel retea: datagrame

Nu exista o notiune privind conexiunea in nivelul retea.

Nu exista nicio setare a ruterelor la timpul de stabilire a conexiunii - fiecare pachet "in conexiune" poate urma cai diferite.

Nu exista o garantie de siguranta, sau avantajele livrarii in ordine.

Avantaje:

- nicio stare de conexiune in rutere
- robust fata de vulnerabilitatile legaturii
- recuperare la sistemele terminal (nivel transport).

Diferite modele de serviciu:

- cel mai bun serviciu din puctus de vedere al efirtului: datagramele
- servciul cu performante garantate: SC

Functia de rutare

Un pachet pentru nivelul retea contine:

- pachetul nivelului transport (port, secventa, recunpastere, date, verificare sume, etc)
- informatii de adresare (de ex., sursa, adresa destinatarului sau identificator VC)
- alte cimpuri (de ex., versiunea, lungimea, timpul de viata)

Ruterul/comutarea actioneaza simplu la receptia pachetului:

- cauta identificatorul de pachet (adresa destinatarului sau identificatorul VC) in tabela de rutare si il directioneaza catre legatura de plecare adecvata (sau in sus daca intr-acolo este destinatia).

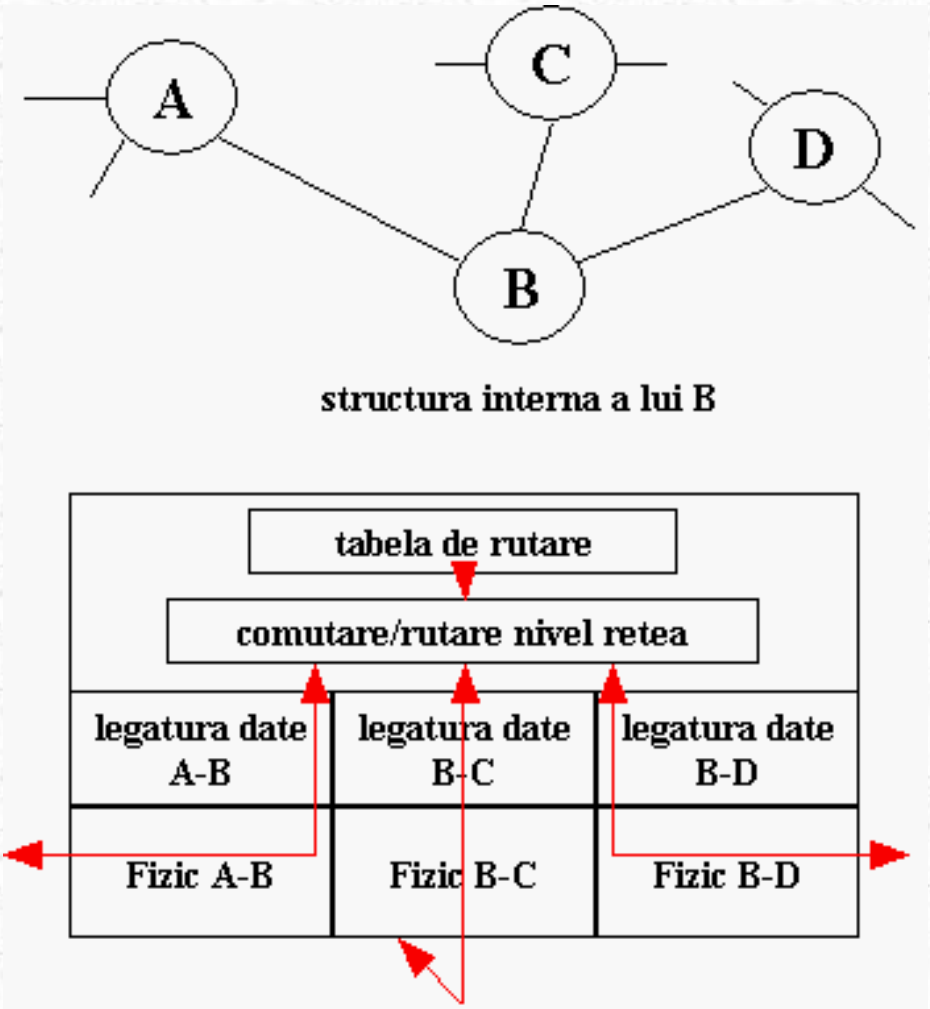


Tabela de rutare: probleme

Probleme de rutare

Scalabilitatea: trebuie sa poata suporta numere mari de gazde, rutere, retele

Se adapteaza rapid si eficient la schimbari in topologie sau modificari semnificative in trafic.

Selectia rutei poate depinde de diferite criterii.

Clasificarea algoritmilor de rutare

Centralizare sau descentralizare

- *centralizat:* situl central proceseaza si rutele distribuite (echivalent: informatii despre rutele de procesare cunoscute ca globale, fiecare ruta efectueaza aceeasi procesare)
- *decentralizat:* fiecare ruter vede numai informatii locale (de la el insusi si vecinii fizic conectati) si determina ruterele pe aceasta baza

Static sau adaptiv

- *static:* tabela de rutare se modifica foarte incet, adesea ca raspuns la interventia umana
- *dinamic:* tabelele de rutare se schimba cu modificarea traficului sau topologiei de retea

Doua abordari de baza adoptate in practica:

- rutarea starii de legaturi: centralizat, dinamic (functioneaza periodic)
- vector distanta: distribuit, dinamic (ca raspuns direct la schimbari).

Rutarea starii de legatura

Fiecare nod cunoaste topologia retelei si costul fiecarei legaturi.

- cvasi-centralizat: fiecare ruter difuzeaza periodic costurile legaturii atasate.

Costul poate reflecta

- intarzierile in asteptare pe legatura
- largimea de banda a legaturii
- toate legaturile cu costuri egale: rute cu calea cea mai scurta.

Folosite in Internet OSPF, ISO IS-IS, DECnet, "noi" (1980) algoritmul de rutare ARPAnet.

Scop: determinarea caii cu costul cel mai mic de la un nod (sursa) la toate celelalte noduri: algoritmul pentru calea cea mai scurta al lui Dijkstra.

Algoritmul pentru calea cea mai scurta al lui Dijkstra: definitii

Definim:

- c(i,j): costul legaturii de la i la j. c(i,j) = infinit daca i,j nu sunt conectate direct. Vom presupune c(i,j) egal cu c(j,i) dar nu intotdeauna adevarat in practica.
- D(v): costul caii cunoscute in prezent cu cheltuielile cele mai mici de la sursa la nodul v.
- p(v): nodul anterior (vecinul lui v) in lungul caii curente cea mai scurta de la sursa la v
- N: set de noduri a carei cale cea mai scurta de la sursa este definitiv cunoscuta.

Presupunem:

- nodul sursei este A
- interativ: dupa k iteratii, se cunosc caile la "cel mai scurt" k (costul caii) la A.

Algoritmul lui Dijkstra: expunere

Initializare:

N = {A}

pentru toate nodurile v

daca v adiacent lui A

atunci D(v) = c(A,v)

altfel D(v) = infinit

Bucla

Gaseste w care nu este in N astfel incat D(w) sa fie minim.

adauga w la N

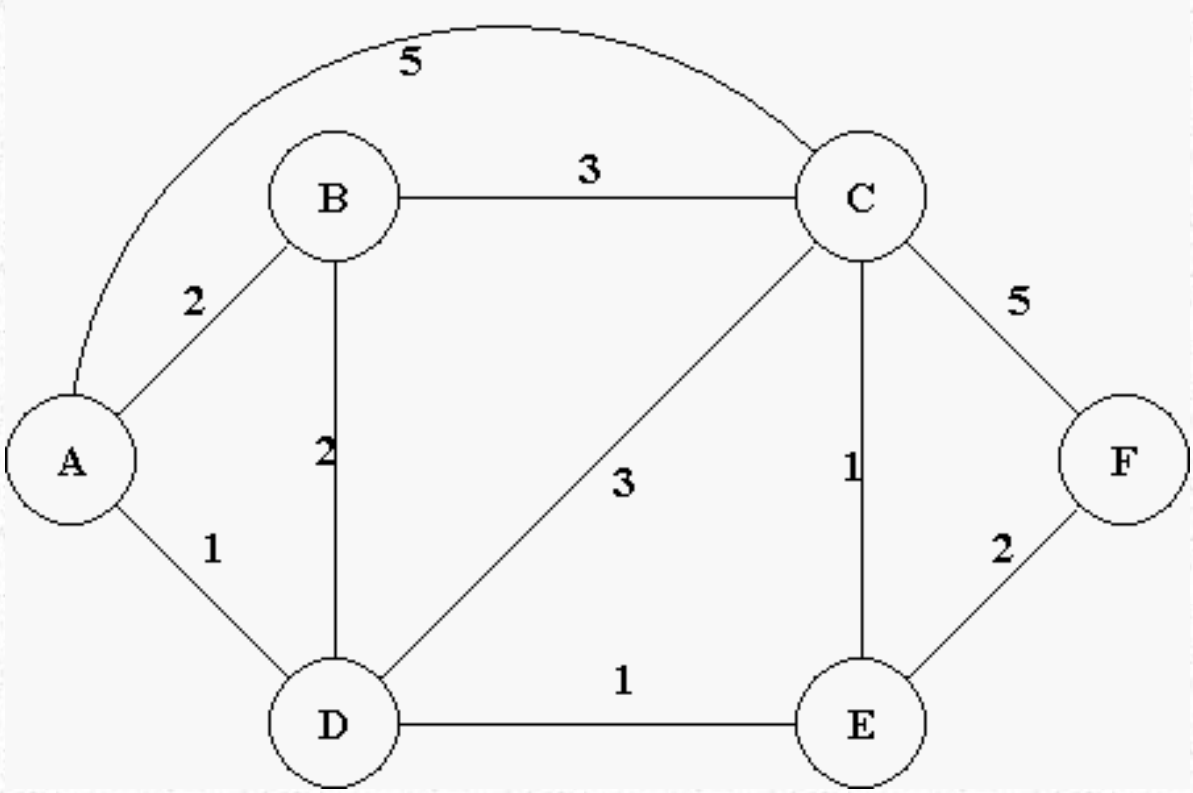
actualizeaza D(v) pentru toti v care nu sunt in N:

D(v) min(D(v), D(w) + c(w,v))

/* noul cost la v este sau costul vechi la v sau costul cunoscut al celei mai scurte cai la w plus de la w la v */

Pana cand toate nodurile ajung in N.

Algoritmul lui Dijkstra: exemplu



treapta	N	D(B),p(B)	D(C),P(C)	D(D),P(D)	D(E),P(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinit	infinit
1	AD	2,A	4,D		2,D	infinit
2	ADE	2,A	3,E			4,E

3	ADEB		3E			4E
4	ADEBC					4E
5	ADEBCF					

- exemplu: in treapta 1: D(C) D(D)+c(D,C)

1+3

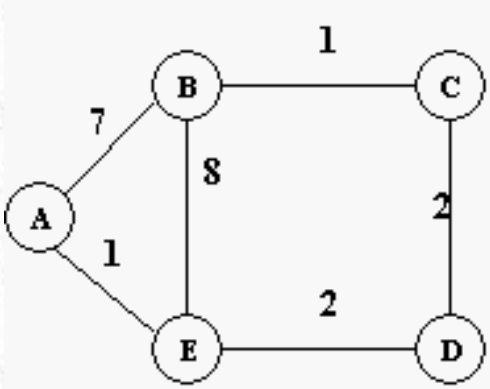
- pentru fiecare coloana, ultima intrare da imediat informatii despre calea cu costul cel mai mic la/de la A, si costul la acel nod
- timpul de rulare pentru cazul cel mai rau: O(N**2)

Rutarea vectorului distanta

Procesare asincrona, iterativa, distribuita:

- La fiecare treapta:
  - primeste informatii de la vecin sau noteaza schimbarile in costul legaturii locale
  - proceseaza
  - posibil sa transmita noi informatii la vecinii adiacenti

Procesarea/comunicarea dintre entitasile nivelului retea!



cost pana la destinatie via

destinatia	D ()	A	B	C
A		1	14	5
B		7	8	5
C		6	9	4
D		4	11	2

Tabela de distanta:

- tabela per nod inregistrand costurile tuturor nodurilor prin fiecare din vecinii sai
- DE(A,B) da costul minim de la E la A *considerand ca primul nod al caii este B*
  - DE(A,B) = c(E,B) + min DB(A,*)
  - minDE(A,*) da costul minim al lui E catre A
  - tabela de rutare derivata din tabela de distanta
- exemplu: DE(A,B) = 14 (nota: nu 15!)
- exemplu: DE(C,D) = 4, DE(C,A) = 6

Algoritmul vectorului distanta

1. Bazat pe algoritmul Bellman-Ford
2. Folosit in multe protocoale de rutare: Internet BGP, ISO IDRP, Novell IPX, ARPAnet original.

Algoritm (in nodul X):

Initializare: pentru toate nodurile adiacente v:

D(*,v) = infinit

D(v,v) = c(X,v)

transmite costul caii cea mai scurta catre fiecare destinatie din vecinatate.

Bucla

Executa algoritmul de actualizare a topologiei distribuite.

Continuu

Algoritmul de utilizare a topologiei

in nodul X:

1. wait (until I see a link cost change to neighbor Y or until I receive update from neighbor W)

2. if (c(X,Y) changes by delta) {

/ change my cost to my neighbor Y */*

change all column-Y entries in distance table by delta if this changes my least cost path to Z

send update wrt Z, DX(Z,*) , to all neighbors

}

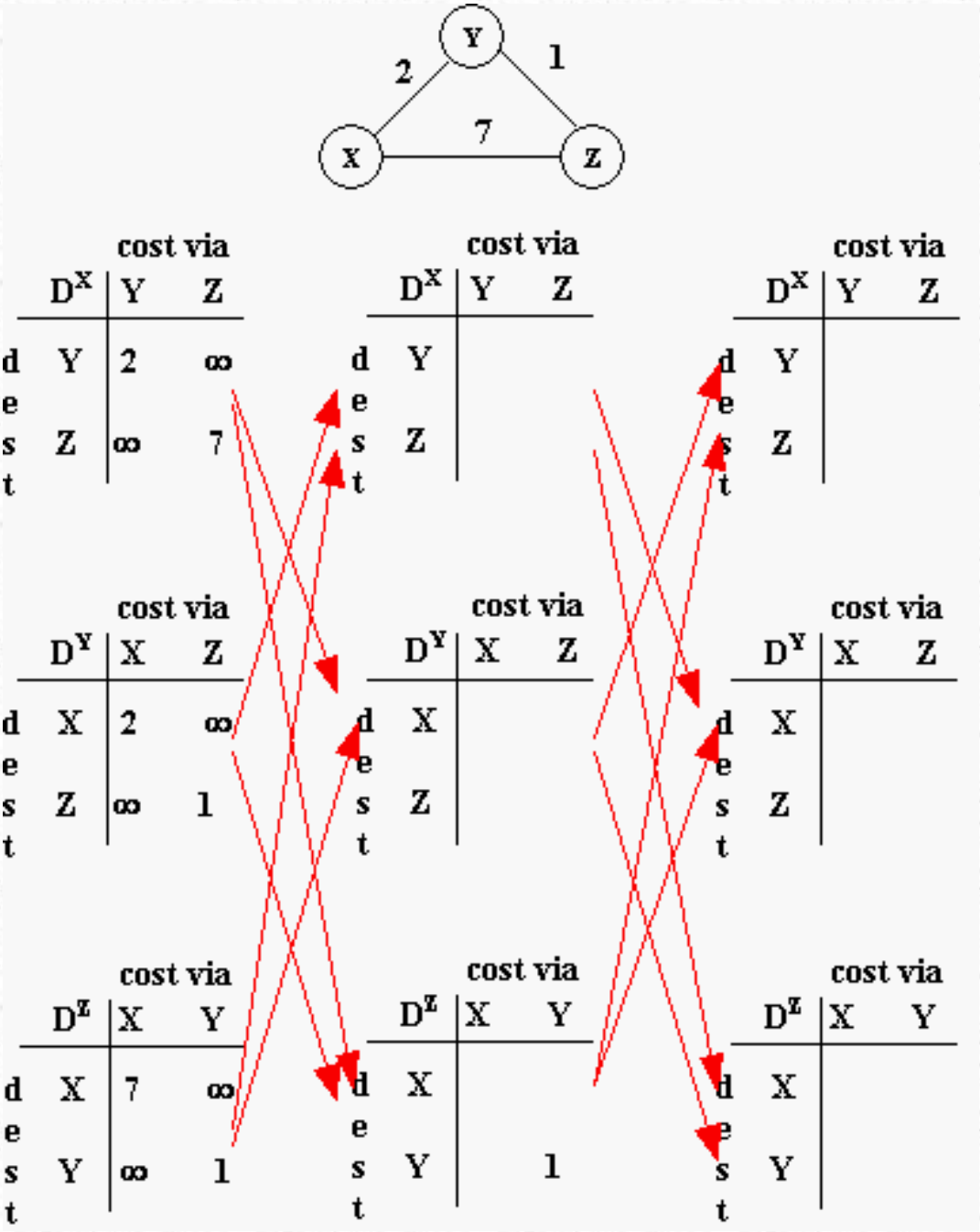
1. if (update received from W wrt Z) {

/ shortest path from W to some Z has changed */*

DX(Z,W) = c(X,W) + DW(Z,*)

if this changes my least cost path to Z send update wrt Z, DX(Z,*) , to all neighbors

Rutarea vectorului distanta: exemplu



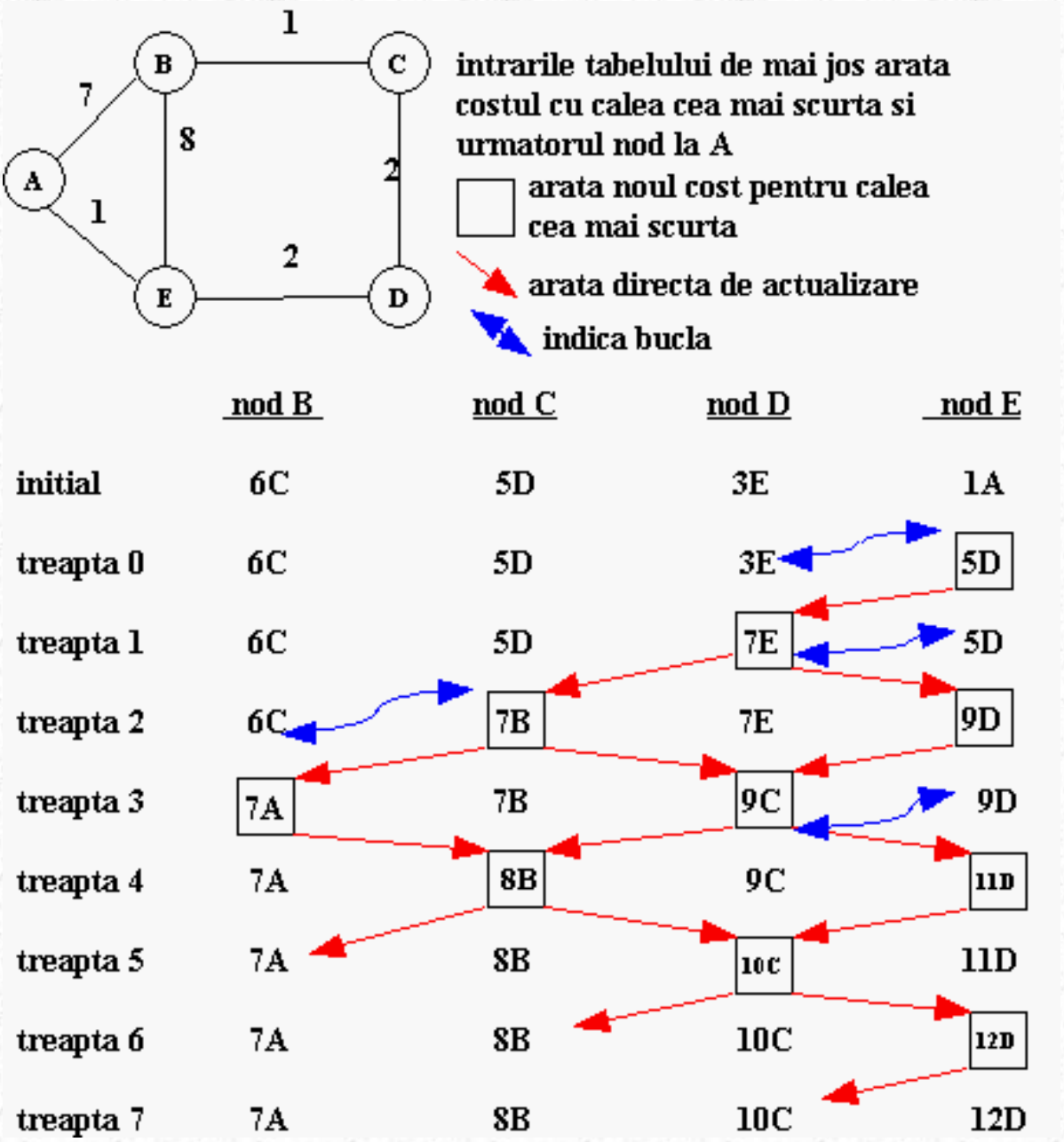
Rutarea vectorului distanta: recuperarea din legaturile defecte

Daca legatura XY pica, seteaza c(X,Y) la infinit si ruleaza algoritmul de actualizare a topologiei.

Bucla:

- tabele de rutare inconsistente: saa se ajunga la rutele A, D prin E, si la rutele E prin D
- buclele eventual dispar (dupa suficiente iteratii)
- buclele determina degradarea performantei, livrarea peste rand

Rutarea vectorului distanta: exemplu de recuperare

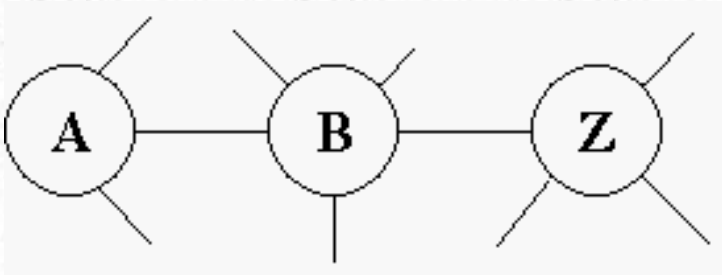


Rutarea vectorului distanta: rezolvarea problemei buclelor

Buclele vor exista in tabele pana cand valorile din tabel "calculeaza" costul rutei alternante.

Algoritm de splitare orizontala

- regula: daca rutele A trec prin Y via B atunci A ii spune lui B ca distanta sa pana la Y este infinita
- exemplu: B nu va trece niciodata traficul sau catre Y prin A



Oscilatii

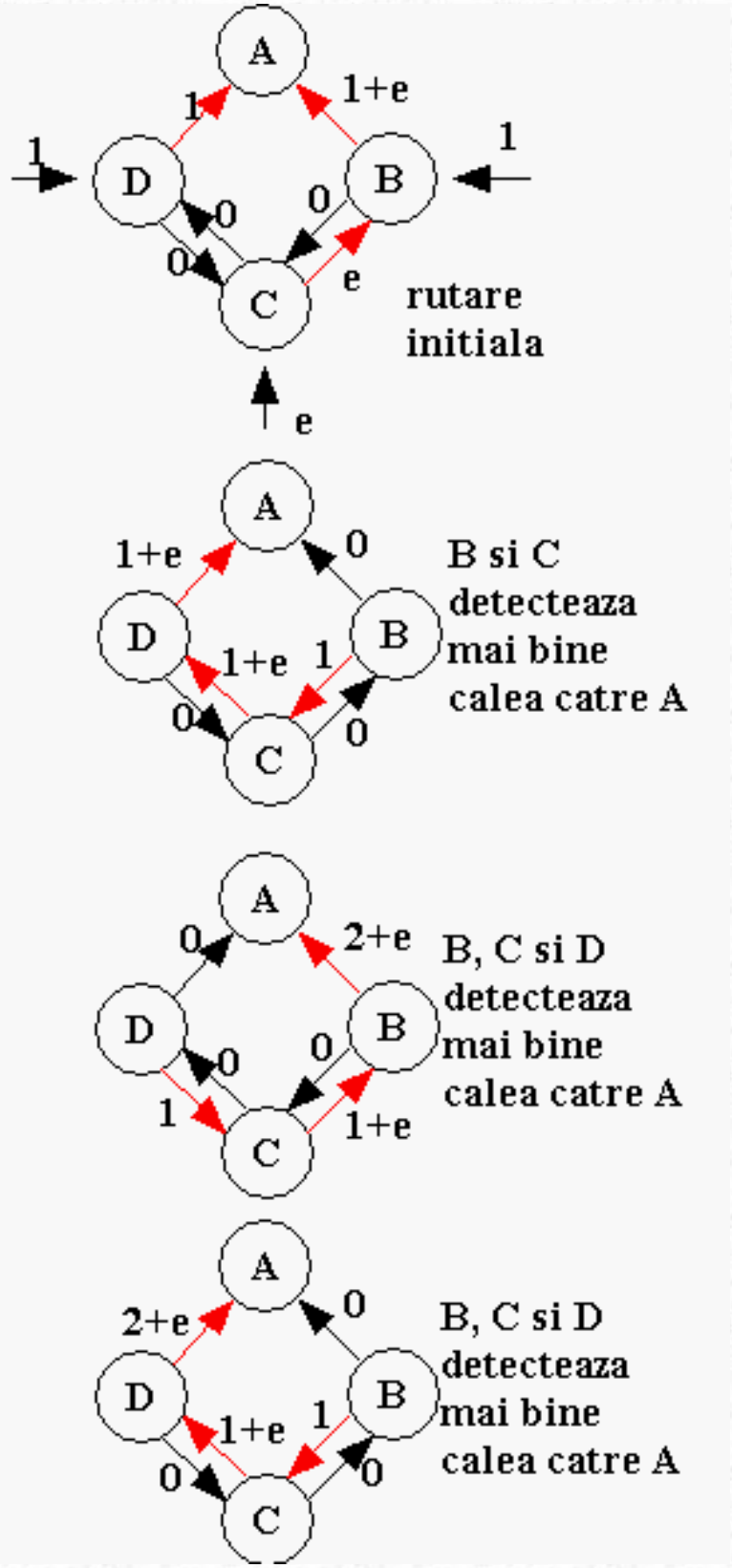
Un scenariu rezonabil

- costul legaturii depinde de cantitatea de trafic transportat
- nodurile schimba costurile legaturii la fiecare T
- presupunem:
  - A este destinatia pentru intreg traficul
  - B, D trimit 1 unitate de trafic la A
  - C trimite e unitati de trafic (e<<1) la A.

intreaga retea poate oscila. Solutii posibile:

- eliminarea schimburilor periodice
- costurile legaturilor nu trebuie sa fie functie de sarcina.

Oscilatiile vectorului distanta



Comparare intre algoritmii LS si DV

Complexitatea mesajului:

- "*LS este mai bun*": DV necesita iterarea cu schimbarea mesajului la fiecare iteratie
- "*DV este mai bun*": daca schimbarile legaturii nu afecteaza calea cu costul cel mai mic, nu se schimba niciun mesaj.

Robustetea: ce se intampla daca ruterul pica, nu se comporta cum trebuie sau este sabotat?

LS poate:

- sa raporteze distanta incorecta la vecinii conectati
- sa corupa/piarda oricare mesaje care trec
- sa raporteze vecinatati incorecte.

DV poate:

- sa avertizeze asupra costurilor pentru calea cea mai scurta la nici o/toate destinatiile.

Viteza de convergenta

DV:

- poate itera de mai multe ori in timp ce converge
- bucle, numarare la infinit, oscilatii
- nu poate propaga noi informatii pana cand nu recalculeaza propriile sale rute

LS:

- necesita 1 difuzare per nod per recalculare
- poate avea probleme din cauza oscilatiilor.

Amandoua au avantaje si dezavantaje

- una din ele va fi utilizata cu preponderenta in retea.

Rutarea difuzarii

Difuzarea: trimiterea unui pachet la toti cei N destinatari

- rutarea se actualizeaza in cazul LS
- avertisment privind serviciul/solicitarea in stratul aplicatie (de ex., Novell)

Algoritmul de difuzare 1: N transmiteri punct-la-punct

- trimite pachetul la fiecare destinatie, punct-la-punct.
- Pierdere a largimii de banda.
- Necesita cunoasterea tuturor destinatiilor.

Algoritmul de difuyare 2: flooding

- cand nodul primette un pachet de difuzare, il transmite la fiecare legatura
- nodul poate primi multe copii ale pachetului difuzat, deci trebuie sa poata detecta duplicatele

Rutarea difuzarii: directionarea caii inverse

Scop: eliminarea duplicatelor de flooding

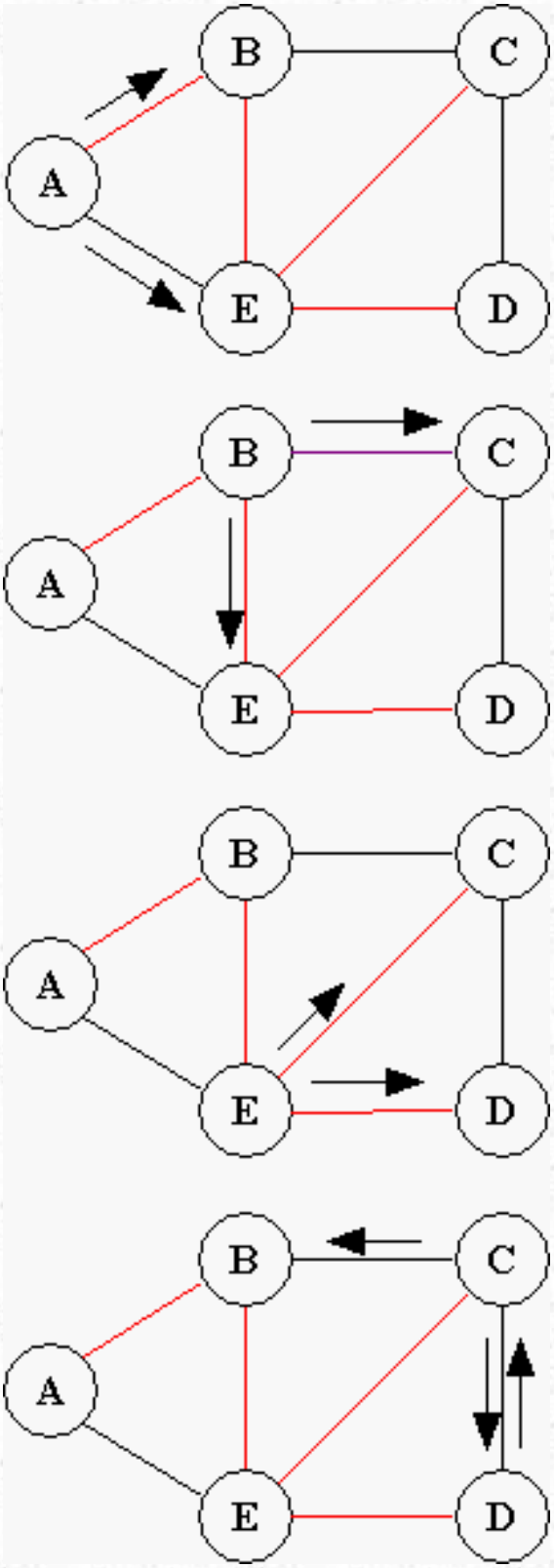
Presupuneri:

- A doreste sa difuzeze

- toate nodurile cunosc nodul predecesor pe calea cea mai scurta inapoi la A

Directionarea caii inverse: daca nodul primeste un pachet de difuzat

- daca pachetul soselte de la nodul predecesor pe calea cea mai scurta la A, atunci floodeaza totti vecinii
- altfel se ignora pachetul de difuzat - care a ajuns deja sau va ajunge de la nodul predecesor



Informatii despre rutarea distribuita

Va fi bun un algoritm de difuzare precum directionarea caii inverse pentru actualizari distribuite ale Starii de Legatura (in rutarea LS)?

Prima incercare (la distributia de difuzare a LS):

Fiecare ruter pastreaza o copie a celui mai recent pachet LS (LSP) primit din fiecare celalalt nod

Dupa primirea LSP(R) de la ruterul R:

- daca LSP(R) nu este identic cu copia stocata

atunci stocheaza LSP(R), actualizeaza informatiile LS pentru R, si floodeaza LSP(R)

altfel ignora duplicatul

A 2-a incercare (la distributie de difuzare LS):

Fiecare ruter pune un numar de secventa pe LSP proprii.

Dupa primirea LSP(R) de la R

- daca (seq # > seq # a copiei stocate) a LSP(R)

atunci stocheaza LSP(R), actualizeaza informatille LS pentru R, si floodeaza LSP(R)

altfel ignora duplicatul.

A 3-a incercare (la distributia de difuzare LS)

Foloseste secvente de numere "mari".

Adauga camp pentru "varsta" pe baza de timp

- fiecare ruter scade valoarea campului varstei cum LSP(R) rezida in memorie
- informasii de rutare pentru asteptarea locala (uitare) daca varsta este zero
- nu floodeaza
- pachet cu varsta zero.

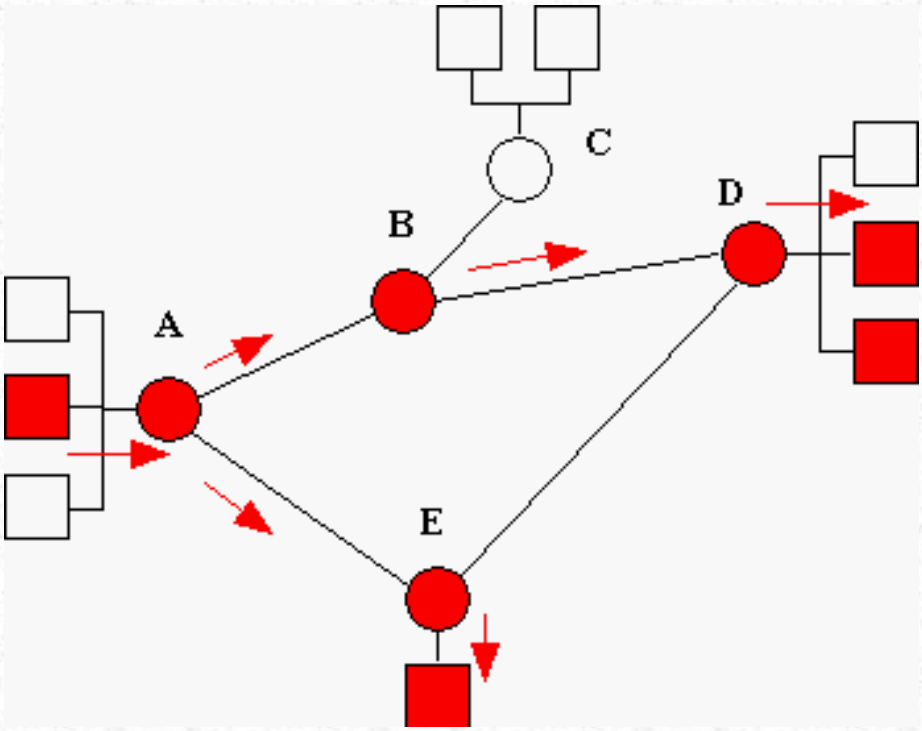
inlatura LSP(R) in asteptare (pentru transmisie) dar netransmise inainte de a flooda LSP(R) mai noi.

Rutarea spre mai multe statii

SCOP: livrarea pachetului de la un expeditor la mai mule gazde (dar nu toate)

- livrarea citre M gazde im N retele gazda (M<N)
- *optiunea 1:* expeditorul stabileste M conexiuni punct-la-punct

- *optiunea 2:* expeditoul trimite un pachet, care este duplicat si directionat, dupa cum este cerut de rutere:
  - ruterul A duplicheaza pachetul
  - ruterul B directioneaza *selectiv*.



Abstractizarea multistatii:

- adresa multistatie asociata cu grup multistasie
- gazde se alatura/parasesc grupul multistatie
- expeditorul trimite pachetul la adresa multistatie (destinatie)
- ruterele livreaza gazdelor care se alatura adresei de grup.

Rutarea multistatie: Starea legaturii

SCOP: rutarea pachetelor catre toate gazdele care se alatura adresei multistatie

Fiecare ruter foloseste floodingul starii de legatura pantru a distribui:

- costul catre vecini
- adresa multistatie care se doreste a fi primita (are gatda atasata gazda atasata acelui grup)

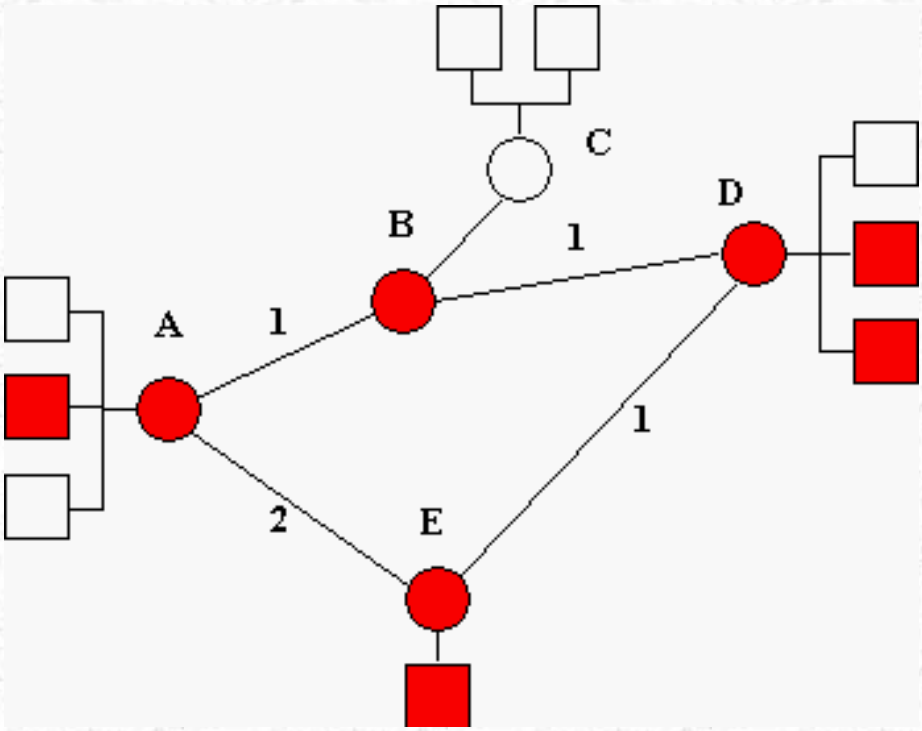
Fiecare ruter proceseaza:

- rutarea obisnuita unistatie (de ex., algoritmul Dijkstra)
- arbore director pentru M noduri unite la fiecare adresa activa multistatie
- *arbore director:* set de legaturi care conecteaza M noduri impreuna intr-un arbore.

Un posibil algoritm pentru arborele director:

- gaseste nodul cu "cea mai mica" adresa de gazda printre M

- foloseste algoritmul Dijkstra pentru a construi arborele, rutat la acea gazda care ajunge la toti M
- nota: arborele de mai jos nu este "costul minim"!



Rutarea multistatie: Vectorul distanta

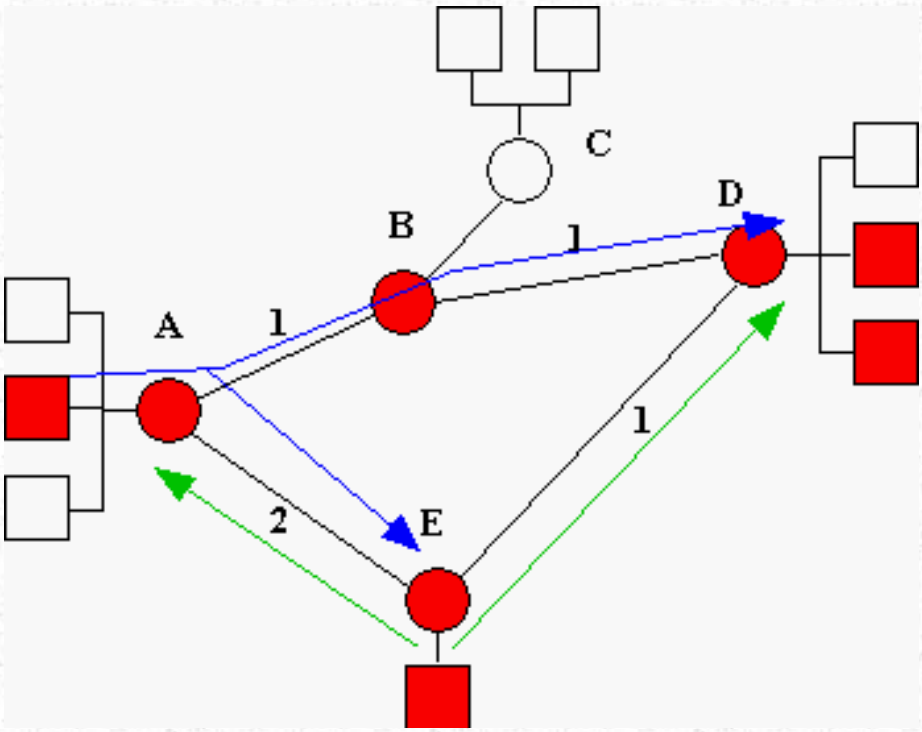
Pachetul soseste cu adresa multistatie *m*

- priveste la adresa expeditorului
- foloseste directionarea catre calea inversa pentru a flooda catre ruterele de jos care doresc pachete m-statie cu acea adresa

Cum stim daca ruterul "de jos" doreste pachet m-statie?

- gazda informeaza cel mai apropiat ruter despre adresele pe care le doreste
- ruterul X spune ruterului Y:
  - daca X doreste adresa m-statii si Y pe cea mai scurta cale a lui X la sursa, atunci Z doreste acea adresa

Rutarea multistatie: Exemplu DV



- E va directiona pachetele m-statii ale lui A catre D
- E nu va directiona pachetele m-statii ale lui A catre D
- A nu va directiona pachetele m.statii ale lui E catre B
- B nu va directiona niciun pachet m-statii catre C.

Rutarea ierarhica

Problema: dimensiunea cresterii retelei, tabela de rutare, cresterea complexitatii

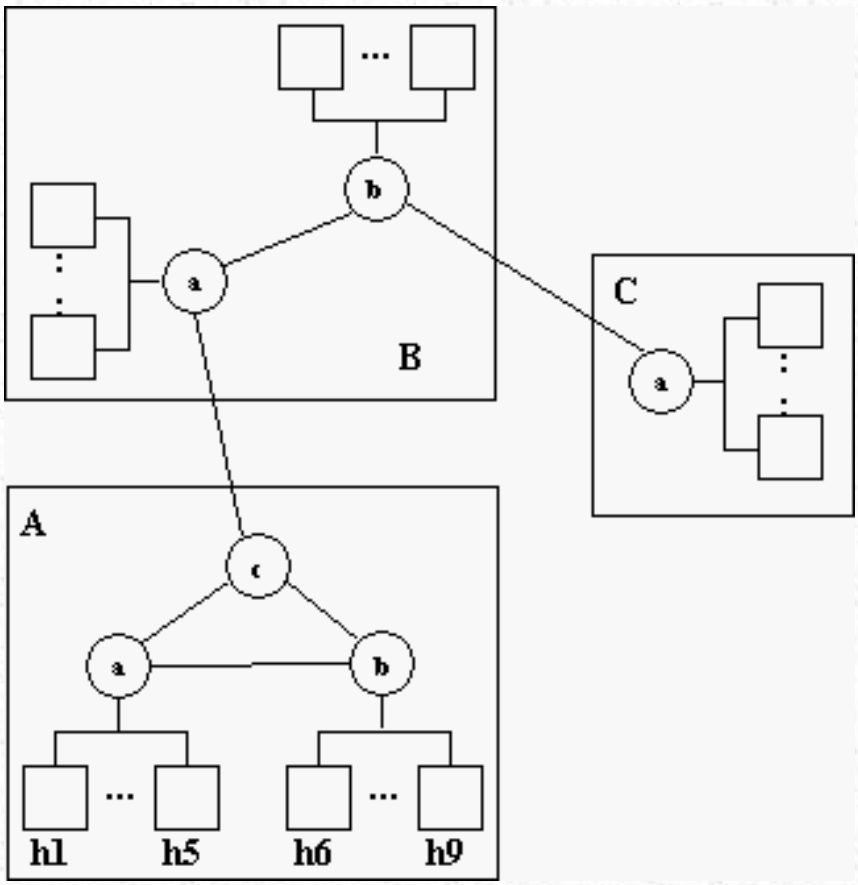
- milioane de noduri (gazde, rutere) in Internet.

Solutie: noduri agregate ierarhic in "regiuni" (domenii)

- nodul poseda cunoasterea deplina a ruterelor, structura topologica din interiorul regiunii
- una (sau mai multe) noduri din regiune responsabile pentru rutarea in afara

Teminologie:

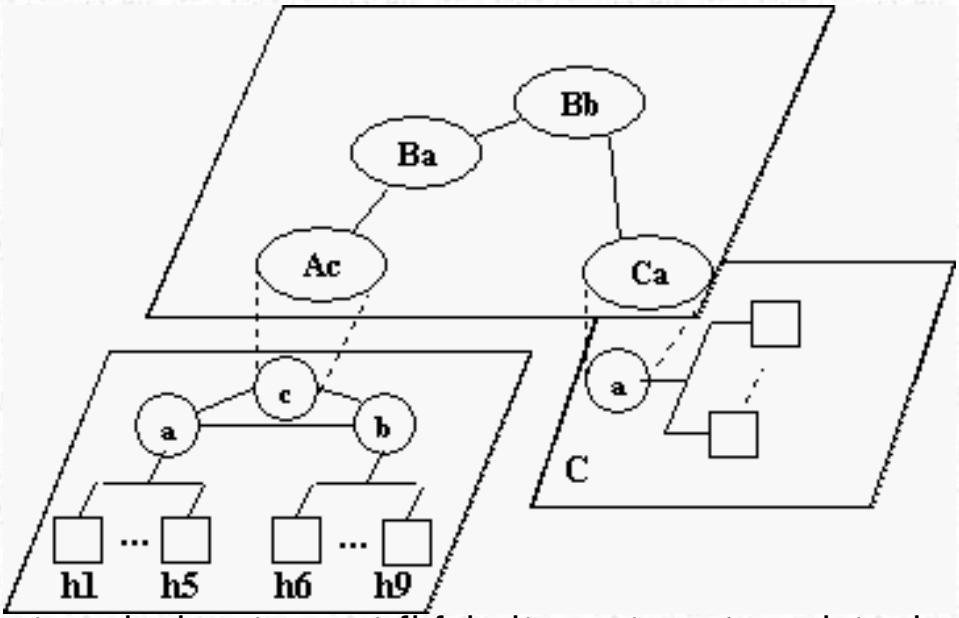
- rutare intradomeniu: din domeniu
- rutare interdomeniu: intre domenii
- sistem autonom (SA): domeniu, regiune, domeniu administrativ
- poarta: rute la/de la domeniu, respectiv ruter de margine.



Trei domenii: A, B, C

A.a, A.b A.c ruleaza protocol de rutare interdomeniu.

A.c, B.a, B.b, C.a ruleaza protocol de rutare intradomeniu printre ele insele.

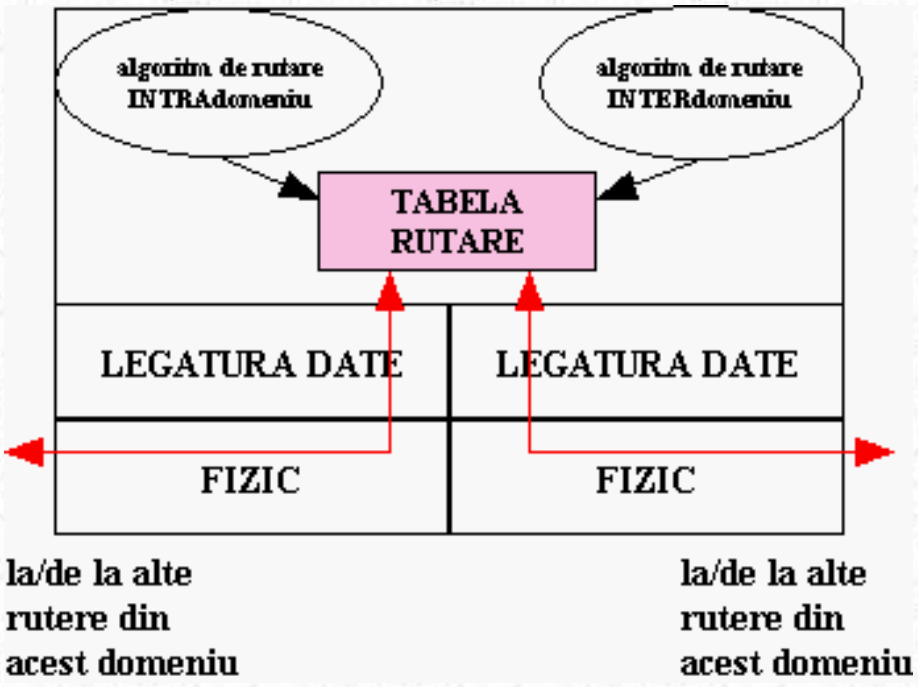


Diferite protocoale de rutare pot fi folosite pentru rutarea interdomeniului si intradomeniului.

A.o tabela de rutare:

destinatie	urmatorul salt
h6	A.b
...	A.b
h9	A.b
toti ceilalti (ruta obisnuita)	A.c

A priveste in interiorul A.c:



Gazde si rutere

Gazde (sisteme terminale) nu executa in mod normal nicio rutare

- porneste pachetele in directia lor
- trimite pachetele catre ruterul cel mai apropiat

Cum afla gazdele identitatea ruterului:

- A1: adresa IP a ruterului codat la nivel de hard in fisier (vezi /etc/networks pe multe sisteme UNIX)
- A2: descoperirea ruterului: RFC 1256
  - ruterul difuzeaza periodic existenta sa gazdelor atasate
  - gazda (la pronire) dsifuzeaza intrebarea (cine este ruterul meu) pe legaturile/LAN-urile atasate.

Studiu de caz al nivelului retea: Internetul

<---------- 32 biti ---------->

versiune	lungime header	tipul serviciului	lungimea pachetului (octetes)	
identificator 16 bit			steaguri	ofset de fragmentare 13 bit
timp de viata		protocol nivel superior	verificarea sumei pentru header	
sursa adresa IP 32 bit				
destinatie adresa IP 32 bit				
optiuni (daca exista)				
date				

Campuri in pachetul IP:

- *numarul pachetului:* (al protocolului IP), versiunea curenta este 4, noua versiune este 6
- *lungimea headerului:* datorita optiunilor, lungimea headerului este variabila
- *TOS*: nefolosit, idea a fost de a permite diferite nivele de siguranta, timp real, etc.
- lungimea pachetului: date plus header
- *identificator*: folosit cu fragmentare IP pentru a identifica fragmente apartinand aceluiasi pachet original IP
- *steaguri:* 2 biti: nu fragmenteaza, mai multe fragmente
- *ofset de fragmentare:* daca acesta este un fragment, apartinand pachetului original
- *timpul de viata:* decrementat de fiecare ruter, astfel incat un pachet nu va circula in bucla mereu in retea
- *protocol: care protocol de nivel superior demultiplexeaza si catre cine*. Vezi RFC 1700
- *verificarea de suma a headerului:* reprocesata la fiecare hop, cum TTL se schimba
- *sursa, adresa IP de destinatie:* a expeditorului original si eventual a destinatarului

Fragmentarea IP si Reasamblarea

Pachetul nivelului de transport poate fi prea mare pentru a transmite intr-un singur pachet.

Protocolul legaturii de baza va constrange lungimea maxima IP.

Fragmentarea: pachetul IP s-a divizat in *fragmente* de catre IP

- fiecare fragment devine propriul sau pachet IP
- fiecare adresa are acelasi identificator, sursa, adresa de destinatie
- ofsetul fragmentului da ofsetul datelor de la startul pachetului original
- bit pentru mai multe fragmente: 0 inseamna ultimul bit in acest fragment
- fragmente nereasamblate pana la destinatia finala.

	id	flg	offset	src	dest	data	
pachet original	X	0		Y	Z		
						0	8k
	id	flg	offset	src	dest	data	
apar doua fragmente	X	1	0	Y	Z		
						0	4k

id	flg	offset	src	dest	data	
X	0	4k	Y	Z		
					4k	8k

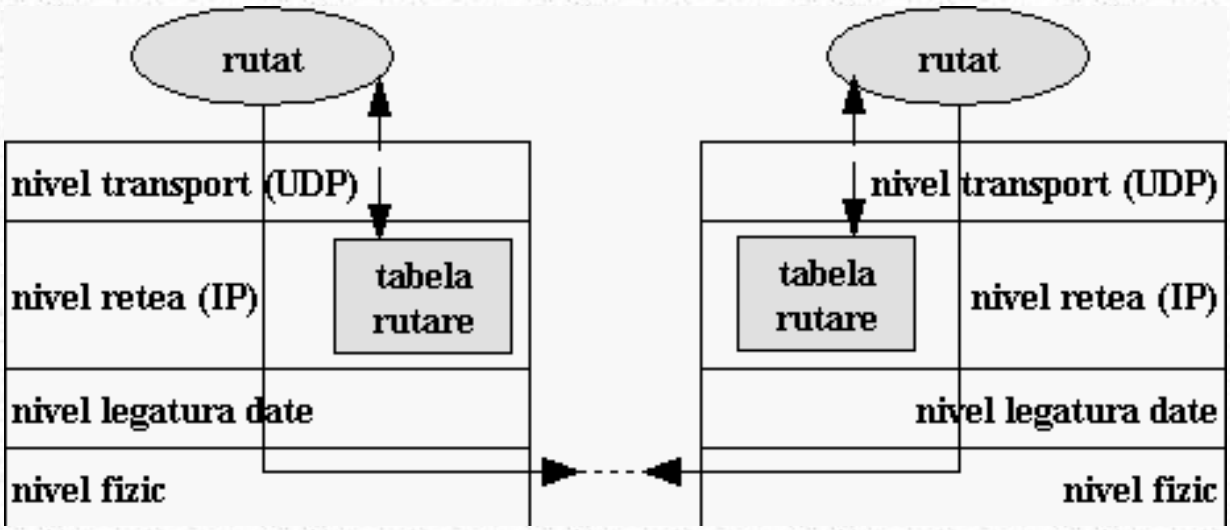
Rutarea intradomeniu a Internetului: RIP

RIP: Rutarea Protocolului de Informatii (Routing Information Protocol), foloseste algoritmul vectroului distanta, cu costuri de legatura 1

- cea mai scurta cale
- trimite tabela de rutare catre vecini fiecare 30 secunde, sau cand se schimba costul rutei.

Implementat ca demon (proces la nivel utilizator)

- comunica cu alt ruter atasat folosind pachete UDP
  - nota: apachetele UDP pot fi pierdute!
  - daca ruta in vecinatate nu este actualizata in trei minute, se asteapta rita (se seteaza costul la infinit)
- apelat rutat pe sistemele UNIX.



RIP-2

- suporta autentificarea (parola in clar)

O tabela de rutare RIP

Exemplu

Destinatie	Poarta	Steaguri	Refcnt	Utilizare	Interfata
127.0.0.1	127.0.0.1	UH	25	2260	lo0
default	128.119.40.254	UG	5	15223	ln0
128.119	128.119.40.195	U	28	188671	ln0

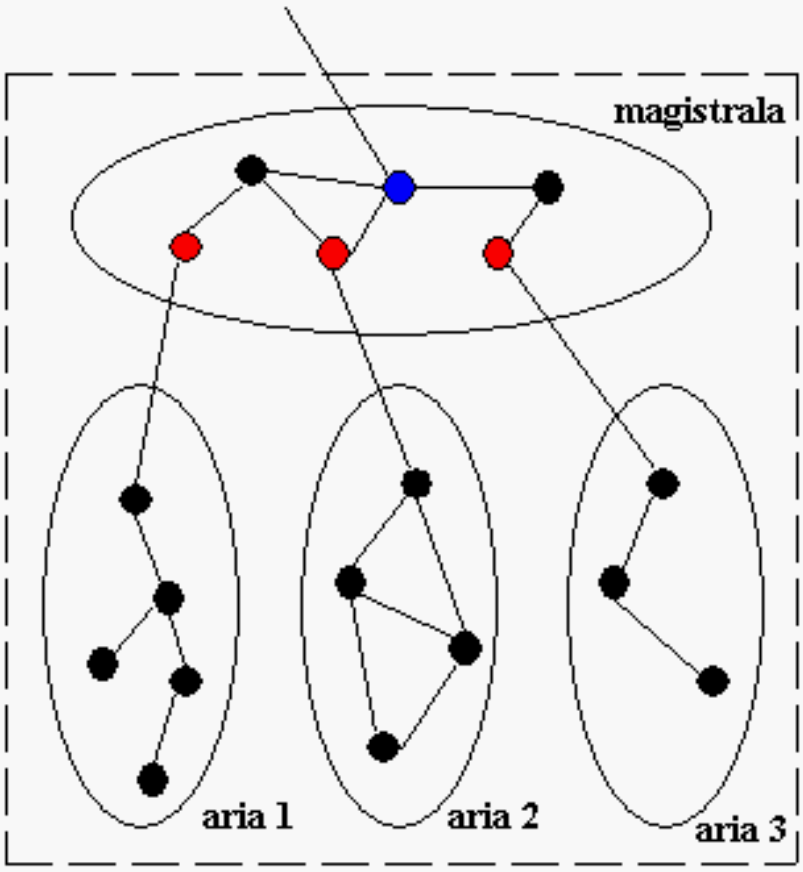
Rutarea intradomeniu a Internetului: OSPF

OSPF: deschide cea mai scurta cale in primul rand

- deschiderea: un standard publicat (RFC 1247)
- protocolul de poarta intern: pentru intradomeniu iesind dintr-un sistem autonom (SA)
- foloseste algoritmul starii de legatura pentru a determina rute
  - fiecare legatura (interfata) care iese are asignata un cost adimensional
  - diferite costuri pot fi folosite pentru diferite TOS
  - *echilibrarea sarcinii:* cu mai multe cai de cost egal la destinatie, va distribui sarcina peste amandoua cai.

OSFP: suport pentru ierarhie

- sistem autonom divizat in "arii"
- o arie designata drept "magistrala"
  - rutere de la marginea ariei in ruta magistrala dintre arii
  - exista si alte rutere in magistrala
- ruterul de la marginea SA comunica cu lumea din afara.



- ruterele de la suprafata: rosu
- ruterul de margine: albastru.

Rutarea intra-arie:

- niciodata nu intersecteaza magistrala

Pentru a trece de la o arie la alta:

- aria sursei-> magistrala -> destination area

Rutarea interdomeniu Internet: BGP

BGP: Border Gateway Protocol (Protocol de Poarta Marginala)

- rutarea intre noduri in diferite sisteme autonome (de ex., rutarea intre retele)
- RFC 1267, 1268
- foloseste o abordare a vectorului distanta.

Rutarea pe baza unei politici

- mai degraba decat costurile la destinatie, ruterele BGP schimba informatii despre calea completa (retele incrucisate) la destinatie
- ruterul poate decide asupra politicii pe care trebuie sa se decida ruta
  - de ex., "traficul de la SA al meu nu trebuie sa incruciseze SA a, b, c, d"

Implementarea BGP:

- Implementat ca demon (proces la nivel de utilizator)
- comunica cu alte rutere BGP folosind TCP

ICMP: Protocolul de Control al Mesajului Internet

Folosit pentru a comunica conditii de eroare la nivel retea si informatii catre protocoalele IP/TCP/UDP sau procesele utilizator.

Adesea considerat ca parte a lui IP, dar mesajele ICMP sunt transmise in datagramele IP.

IP demultiplexeaza pana la ICMP folosind campul de protocol IP.

Mesajul ICMP contine headerul IP si primii 8 octeti ai continutului IP care determmina generarea mesajului ICMP.

Mesaje ICMP selectate:

tip ICMP	cod	descriere
0	0	replica ecou (la ping)
3	0	reteaua de destinatie de neatins
3	1	gazda de destinatie de neatins
3	2	protocolul de destinatie de neatins
3	3	portul de destinatie de neatins
3	6	reteaua de destinatie necunoscuta
3	7	gazda de destinatie necunoscuta
4	0	incetinirea sursei (controlul congestiei)
8	0	solicitare de ecou
9	0	avertisment al ruterului

10	0	descoperirea ruterului
11	0	TTL expirat
12	0	header IP gresit

IPv6: urmatoarea generatie IP

Modificari la Ipv4:

- adrese de 128 bit (deci nu rulam in afara adreselor IP)
- simplificarea headerului (procesare mai rapida)
- suport mai mult pentru tipul serviciului
  - prioritati
  - identificator de trafic: identifica pachetele intr-o conexiune
- securitate

<---------- 32 biti ---------->		
versiune	**prioritate**	**nume trafic**
lungime payload	**urmatorul header**	**limita de salt**
adresa sursa (128 biti)		
adresa destinatie (128 biti)		
date		

Note:

- nicio fragmentare in retea
  - pachet prea mare genereaza erori ICMP la sursa
  - fragmentare sursa via header extensie
- nicio verificare suma (deja realizata la niveluri transport si legatura date)

Tranzitia de la IPv4 la IPv6

Internetul este prea mare pentru "marcare":

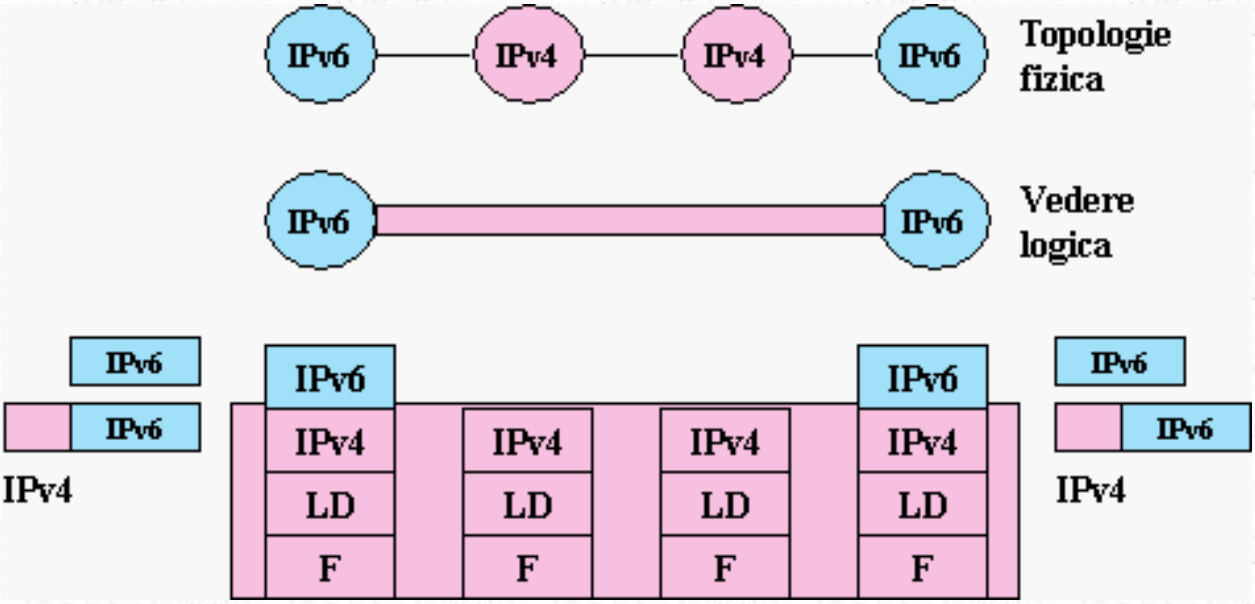
- nu se pot opri toate ruterele IP, instala IPv6 si apoi reporni
- nodurile IPv4 vor fi mostenite
- nodurile IPv6 pot ruta pachete IPv4
- nodurile IPv4 nu pot ruta pachete IPv6.

Tunelare:

- sursa si destinatia vorbesc comunica prin protocolul de retea X
- nodurile intermediare din punct de vedere fizic comunica prin protocolul de retea Y
  - sursele iau pachetul cu protocolul X, ii introduc (incapsuleaza) pachetul cu protocolul Y
  - nodurile intermediare ruteaza folosind protocolul Y
  - destinatorul primeste pachetul folosind protocolul Y, inlatura pachetul cu protocolul X

- *reteaua intre sursa si destinatie pare o simpla legatura la protocolul X*

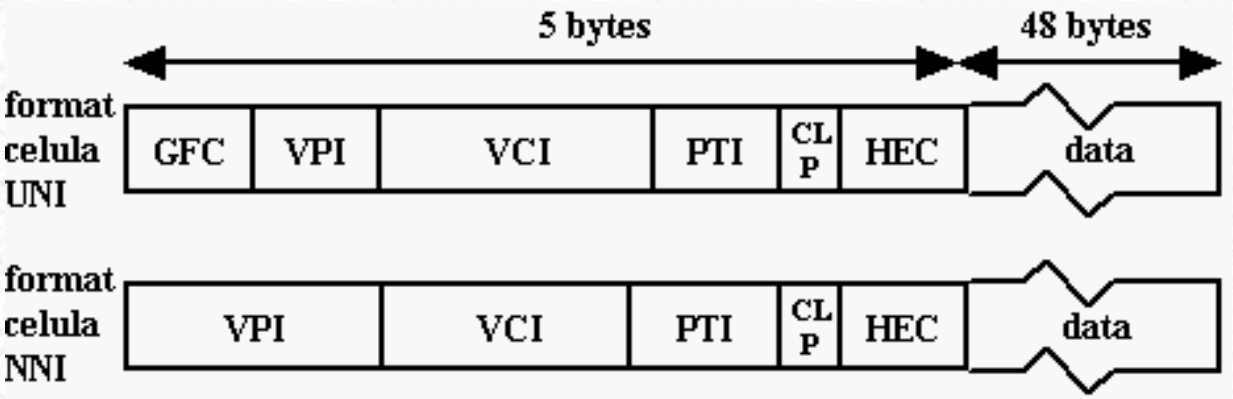
Imaginea tunelarii



Studiu de caz: nivel retea ATM

ATM: format de pachet (celula)

- UNI: interfata utilizator-retea (user-network interface) (de la gazda la comutator)
- NNI: interfata retea-retea (network-network interface) (comutator-comutator)



GFC: controlul generic al traficului (generic flow control) (nefolosit)

VPI: identificator cale virtuala (virtual path identifier)

VCI: identificator circuit virtual (virtual circuit identifer)

- VPI si VCI impreuna formeaza un identificator apel/conexiune

PTI: tip payload: 3 biti

- 111: celula RM (reapeleaza controlul congestiei RM)
- 000: celula utilizator
- 010: celula utilizator, experienta in congestie (reapel EFCI)

CLP: prioritate de pierdere a celulei (cell loss priority (1 bit))

- bit de prioritate pentru inlaturare

HEC: corectie de eroare a headerului (header error correction)

DATA: 48 octeti de date.

Observatii despre celula ATM

Foarte mica

- reflectand originile telefoniei
- 48 octeti este un compromis, media intre 64 si 32

Nu exista nicio adresa explicita de sursa

- folosite in schimb VCI/VPI
- comutare mai rapida (VPI/VCI poate indea in tabele)
- VPI/VCI de 28 bit pentru comutare in loc de adresa IP de 128 bit in IPv6 (economie)

Lungime fixa pentru comutare mai rapida.

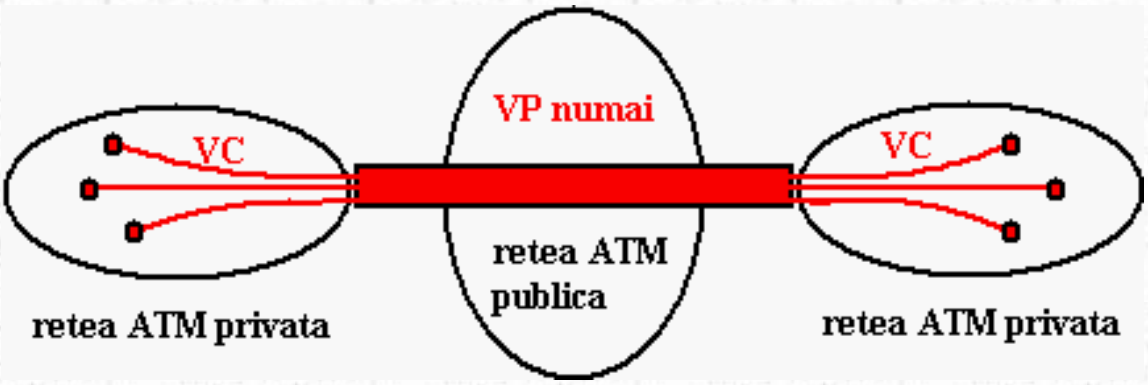
Prioritate minimala.

Retele ATM: orientate pe circuit virtual

VCI/VPI impreuna identifica apelul

Multiple apeluri (VCI) legate in acelasi VP

- reteaua poate comuta numai pe baza VP
- mai putine stari (reteaua vede numai VP)
- toate VC in VP urmeaza aceeasi cale.



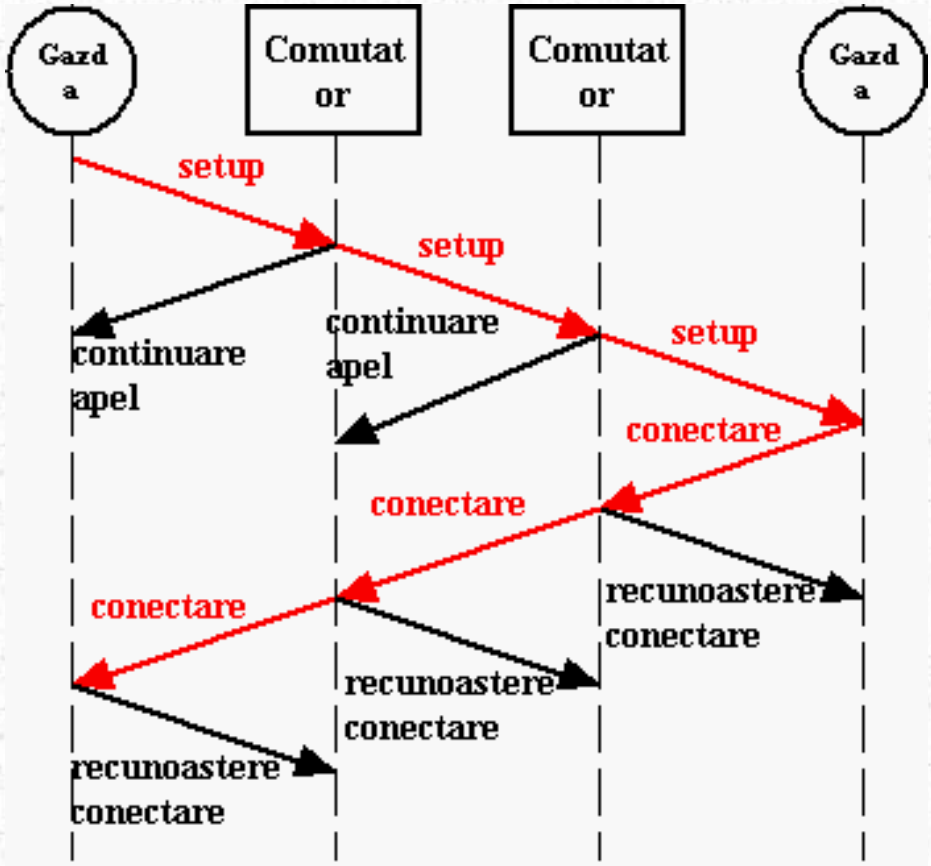
Setarea conexiunii in ATM

Mesajele ("semnalul") folosite pentru a seta apelul prin retea.

Informatii despre stare (despre comutarea VP - care linie de iesire sa comute VC care vine) se seteaza in comutatoare.

Semnificatia mesajelor de setare a apelului:

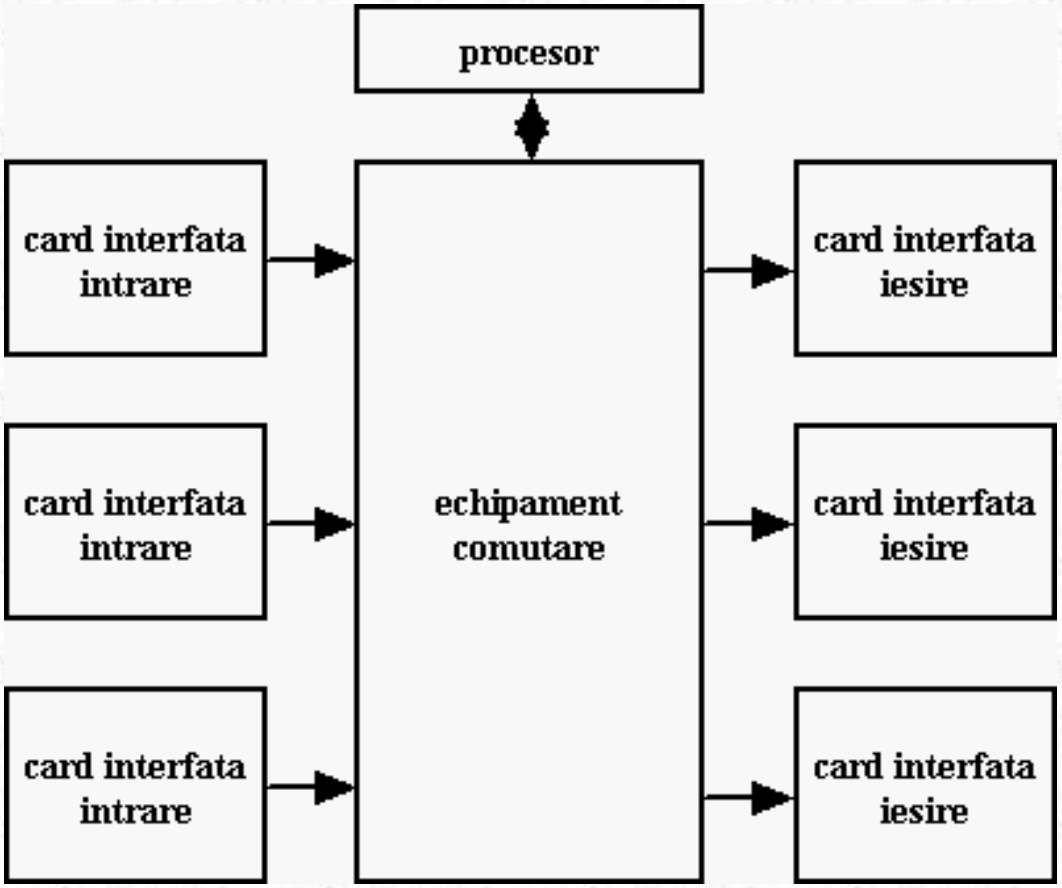
Mesaj	transmis de gazda retelei	primit de la retea
SETUP	stabileste conexiunea	apel care soseste
CALL PROCEEDING	gazda vede apelul	reteaua asteapta apelul
CONNECT	accept apelul sosit	apelul dvs. in asteptare
CONNECT ACK	conecteaza recunoastere primita	conecteaza recunoastere primita



Observatii:

- fata de Internet, comutatoarele sunt implicate in setarea apelului
  - crearea starii
  - recunoasterea intre comutatoare
- asteapta un RTT inainte de a transmite date
  - se deosebeste de UDP
  - la fel ca TCP
- daca conexiunea pica
  - alte comutatoare trebuie sa inlature starea
- standardul ATM nu specifica un protocol de rutina.

Comutatoare si rutere: introspectiva



Carduri de interfata la intrare:

- procesare nivel fizic
- bufereaza memoria pentru a pastra pachetul care soseste

Echipamentul de comutare:

- deplaseaza pachetele de la intrare la iesire

Carduri de interfata la iesire:

- bufereaza memoria pentru a pastra pachetele care pleaca
- proceseaza nivelul fizic

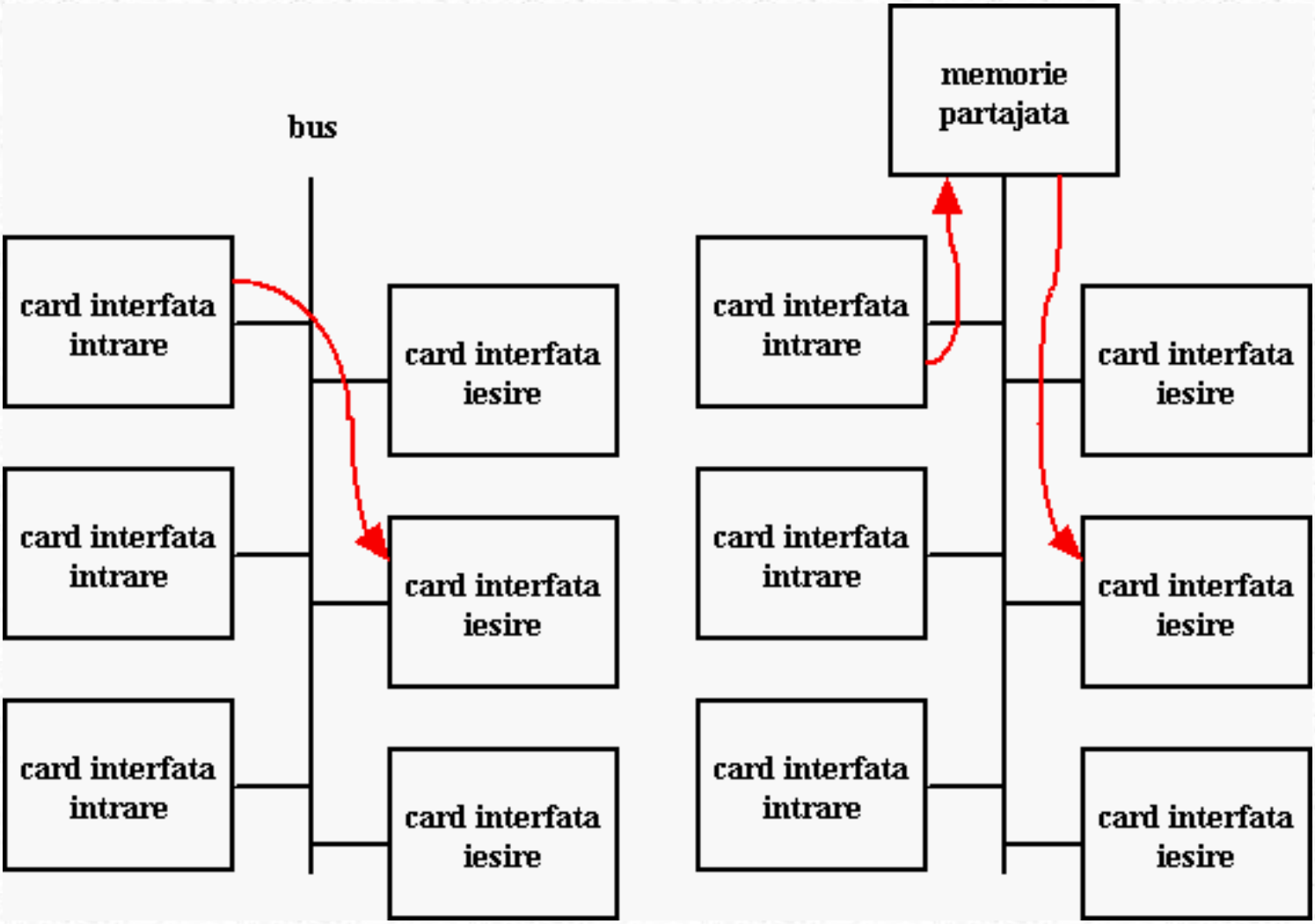
Procesor de control: actualizeaza tabela de rutare, functii (management) de supervizare

- in mod tipic nu se va atinge de pachetele care vor fi comutate

Echipamentul de comutare

Doua modalitati obisnuite de comutare:

- *comutare via memorie:* porturile liniei de intrare scriu in memorie, porturile de iesire citesc din memorie
- *comutare via o magistrala (bus):* magistrala backplane ("fund de sertar") conecteaza portile de intrare si iesire
  - de ex.: Cisco AGS+ are o magistrala backplane de 533 Mbps.



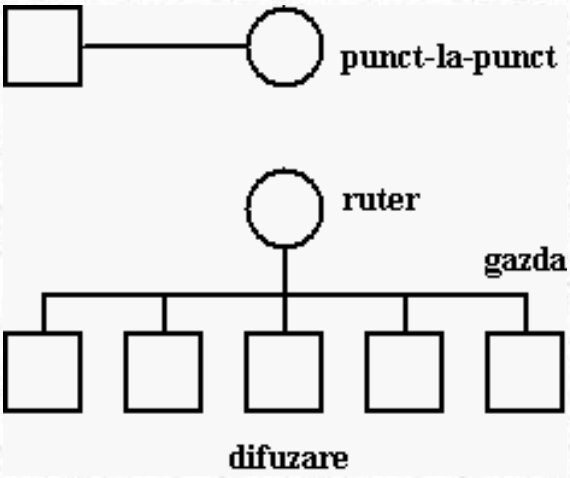
Back Top Next

7. Nivelul Legatura Date

Nivelul Legatura Date: Introducere

Servicii: livrare sigura a unui pachet de legatura date intre doua masini conectate fizic

Doua tipuri de legaturi: punct-la-punct, difuzare



Legaturi punct-la-punct: un expeditor, un destinatar

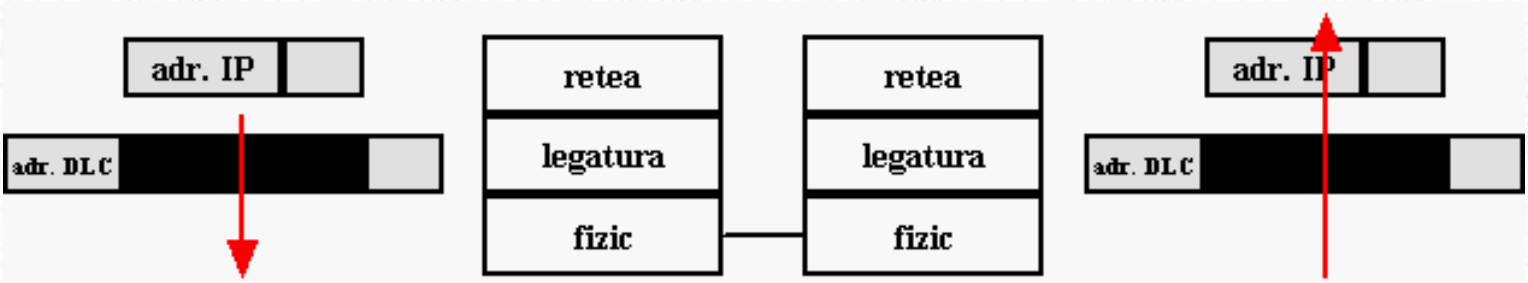
- cadru: recunoasterea bitilor pe fir ca pachete
- comunicatii sigure

Legaturi de difuzare: mai multi expeditori, mai multi destinatari potentiali

- incadrare
- comunicatii sigure
- accesarea unui mediu partajat
- adresare

Multe probleme de la nivele superioare apar si aici.

- comunicatii sigure: ARQ, verificarea sumei, timere, numere secventiale
- adresare
  - adresele nivelului legatura date sunt diferite de cele ale nivelului retea!



Control Legatura Date punct-la-punct

Foloseste aceleasi tehnici ca in nivelul transport

HDLC: high level data link protocol (protocol legatura date de inalt nivel)

- (este invechit)

Formatul cadru HDLC:

steag	adresa	control	date	verificare suma	steag
8 biti	8	8	arbitrar	16	8

- steagul (01111110) este folosit pentru a marca inceputul/sfarsitul cadrului
  - umplerea bitilor: daca sunt cinci de 1 consecutivi in date, expeditorul adauga un 0, pe care destinatarul il inlatura
- adresa nodului receptor (pentru legaturi de difuzare)

HDLC: campul de control

1	3	1	3		1		2	1	3
0	nr. secv.	P/F	nr. recun.		1	0	comanda	P/F	nr. recun
camp control: date					camp control: supraveghere				

Formatul campului de control pentru cadrele "data":

- numar secventa 3-bit
- numar recunoastere 3-bit
- P/F 1 bit pentru a indica expeditor-catre-destinatar sau invers.

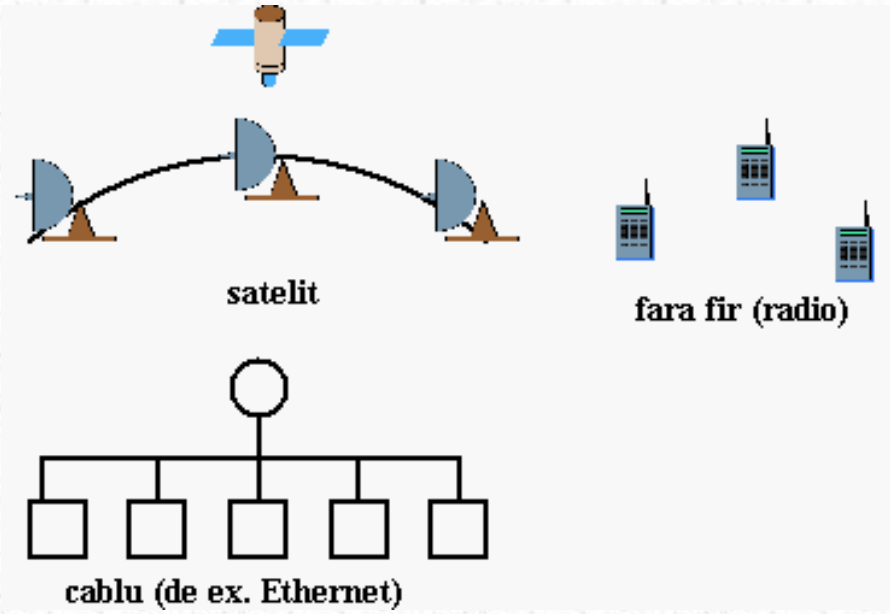
Formatul campului de control pentru cadre "supraveghere":

comanda	semnificatia
Receive Ready	recunoastere
receive not ready	controlul fluxului: nu e gata
reject	recunoastere negativa: retransmite inapoi la N
selective reject	recunoastere negativa: retransmite selectiv

Legaturi de difuzare: Protocoale de acces multiplu

Canal de comunicatii unic partajat

- doua sau mai multe transmisii simultane prin noduri: interferenta
- numai un nod poate transmite cu succes la un moment dat
- exemplu de medii cu acces multiplu:



Protocoale de acces multiplu

Algoritm distribuit care determina modul de partajare a canalului de catre statii, respectiv cand va transmite statia.

Comunicatiile cu canal partajat trebuie sa foloseasca insusi canalul!

La ce trebuie sa ne uitam an cazul protocoalelor cu acces multiplu:

- sunt sincrone sau asincrone
- informatiile necesare despre celelalte statii
- robustetea (de ex., erori de canal)
- performanta.

Cateva protocoale de acces multiplu

O taxonomie a protocoalelor de acces multiplu

Protocoale cu acces intamplator: statiile isi disputa canalul, coliziune (pot apare transmisii suprapuse):

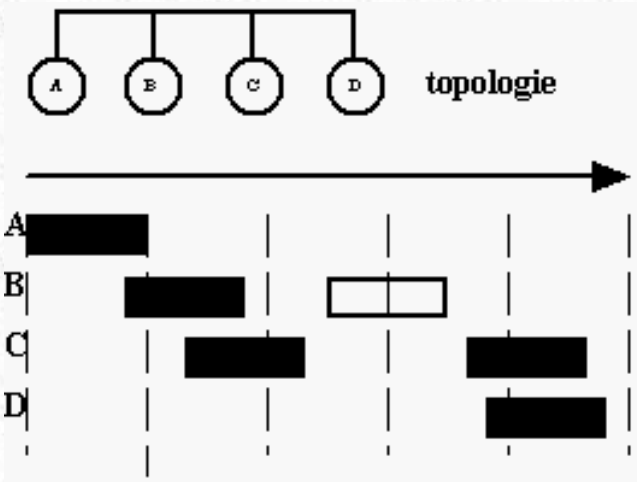
- Aloha
- Aloha discretizat
- expeditorul simte accesul multiplu: Ethernet
- acces intamplator la grup

Protocoale cu acces controlat: statiile rezerva sau au canal asignat, fara coliziune

- alocarea canalelor predeterminate: acces multiplu cu diviziune de timp
- alocarea adaptiva la cerere a canalului
  - protocoale de rezervare
  - trecerea jetoanelor (magistrala de jetoane, inel de jetoane)

Protocolul Aloha

- *simplu*: daca ai pachete de transmis, "transmite pur si simplu"
- daca pachetele sufera coliziune, se va incerca retransmiterea lor mai tarziu



Analiza protocolului Aloha

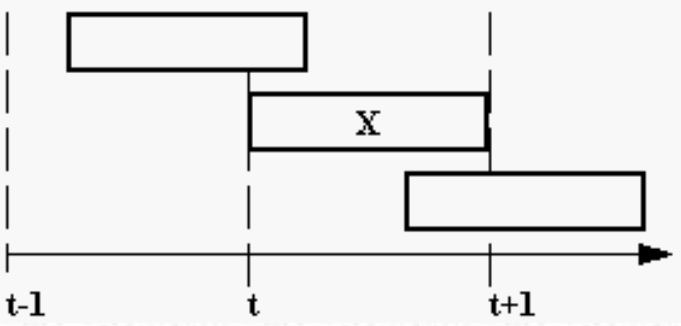
Scop: intelegerea cantitativa a performantelor protocolului Aloha

- pachete de lungime fixa
- timpul de transmisie a pachetului este unitatea de timp
- trec: S: numarul pachetelor transmise cu succes (fara coliziune) per unitatea de timp
  - in exemplul anterior, S = 0.2 pachete/unitatea de timp
- sarcina oferita: G: numarul de incercari de transmisie de pachete per unitatea de timp
  - nota: S<G, dar S depinde de G
  - modelul Poisson: probabilitatea a k incercari de transmisie in t unitati de timp:

Prob[k trans in t] = ((Gt)**k)(e**(-Gt))/(k!)

- capacitatea protocolului de acces multiplu: valoarea maxima a lui S peste toate valorile lui G.

Focalizare pe o incercare data de transmisie de pachet



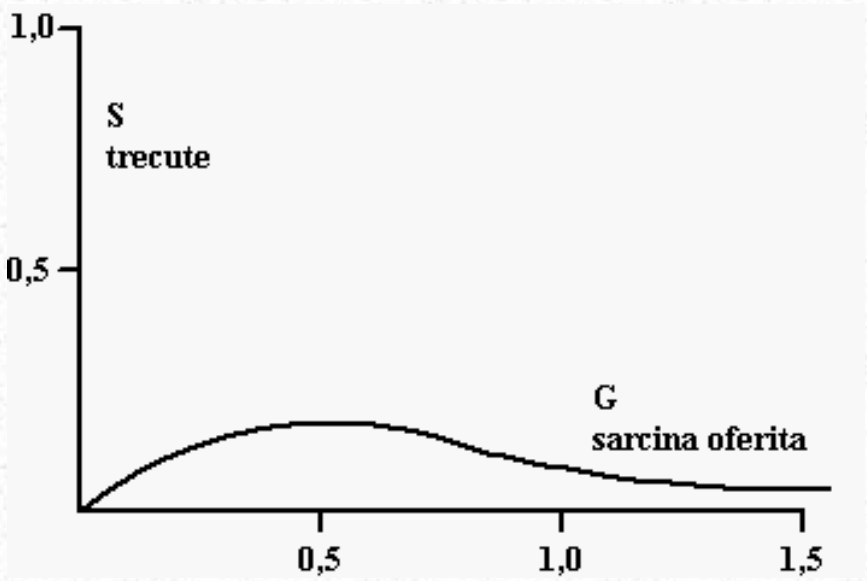
S = viteza incerarii de trans. a pach. * prob[trans reusite]

= G*prob[niciun alt pach nu se suprapune cu incercarea de trans]

= G*prob[0 alte incercari de trans in 2 unitati de timp]

= Ge*(-2G)

Treceri Aloha



Nota: trecerea maxima este 18% din capacitatea fizica a canalului

- cumperi o legatura de1 Mb, desi nu va fi niciodata mai buna de 180Kb!

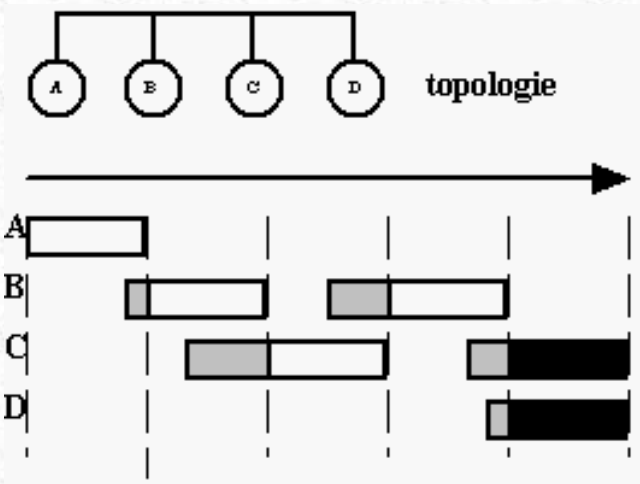
Aloha discret

Sistem sincron: timp divizat in sloturi.

Dimensiunea slotului egala cu timpul de transmisie a pachetului dat.

Cind pachetul este gata pentru transmisie, *asteapta* pana cand incepe *urmatorul* slot.

Pachetele se pot suprapune complet sau nu.



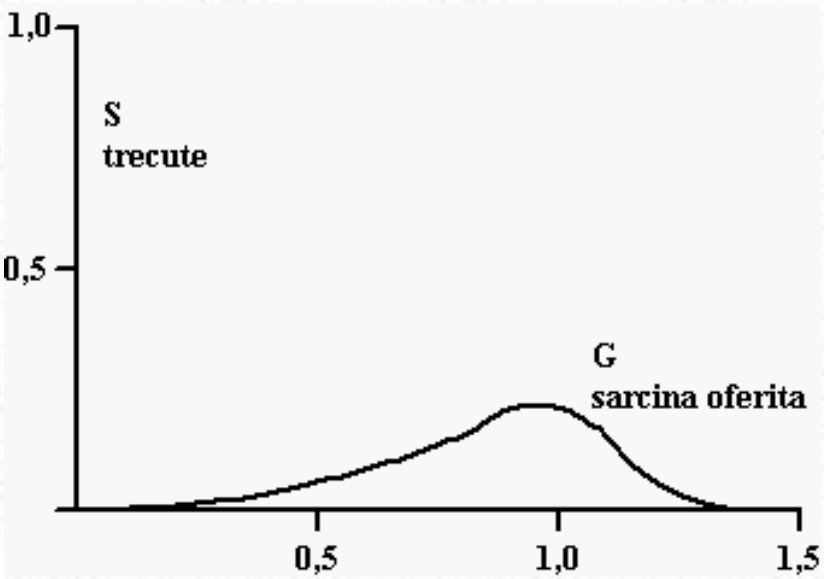
Performanta in cazul Aloha discret

S = G*prob[nicio alta transmisie sa nu se suprapuna]

= G*prob[0 alte incercari de transmisie]

= G*prob[0 alte sosiri in slotul anterior]

= Ge**(-G)



Protocoale de intelegere a expeditorului

- Aloha este ineficient (si prost crescut!): nu asculta inainte de a vorbi!
- Accesul multiplu cu intelegera expeditorului (Carrier Sense Multiple Access: CSMA)

CSMA ne-persistent:

1. intelege (asculta) canalul
2. if {canalul acsultat este ocupat}

then wait random time; go to 1

else transmit packet

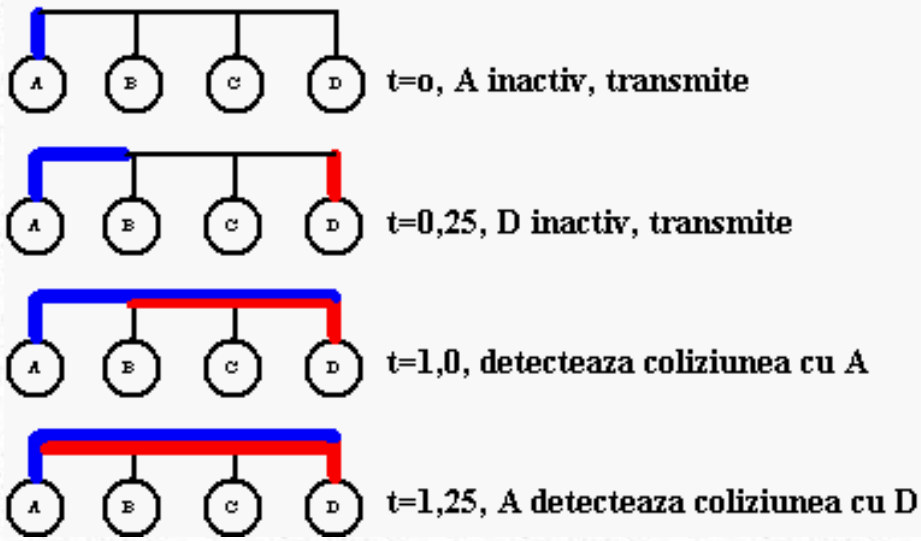
CSMA p-persistent:

1. intelege (asculta) canalul
2. when {channel sensed idle}

transmit with probability p

else wait random time, go to 1

Canalul respectiv nu va elimina toate coliziunile.



Performanta va depinde de lungimea canalului

- inarzieri mari de propagare: performanta slaba
- lungimea retelelor CSMA trebuie limitata.

CSMA/CD

CSMA cu detectia coliziunii (CD):

- asculta in timp ce vorbeste!
- opreste transmisiunea cand un alt pachet s+a ciocnit cu pachetul dvs.

Asteapta un timp arbitrar inainte de a incerca sa retransmita.

Performanta depinde (ca si in cazul CSMA) de lungimea canalului.

Studiu de caz: Ethernet

CSMA/CD, 1-persistent

Standardul IEEE 802.3

Canalul: cablu coaxial (de obicei)

T: intervalul arbitrar minim

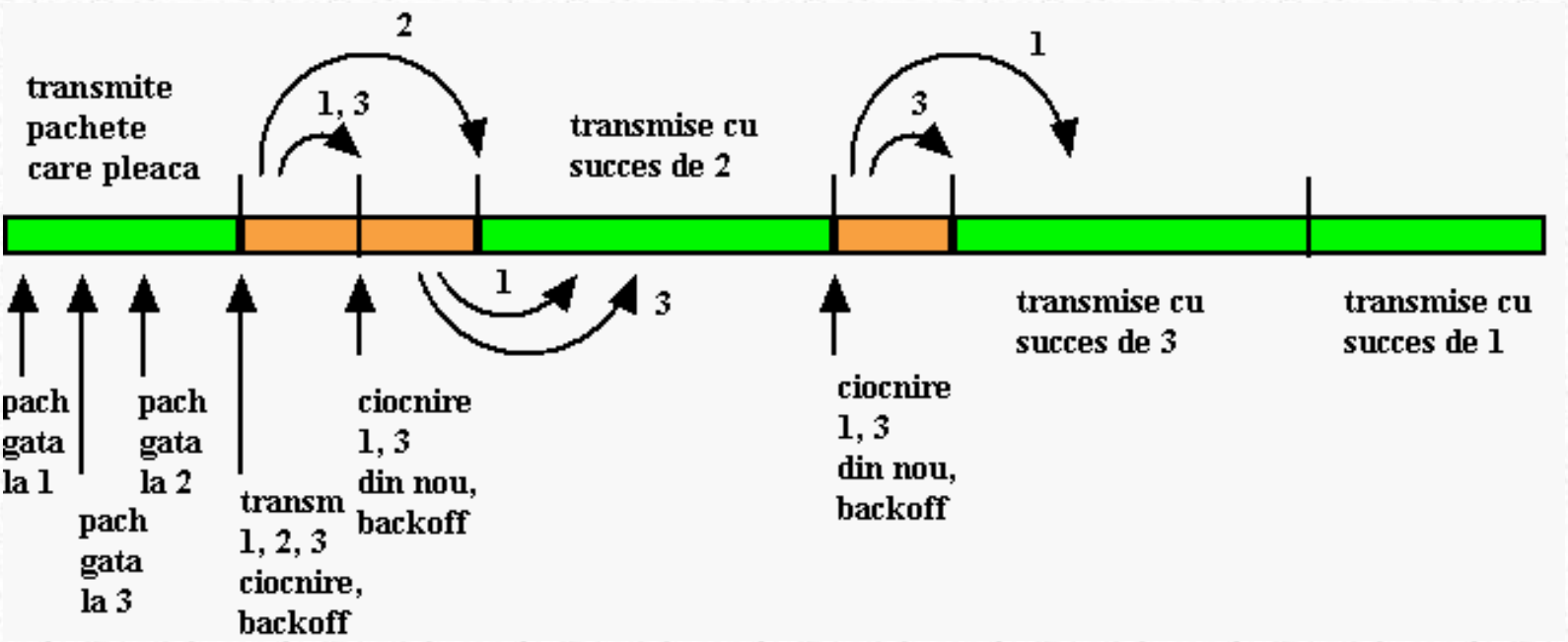
Rezolutia coliziunii: *binar:* pachetele sosesc (de la nivelul superior) pentru transmisie.

1. *Seteaza L=1, marcheaza pachetul ca "gata"*
2. *dupa transmisia reusita, toate gazdele cu pachete "gata" pot transmite*
3. *if {coliziune}*
   - *L=L*2, pana la 1024*
   - *asteapta cantitatea arbitrara de timp peste L*T unitati de timp*
   - *dupa asteptare, pachetul este din nou "gata"*
   - *go to 2*

Nota: intervalul de backoff este ajustat dinamic in functie de sarcina

- gazde diferite vor avea valori L diferite
- incarcare usoara: de obicei L mic
- incarcare grea: L mai mare.

Ethernet: exemplu



Mai multe despre ethernet

- Standarde 10 Mb/sec 100 Mb/sec
- Formatul pachetului:

preambul	cadru de start	adresa de destinatie	adresa sursei	lungime	date	tampon	verificare suma

- *preambul:* 7 biti care permit sincronizarea ceasurilor expeditor/destinatar

- *startul cadrului:* 1 octete, desemneaza startul cadrului (precum HDLC)
- *adresa de destinatie*:
  - "adresa fizica" de 48 bit
  - diferita de adreasa IP!!!!
  - fiecare placa Ethernet din lume are propria sa adresa unica la nivel de hard (asignata de IEEE sau vanzator)
  - adresa destinatarului pentru pachetele de difuzie: va fi primita de toate gazdele atasate LAN-ului
- *adresa sursei*: adresa fizica 48 bit
- *lungimea:* 2 octetes, lungimea maxima a pachetului in 1500 octetes
  - reapeleaza fragmentarea IP
- *data:* contine (este) un pachet (de ex., pachet IP) trimis in jos de la nivelul superior
- *tamponul:* utilizat pentru a asigura umplerea suplimentara a datelor > 46 octetes
- *verificarea sumei*.

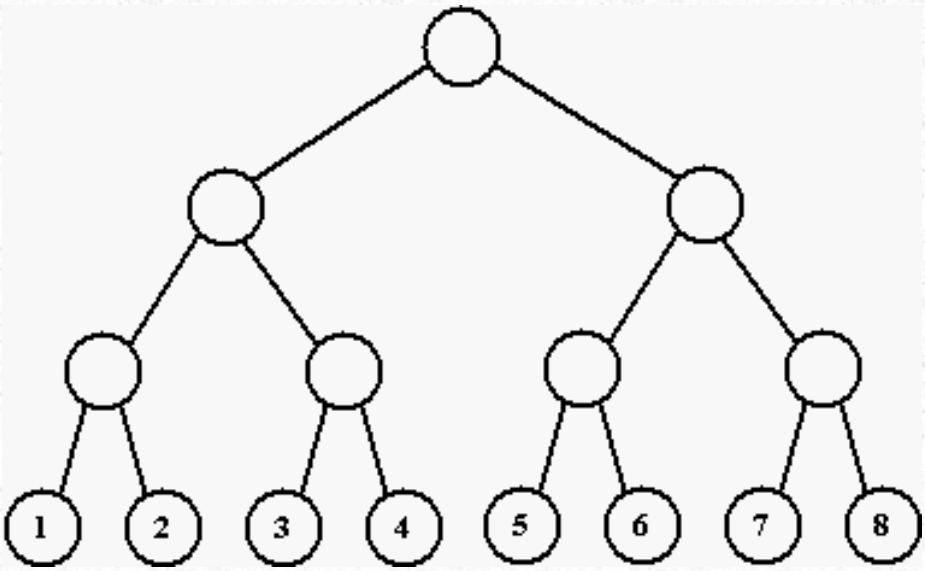
Protocoale de acces intamplator al grupului

Mai bune decat backofful arbitrar pentru a separa statiile care intra in coliziune, cautare structurata pentru exact o statie.

Permite grup de statii.

Daca apare coliziunea, divide grupul pana cand numai o statie pregatita este permisa.

Structura arborescenta.



1. toate statiile rutate la nodul de rutare disponibile
2. if {nicio statie nu transmite)

return

else if (one station sends)

return

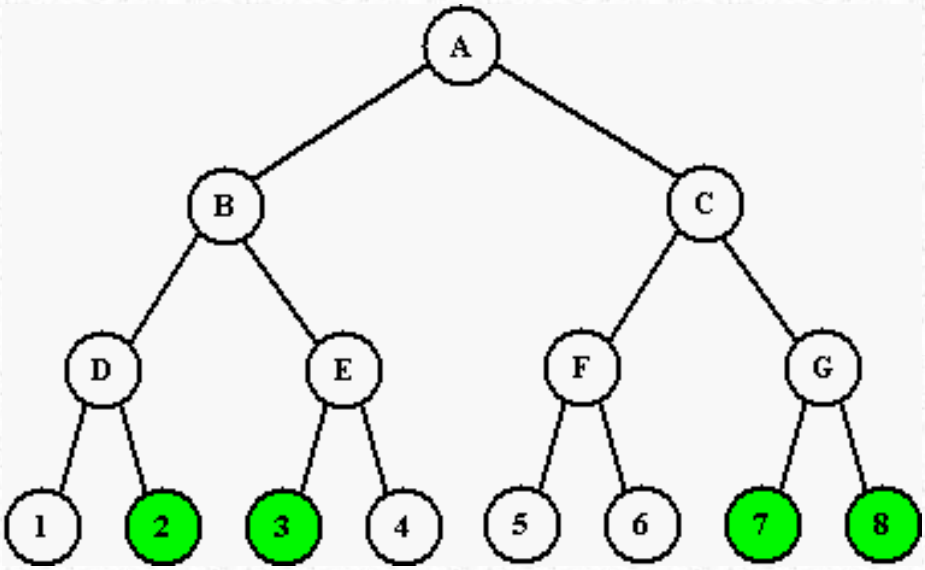
else /* collision */

resolve(leftchild(rootnode))

resolve(rightchild(rootnode))

Acces intamplator in grup: exemplu

Presupunem statiile 2, 3, 7, 8 gata cu pachetele



A disponibil, coliziuni

B disponibil, coliziuni

D disponibil, SUCCES cu 2

E disponibil SUCCES cu 2

C disponibil, coliziuni

F disponibil, inactiv

G disponibil, coliziuni)puteau fi evitate!)

7 disponibil, SUCCES

8 disponibil, SUCCES

Protocoale de trecere a jetoanelor

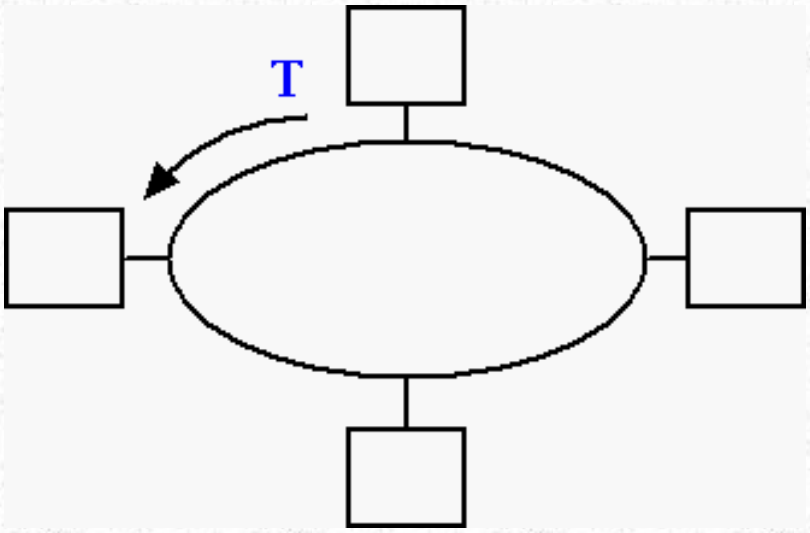
Jetoanele circula printre statii

Media:

- conexiunea inel jetoane: IEEE802.5, FDDI
- magistrala jetoane, IEEE802.4

Pentru a transmite

- statia trebuie sa cpatureze jetonul
- transmite pachetul in timp ce pastreaza jetonul
- elibereaza (trimite in afara) jetonul



Trecerea jetoanelor: Standardul IEEE802.5

4 Mbps

Timpul maxim de pastrare a jetoanelor: 10 ms, limitand lungimea pachetului

Formatul pachetului (jeton, data):

SD	AC	FC						
SD	**AC**	**FC**	**adr dest**	**adr caut**	**data**	**verif suma**	**ED**	**FS**

- *SD, ED* marcheaza startul, sfarsitul pachetului
- *AC octete de control al accesului*:
  - *bit de jeton*: valoarea 0 inseamna ca jetonul poate fi capturat, valoarea 1 inseamna ca datele urmaresc FC
  - biti de prioritate: prioritatea pachetului
  - *biti de rezervare:* statia poate scrie acesti biti pentru a preveni statiile cu prioritate mai mica a pachetului de a captura jetonul dupa acesta este eliberat
- *FC: controlul cadrului* folosit pentru monitorizare si mentinere
- *sursa, adresa de destinatie:* adresa fizica de 48 bit, ca in Ethernet
- *data:* pachet din nivelul retea
- *checksum*
- *FS: statutul cadrului (frame status):* setat de destinatar, citit de expeditor
  - setat pentru a indica ca destinatia este in regula, pachetele copiate de la inel sunt OK
  - recunoasterea nivelului DLC.

Acces multiplu prin diviziunea timpului: protocol de rezervare

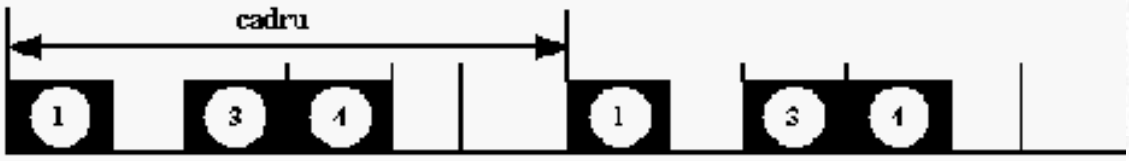
TDMA: acces multiplu prin divizarea timpului (time division multiple access)

Accesul la canal in "runde".

Fiecare statie obtine un slot de lungime fixa (timpul de transmisie a pachetului) in fiecare runda

Sloturile nefolosite ajung inactive.

Exemplu: LAN cu 6 statii, 1, 3, 4 au pachete, 2, 5, 6 sunt inactive.



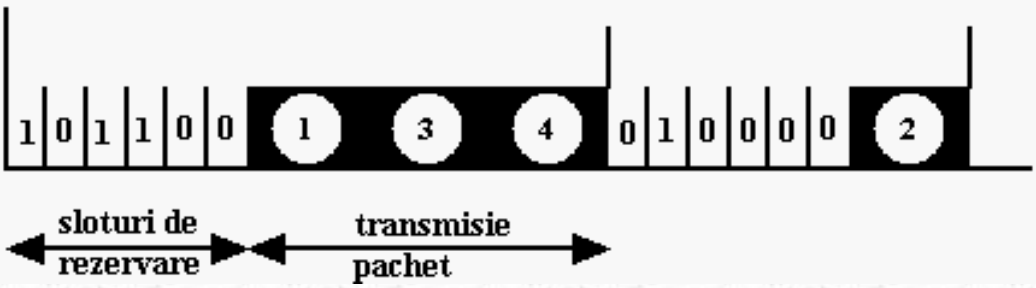
Protocoale pe baza de rezervare

Se doreste eliminarea sloturilor uzate din TDMA.

Accesul la canale in runde (iar). Fiecare runda incepe cu N *scurte sloturi de rezervare*.

- timpul slotului de rezervare este egal cu intarzierea canalului datorita propagarii cap-cap.
- statia cu mesajul de transmis posteaza rezervarea (1) in slotul sau de rezervare
- sloturile de rezervare sunt vazute de toate statiile.

Dupa rezervarea sloturilor, transmisiile mesajelor este ordonata prin prioritati cunoscute.



Evaluarea critica a protocoalelor cu acces multiplu

Acces intamplator: Alohas, CSMA, grup

Controlat, predeterminat: TDMA

Adaptiv cu cerere controlata: jetoane, rezervare

ARP: Protocolul de rezolutie a adresei (Address Resolution Protocol)

IEEE802.* (Ethernet, inel/magistrala de jeton) cardurile de interfata recunosc numai adresele nivelelului *fizic* de 48 bit IEEE 802 pe pachete.

Nivelul retea foloseste adresa IP (32 bits).

Cum se determina adresa fizica a masinii cu adresa IP data?

ARP:

- A cunoaste adresa IP a lui B, vrea sa afle adresa fizica a lui B
- A difuzeaza pachetul de solicitare ARP, continand adresa IP a lui B
- toate masinile de pe LAN primesc solicitarea ARP
- B primeste pachetul ARP, raspunde lui A cu adresa nivelului fizic (a lui B)

- A tine in cache (salveaza) perechile de adrese IP - fizica pana cand informatia devine perimata (s-a scurs timpul
  - starea softului: informatii care au expirat

Rutarea si adresele nivelului fizic: sinteze

Gazda IP A stie ca ruterul R este urmatorul hop catre destinatia IP B:

A creaza pachetul IP cu sursa A, destinasia B.

A foloseste ARP pentru a obtine adresa fizica a lui R.

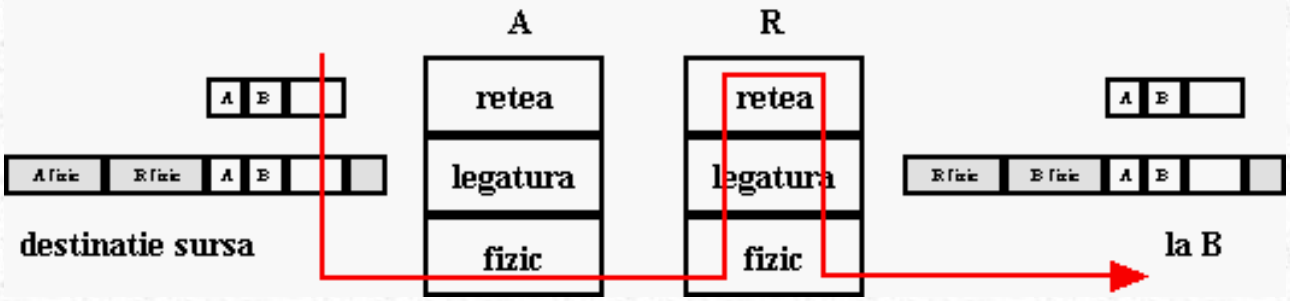
A creaza pachetul Ethernet cu adresa fizica a lui R ca destinatar, pachetul Ethernet contine pachetul IP A-la-B.

A trimite pachetul Ethernet.

R primeste pachetul Ethernet.

R inlatura datagrama IP din pachetul Ethernet, vede ca este destinat lui B.

R creaza pachetul la nivel fizic, continand datagrama A-la-B si trimite catre urmatorul ruter pe ruta catre B.



Interconectarea LAN-urilor

De ce nu un LAN mai mare?

- cantitatea de trafic suportabil este limitata: pe un singur LAN, toate statiile trebuie sa imparta largimea de banda
- lungime limitata: 802.3 specifica lungimea maxima a cablului
- numar de statii limitate: 802.4/5considera intarzieri la trecerea jetonului la fiecare statie.

Punti versus Repetoare la interconectarea LAN-urilor

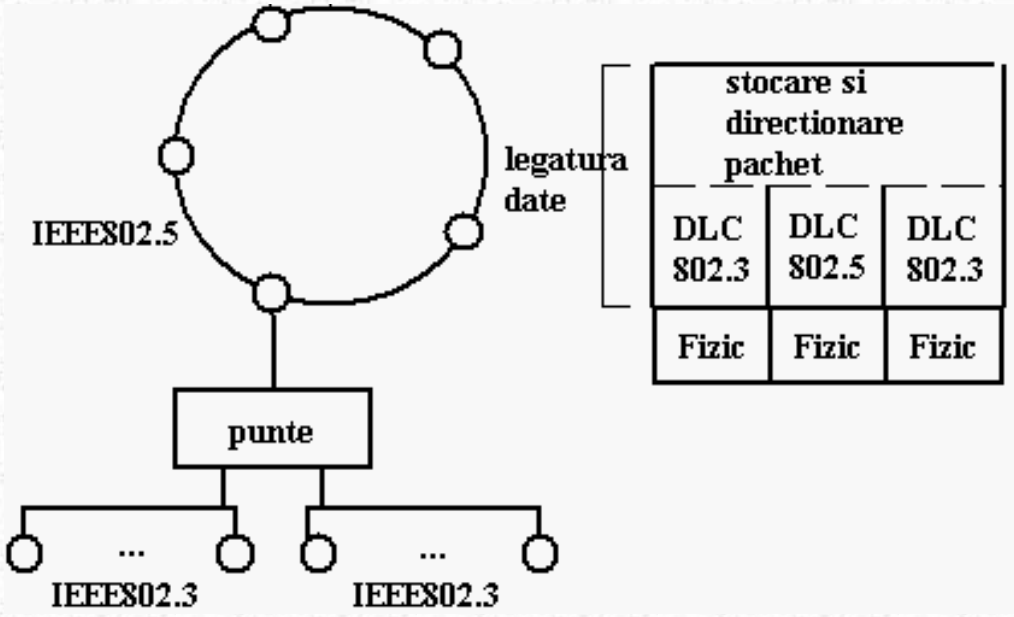
Repetoare

- copiaza (amplifica, regenereaza) bitii intre segmentele LAN-urilor
- nicio stocare de pachete
- interconexiune (numai) la nivel fizic al LAN-urilor.

Punte

- primeste, stocheaza (cand este nevoie) pachete intre LAN-uri
- are doua nivele pentru protocol: fizic si legatura (acces media).

Punti versus rutere



Puntile sunt, intr-un fel, rutere

- vor sti adresele de nivel fizic ale statiilor pe fiecare LAN interconectat
- primesc pachetele transmise pe LAN si le directioneaza *selectiv*

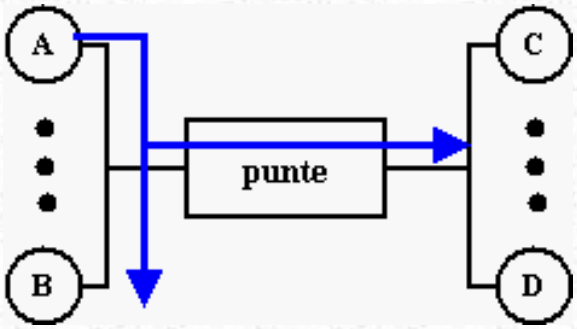
Puntile nu sunt rutere

- nu stiu nimic despre "lumea exterioara", numai statiile pe LAN-urile interconectate
- puntile nu schimba tabele de rutare
- au de-a face numai cu adresele la nivel fizic.

Punti: Directionare pachete

Puntile *filtreaza* pachetele

- pachetele pe segmentul intra-LAN sunt directionate in alte segmente de LAN
- pachetele din segmentul inter-LAN trebuiesc directionate in alta parte



Tehnici de directionare a pachetelor

- pachete flood (un inconvenient evident)
- protocol asemanator cu deascoperirea ruterului
  - permite puntii sa identifice gazdele pe segmentul LAN

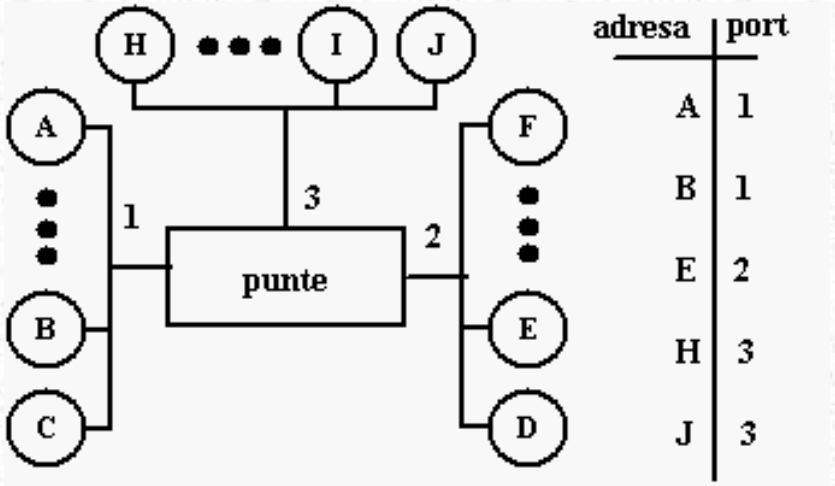
- puntea "observa" traficul si "invata" care statii sunt atasate
  - transparenta: se adauga doar puntea la LAN, toate gazdele se comporta ca si cum puntea nu ar fi acolo

Punti transparente

1. puntea primeste fiecare pachet transmis pe fiecare LAN atasat
2. puntea stocheaza pentru fiecare pachet
   - adresa fizica a expeditorului
   - portul (segmentul LAN de sosire) pe care a fost primit pachetul
3. pentru fiecare pachet primit pe fiecare port se cauta adresa fizica a destinatarului in tabel
   - daca nu se gaseste, se floodeaza in toate LAN-urile atasate
   - daca de gaseste, se directioneaza numai catre LAN-ul specificat
4. tabelul de directionare sters daca nu este improspatat (prin 2).

Punti transparente: exemplu

Exemplu: presupunem ca C transmite pachete la D si D raspunde cu pachet catre C



- C trimite pachet, puntea nu are nicio informatie despre D, astfel incat floodeaza ambele LAN-uri
  - puntea noteaza ca C este pe portul 1
  - pachetul ignorat pe LAN-ul superior
  - pachetul primit de D
- D genereaza raspuns catre C, transmite
  - puntea vede pachetul din D
  - puntea noteaza ca D este pe partea 2
  - puntea stie pe C pe portul 1, astfel incat se directioneaza selectiv pachetele in afara pe partea 1.

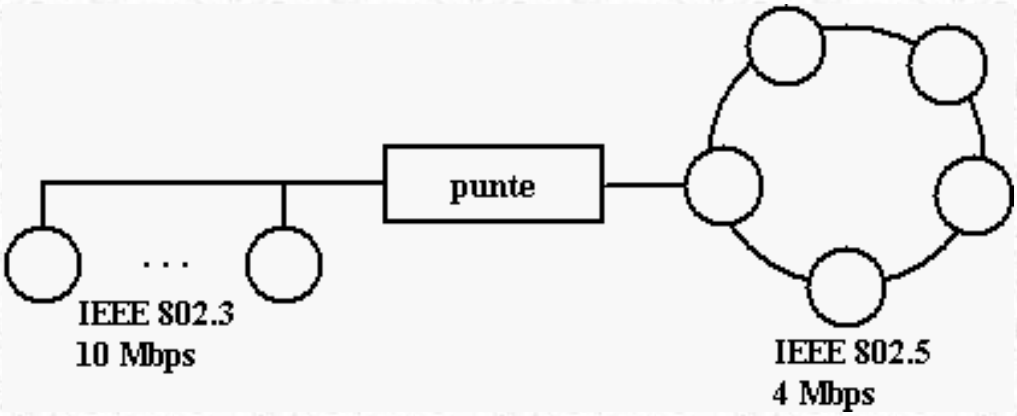
Punti: probleme ale standardelor cu 3 LAN-uri

Procesare

- s-ar putea sa fie nevoie ca puntea sa translateze intre 3 802 standarde (fiecare 802.(are format diferit)
- pachetul translatat necesita noi verificari de suma.

Nepotrivirea vitezei

- diferite LAN-uri 802,(opereaza la viteze diferite



Nepotriviri de dimensiune

- dimensiunea maxima a pachetului 1518 octete, 802.4 are dimensiunea maxima a pachetului 8191 octete

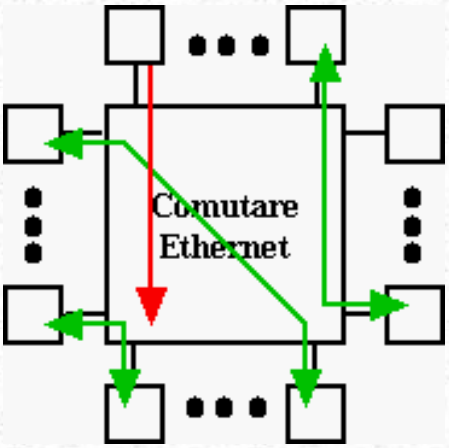
LAN-uri comutate 802.3

LAN-uri 802.* generale cu punti interconectate

- se poate solicita conversia pachetului

Ethernet comutat:

- "hub"-ul central interconecteaza segmentele ethernet
- in practica, fiecare segment are adesea numai un computer
- transmisie simultana catre aceeasi destinatie
  - in primul rand se lasa unul sa treaca
  - este posibil sa se bufereze celelalte pachete.



8. Nivelul Fizic

Mediu: Cablu torsadat bifilar

Doua fire de cupru izolate, rasucite elicoidal

Linie telefonica "standard"

- cablu torsadat bifilar din categoria 3: poate transmite date la mai multi megabiti/sec la o lungime de cativa kilometri
- cablu torsadat bifilar din categoria 5: Ethernet de mare viteza (100Mbit/sec) si ATM (155Mbit/sec)

Mediu: Cablu pentru banda de baza

Transmisie digitala bidirectionata pe cablu coaxial (de ex., Ethernet)

- digital: nu este nevoie de niciun modem

Un singur canal.

Viteza datelor de pana la 1-2 Gbps la peste 1 km.

Mediu: Cablu de banda larga

Foloseste tehnologia TV standard pentru cablu

- transmisie analogica
- modemul este necesar pentru conversia digital -> transmisie analogica-> digital
- in mod traditional: transfer unidirectional (cablu TV), transfer bi-directional posibil

Multiple "canale" posibile pe acelasi cablu fizic

- fiecare canal foloseste benzi de frecvente diferite: multiplexarea prin diviziune in frecventa
- fiecare canal: mai multi megabiti/sec

Repetoare la fiecare 5 km (din cupru).

Mediu: Fibre optice

Transmisie digitala folosind pulsuri de lumina.

Largimea de banda: 100 Gbps pe distante scurte

Unidirectional

Repetoare la fiecare 30 km.

Comunicare versus procesare

- *procesare:* 1 instructiune/100 nsec in 1970 pana la 1 instructiune/nsec in anii 1990 (doua ordine de marime)
- *comunicare* linii de 56 Kbps in anii 1970, zeci de Gbps in anii 1990 (sase ordine de marime)

Mediu: Fara fir

Foloseste spectrul electromagnetic pentru transmisie.

Capacitatile canalului depind puternic de frecventa si de tehnologia transmisiei.

Caracteristicile legaturii:

- mai zgomotoasa (mai multe erori de bit) decat fibra si cablul
- LAN-uri fara fir de 2-10Mbps folosind spectru imprastiat, banda ingusta, infrarosu
- transmisiua pe distante lungi:
  - 128kbps in intervalul 50khz
  - 2-3Mbps in intervalul 900Mhz.

Arhitecturi de retea pentru a ajunge la utilizatorul final

Focalizarea noastra implicita: mediu corporativ/birouri

- utilizatorul final pe LAN-uri

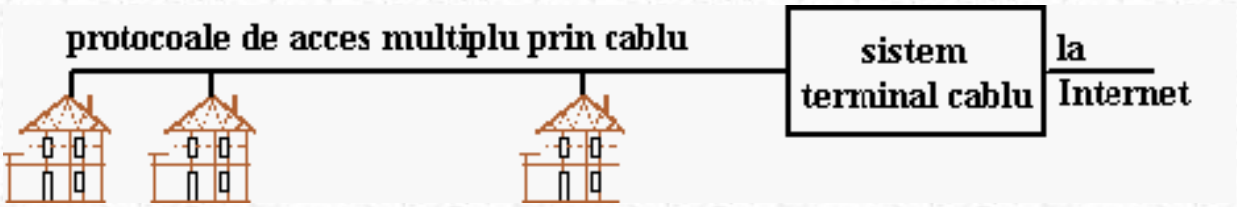
- LAN-uri conectate in campus/companie
- campus/companie conectate la ISP

Datele la utilizatorul rezidential

- acces Internet multimedia
- video la cerere
- curs de retele de calculatoare la cerere

Retele prin sisteme de cablu: modemuri de cablu

Se pot folosi unul sau mai multe canale pe sistemul de cabluri existent pentru utilizatorul rezidential de retea pentru a cabla intre capete



Canale simetrice versus asimetrice

- modemuri de cablu simetrice 4Mbps
- 10M acasa, 768K upstream

Tehnici de acces multiplu

- Zenith: CSMA/cd, Motorola: polling, Baynetworks: TDMA

HFC: cablu de fibra optica hibrid (hybrid fiber cable): fibra pe traseu, cablu acasa.

Nota: numai 5% din reteaua de cablu existent are amplificatori cu doua cai.

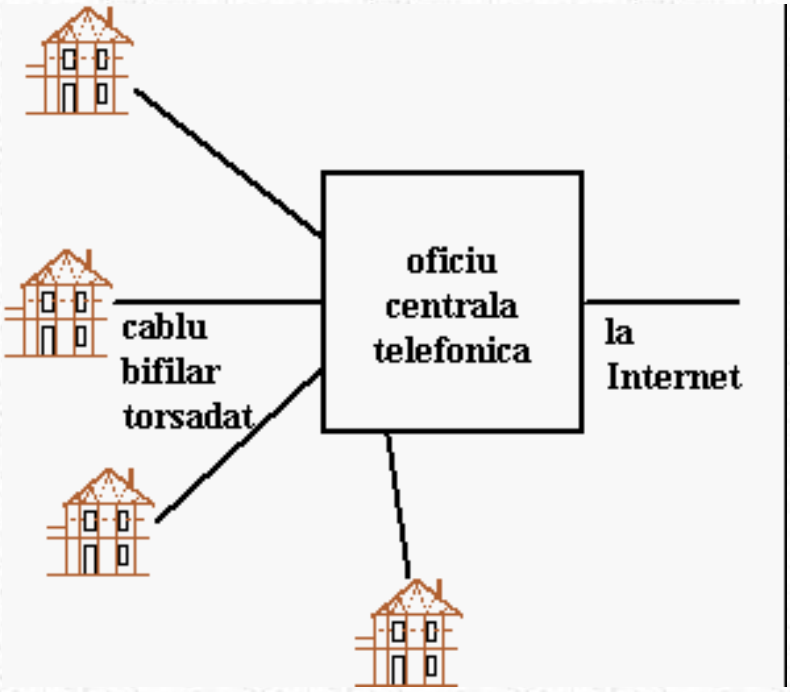
Retele via compania de telefoane: ADSL

ADSL: linie de abonat digitala asimetrica (asymmetric digital subscriber line)

Date de mare viteza prin cablul torsadat biaxial existent.

6Mbps downstream acasa, 640kbps upstream pe firul telefonic existent.

50% din liniile telefonice au capabilitati de ADSL.



Retele via compania de telefoane: ISDN

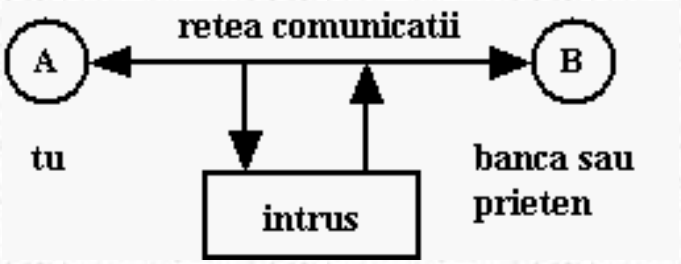
Companiile de telefoane au lucrat 15 ani la standardul ISDN de banda ingusta.

- cablu torsadat biaxial
- viteza de baza: 2 canale 64Kbit plus 1 canal 16bit
- 2B+D

E-mail: engineering@teleactivities.net

9. Securitatea retelei

Securitatea retelei



Intrusul poate

- sa traga cu urechea
- sa inlature, modifice si/sau introduca mesaje
- sa citeasca mesaje.

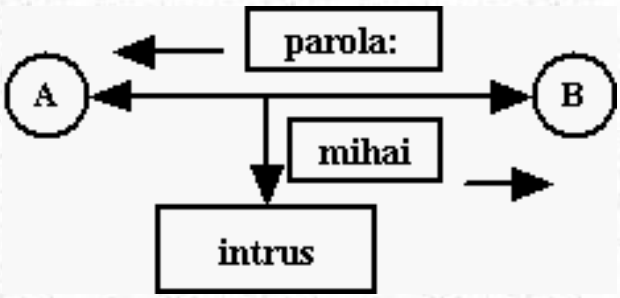
Aspecte importante:

- *criptografia:* asigura secretul informatiilor care trebuiesc transmise
- *autentificare:* dovedesti cine esti si verifica identitatea interlocutorului.

Securitatea in retelele de calculatoare

Resurse utilizator:

- parola de intrare adesea transmisa necriptata in pachetele TCP intre aplicatii (de ex., telnet, ftp)
- parolele ofera protectie redusa



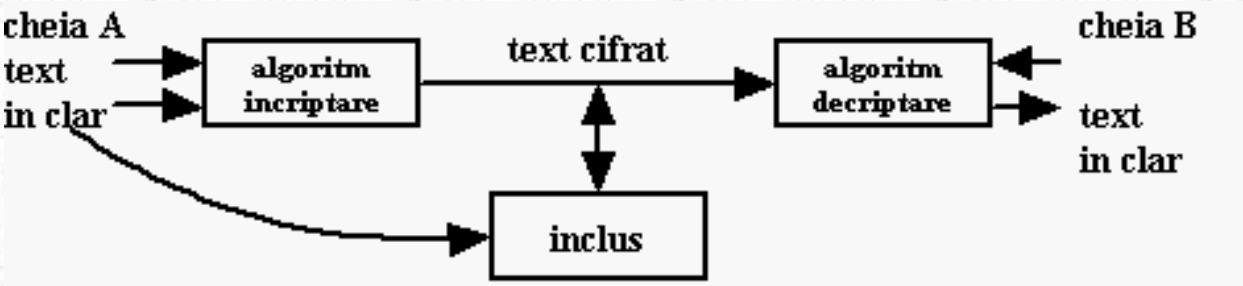
Resurse retea:

- adesea este complet neprotejata de ascultarea, introducerea sau falsificarea mesajelor de catre intrusi
- spoofingul postei electronice, actualizarile ruterului, mesajele ICMP, mesajele de management al retelei: modalitati mai eficiente de protectie.

Atentionari:

- intrusul care ataseaza masina sa (obtinand codul SO, privilegii de administrator) in retea, poate sa suprascrie multe masuri de securitate oferite de sitem
- utilizatorul trebuie sa aiba un rol mult mai activ

incriptarea



Text in clar: mesaj necriptat.

Text cifrat: forma incriptata a mesajului

Intrusul poate

- intercepta transmisia textului cifrat
- intercepta perechea text in clar/text cifrat
- obtine algoritmii de incriptare/decriptare

Un algoritm simplu de incriptare

Cifru de substitutie:

```
abcdefghijklmnopqrstuvwxyz
```

```
poiuytrewqasdfghjklmnbvczx
```

- inlocuie fiecare caracter al textului in clar din mesaj cu caracterele corespunzatoare din textul cifrat:

```
plaintext: Charlotte, my love
```

```
ciphertext: iepksgmmy, dz sgby
```

- *cheia* reprezinta imperecherea dintre caractere textului in clar si cele ale textului cifrat
- *cheia simetrica:* expeditorul si destinatarul utilizeaza aceeasi cheie
- exista 26! (aprox 10**26) de chei diferite posibile: improbabil sa fie ghicite din intamplare
- cifru de substitutie poate fi descifrat folosind frecventa observata a literelor
  - in limba engleza, 'e' este cea mai comuna litera, iar ;the' este cel mai obisnuit cuvant.

DES: Standard de incriptare a datelor (Data Encryption Standard)

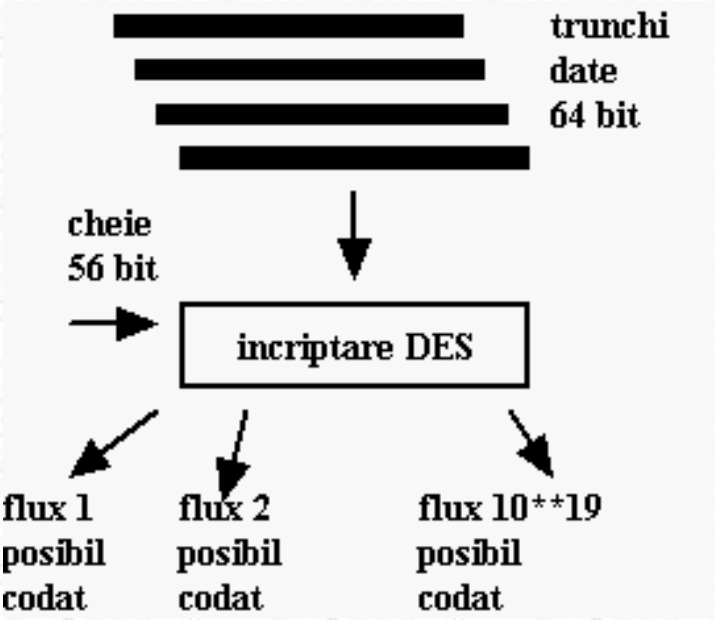
incripteaza datele in trunchiuri de cate 64-bit.

Algoritmul de incriptare/decriptare este un standard publicat.

- fiecare stie cum sa procedeze

Cifrul de subtitutie are pentru trunchiurile de 64 bit: cheia de 56 bit determina care din cele 56! cifruri de substitutie va fi folosit

- substitutia: 19 etape de transformare, 16 implicand functii ale cheii



- decriptarea este realizata parcurgand in sens invers etapele incriptarii
- expeditorul si destinatarul trebuie sa foloseasca aceeasi cheie.

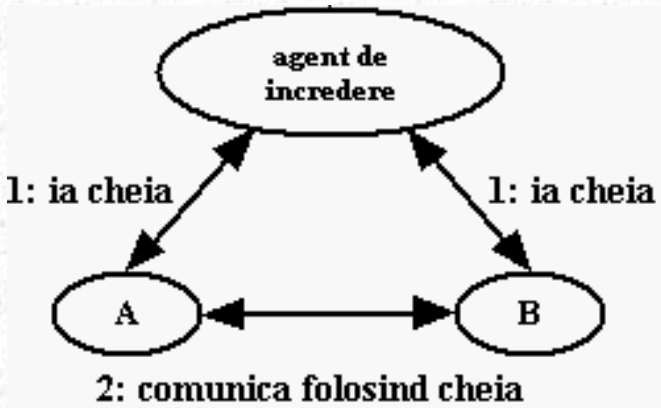
Problema distributiei cheii

Problema: cum cad comunicantii de acord asupra cheii simetrice?

- N comunicanti implica N chei

Distributia agentului de incredere:

- chei distribuite prin agensi de incredere centralizati
- oricare comunicant are nevoie doar sa stie cheia de comunicare cu agentul de incredere
- pentru comunicatii intre I si J, agentul de incredere va oferi o cheie.



Criptografia prin chei publice

Chei de incriptare/decriptare separate

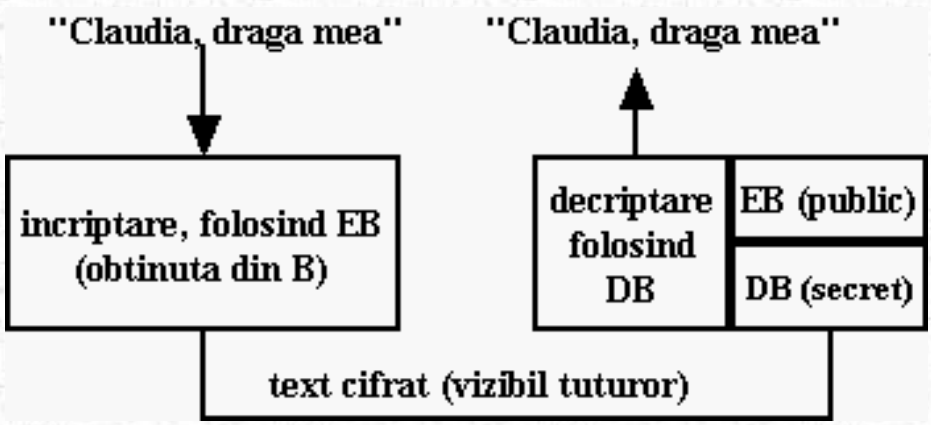
- destinatarul face *cunoscuta* (!) cheia sa de incriptare
- destinatarul pastreaza secretul cheii

Pentru a transmite catre destinatarul B, se incripteaza mesajul M folosind cheia lui B accesibila public, EB

- transmite EB(M)

Pentru a decripta, B aplica cheia sa privata de decriptare DB la primirea mesajului.

- procesarea DB(EB(M)) da M



Cunoasterea cheii de incriptare nu ajuta la decriptares decriptarea nu este un proces invers al ancriptarii.

Numai destinatarul poate decripta mesajul.

RSA: incriptarea/decriptarea prin chei publice

RSA: un algoritm de incriptare/decriptare folosind chei publice.

Entitatile care doresc sa primeasca mesaje incriptate:

- aleg doua numere prime, p, q mai mari ca 10**100
- calculeaza $n=pq$ si $z = (p-1)(q-1)$
- se alege numarul d care nu are niciun factor comun cu z
- se calculeaza e astfel incat $ed = 1 \mod z$, adica:

$numarul\ intreg\ din(\ (ed)\ /\ ((p-1)(q-1))\ ) = 1$, $respectiv$,

$ed = k(p-1)(q-1) + 1$

- trei numere:
  - e, n se fac publice
  - d se pastreaza secret

Pentru incriptare:

- se imparte mesajul in i *blocuri*, bi de marime k: $2^{**}k < n$
- se incripteaza: $encrypt(bi) = bi^{**}e \mod n$

Pentru decriptare:

- $bi = encrypt(bi)^{**}d$

pentru a sparge RSA

- trebuie sa se stie p, q, cunoscandu-se $pq=n$, n cunoscut
- se factorizeaza 200 digiti n in numere prime si iti trebuiesc 4 miliarde de ani folosind metodele cunoscute.

RSA Exemplu

- se alege p=3, q=11, se da n=33, (p-1)(q-1)=z=20
- se alege d = 7 intrucat 7 si 20 nu au niciun factor comun
- se calculeaza e = 3, a.i. ed = k(p-1)(q-1)+1 (nota: k=1 aici)

Expeditor:

text in clar		foloseste e=3	text cifrat
char	#	#**3	#**3 mod 33
S	19	6859	28
U	21	9261	21
N	19	2744	5

Destinatar:

text cifrat		foloseste d=7	text in clar
C	c**7	c**7 mod 33	char
28	13492928512	19	S
21	1801088541	21	U
5	78125	14	N

Alte observatii privind RSA

De ce functioneaza RSA?

- Din teoria numerelor rezulta: daca p, q prime atunci bi**((p-1)(q-1)) mod pq = 1
- folosind mod pq aritmetic:

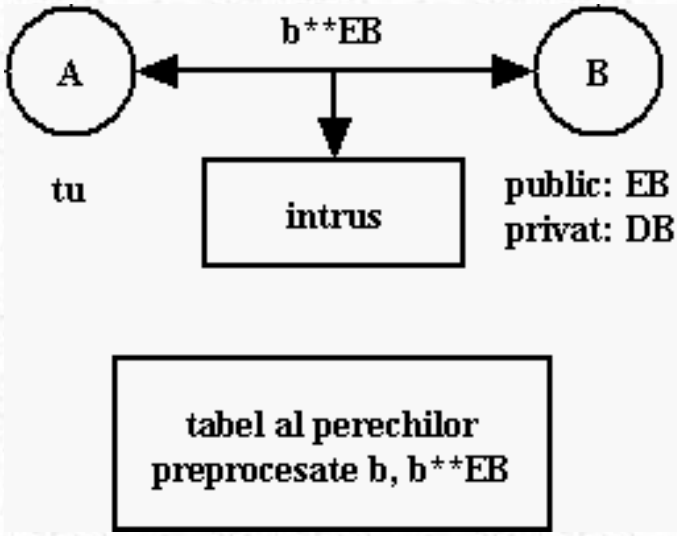
```
(b**e)**d = b**(ed)

= b**(k(p-1)(q-1)+1) for some k

= b b**(p-1)(q-1) b**(p-1)(q-1) ... b**(p-1)(q-1)

= b 1 1 ... 1

= b
```

- Nota: putem de asemenea incripta cu d sau cu e
  - Aceasta modalitate ne ajuta fiind mai scurta

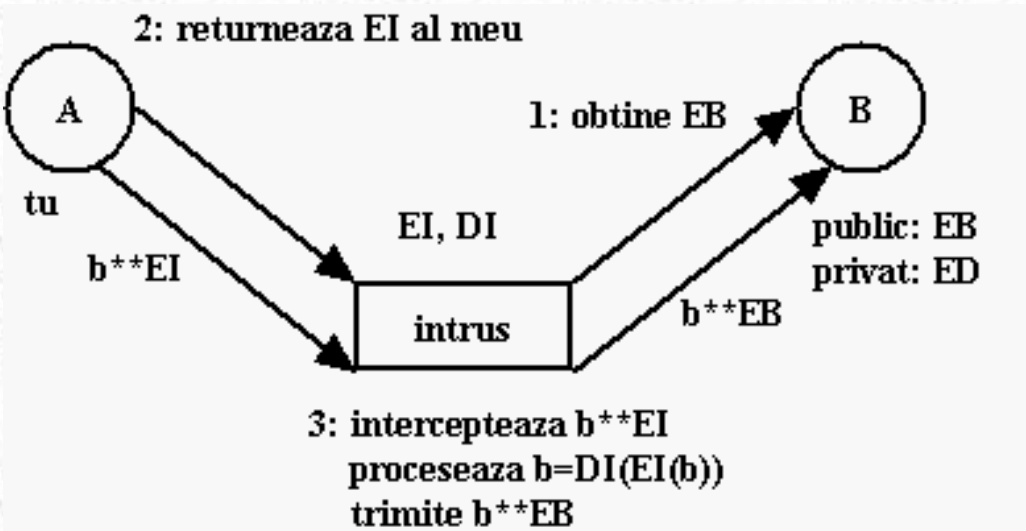
Cum se sparge RSA?

Prin forta bruta: se obtine cheia publica a lui B

- pentru fiecare bi posibil din textul in clar, calculeaza bi**e
- pentru fiecare bi**e observat, putem afla astfel bi
- mai mult: alegem marimea lui bi "suficient de mare"



Intermediarul: intercepteaza cheile, spoofeaza identitatea:



Autentificarea

intrebare: cum stie un destinatar ca entitatea cu care comunica la distanta este cine spune ca este?

Abordarea 1: autentificarea pe baza de cheie peer-peer

- A, B (doar) stiu cheia de securitate pentru incriptare /decriptare
- A transmite un mesaj incriptat lui B si B il decripteaza

```
A catre B: msg = encrypt("I am A")
B proceseaza: if decrypt(msg)=="I am A"
   then A is verified
   else A is fradulent
```

Autentificarea folosind Nonces

pentru a demonstra ca A este in legatura, B trimite un numar "doar - o - data - in - viata" (nonce) lui A, pe care A il incodeaza si il returneaza lui B

```
A catre B: msg = encrypt("I am A")
B proceseaza: if decrypt(msg)=="I am A"
   then A is OK so far
B catre A: once-in-a-lifetime value, n
A catre B: msg2 = encrypt(n)
B proceseaza: if decrypt(msg2)==n
   then A is verified
   else A is fradulent
```

Autentificarea folosind chei publice

B doreste sa autentifice pe A

A a realizat o cheie proprie de incriptare cunoscuta EA

Doar A cunoaste DA

Simetrie: DA(EA(n)) = EA (DA(n))

```
A catre B: msg = "I am A"
```

```
B catre A: once-in-a-lifetime value, n
A catre B: msg2 = DA(n)
B proceseaza: if EA (DA(n))== n
   then A is verified
   else A is fradulent
```

Semnaturi digitale folosind chei publice

Scopul semnaturii digitale:

- expeditorul nu poate respinge un mesaj netransmis niciodata ("Eu nu am transmis niciodata asa ceva")
- destinatarul nu poate contraface un mesaj primit

Sa presupunem ca A solicita ca B sa "semneze" mesajul M

```
B transmite DA(M) lui A
A proceseaza daca EA ( DA(M)) == M
  atunci A a semnat M
```

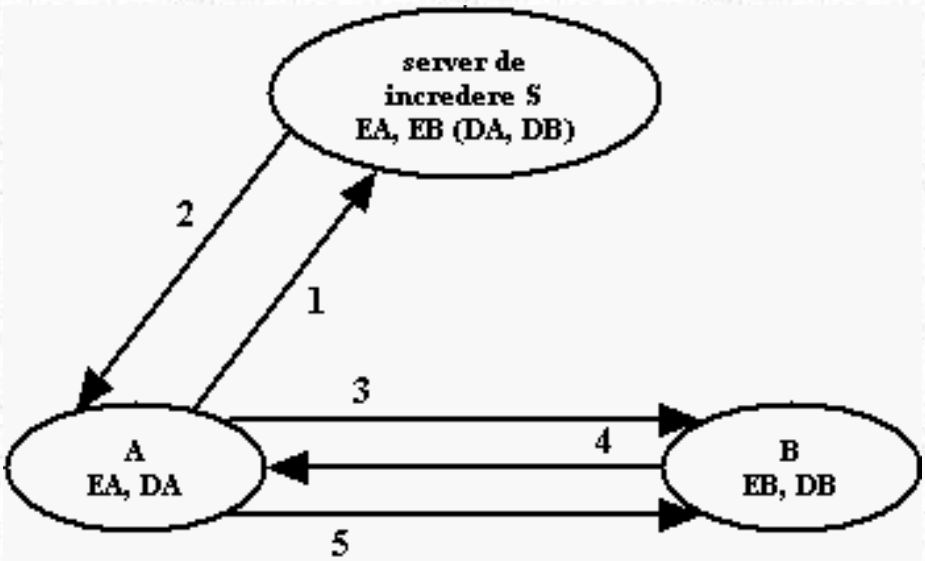
Schimb de chei simetrice: server de incredere

Problema: cum se inteleg entitatile distribuite asupra unei chei?

Presupunere: fiecare entitate are propria sa cheie unica, pe care o stie numai ea si serverul de ancredere.

Serverul:

- va genera o cheie numai pentru o sesiune pe care A si B o folosesc pentru a incripta comunicarea
- va folosi cheile lui A si B pentru a comunica cheia de sesiune lui A si B



Scenariu anterior:

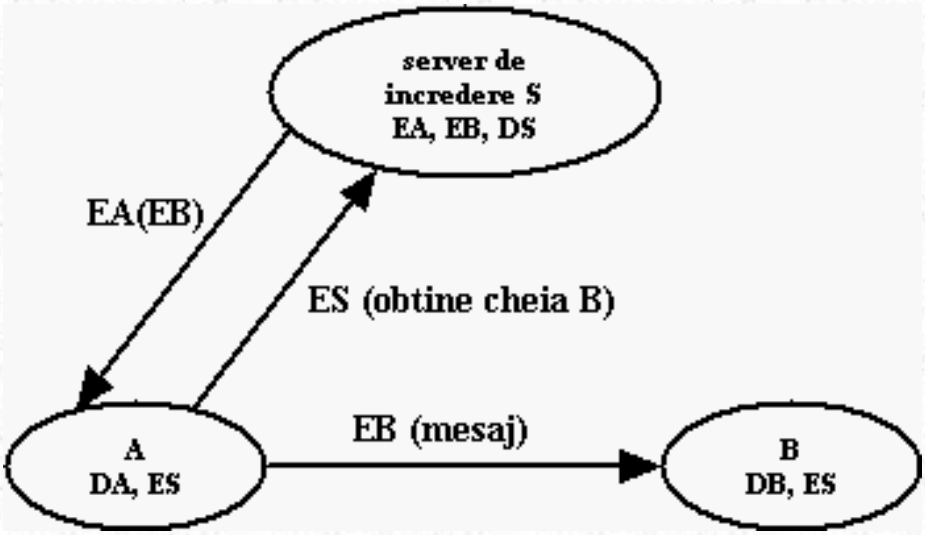
1. A trimite mesajul criptat catre S, continind A, B, nonce RA: EA(A,B,RA)
2. S il decripteaza folosind DA, genereaza o cheie de sesiune, K, trimite nonce, cheia, si cheia de incodare criptata a lui B catre A: EA(RA,B,K,EB(K,A))
3. A decripteaza mesajul de la S folosind DA si verifica nonce. Extrage K, o salveaza si trimite EB(K,A) catre B.
4. B decripteaza mesajul folosind DB, extrage K, genereaza un nou nonce RB, trimite EK(RB) catre A
5. A il decripteaza folosind K, extrage RB, proceseaza RB-1 si incripteaza folosind K. Trimite EK(RB-1) catre B
6. B il decripteaza folosind K si verifica RB-1.

Restabilirea cheii publice poate fi atacata de un intrus.

Localizeaza toate cheile publice din serverul de incredere.

Oricine poate avea cheia de incriptare a serverului (ED public)

Presupunem ca A doreste sa transmita catre B folosind cheia "publica" a lui B.



Cipul Clipper: Aspecte tehnice

Guvernul SUA a propus un standard federal de procesare a informatiei.

- evident ca trebuiesc incriptate multe lucruri care trec prin lnia telefonica
- tehnica de incriptare pentru Clipper (algoritmul skipjack) este strict secret
- este instalat pe baza de voluntariat in echipamentele de telecomunicatii (in produsele existente).

Setarea apelului: A si B doresc sa comunice

- A, B folosesc tehnici standard de chei publice pentru a se intelege asupra unei chei de sesiune
- cheia de sesiune este incriptata folosind cheia unitatii de cipuri Clipper

- cheia de sesiune incriptata si unitatea ID neincriptata se pun in LEAF (Law Enforcement Access Field - Camp de acces al restrictionarii legale) care este transmis
  - nota: LEAF redundant, A si B cunosc sesiunea K
  - cheia de sesiune este transmisa astfel incat poate fi interceptata!
- Comunicarea sesiunii respective este incriptata folosind cheia de sesiune.

Probleme de confidentialitate

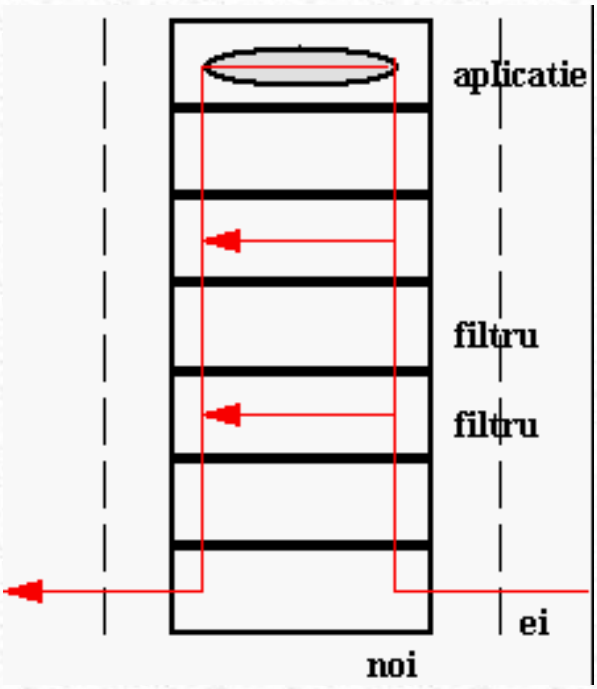
Clipper I: fabricantii de dispozitive descompun cheia cipului unitar in doua:

- cheia cipului unitar inclusa la nivel de hardware intr-un cip imposibil de falsificat si la care nu se poate aplica tehnica de inginerie inversa (reverse-engineering)
- in SUA, jumatate din aceasta cheie se pastreaza la NIST, jumatate la Trezorerie
- guvernul doreste sa poata accesa masini cu ID cunoscut
  - pentru aceasta, FBI trebuie sa prezinte Curtii dispozitii de la ambele agentii, pentru a obtine cheia unitara a cipului
  - aceasta cheie se foloseste pentru a determina cheia de sesiune din LEAF
  - mesajele sunt apoi decriptate folosind cheia de sesiune

Guvernul SUA a interzis exportul tehnologiei de obtinere a cheilor de incriptare mai mari de 40 bit.

- in oct 96 s-a permis, ca masura temporara si de proba, exportul selectiv pentru doi ani a tehnologiei pe 56 bit.

Protectia impotriva intrusilor: Firewalls



Firewall: componenta retea (gazda/ruter+software) situata intre "noi" si cei din afara ("ei").

Pachet filtrat de firewalls: selecteaza pachetele in functie de sursa, sau adresa dee

destinatie (de ex., adresa IP, port).

Aplicatia gateways: cod specific de aplicatie care proceseaza, si/sau retransmite pachetele specifice unei aplicatii

- de ex., gatewazs pentru posta sau telnet
- codul gatewaz al aplicatiei poate fi securizat la nivel de hard
- poate loga toate activitatile

Securitatea: Activitatea Internet

Nivel IP:

- autentificarea headerului: destinatarul poate autentifica expeditorul folosind codul de autentificare a mesajului (message authentication code (MAC))
- ancriptarea continutului: DES, RFC 1829

IPA

- SSL - secure socket layer: suport pentru autentificare si incriptare
  - numere de port: 443 pentru http cu SSL, 465 pentru smtp cu SSL

Nivel Aplicatie

- Posibilitatea mesajelor confidentiale
- secure http: suporta multe autentificari , scheme de incriptare

Securitatea: concluzii

Aspecte cheie:

- incriptarea
- autentificarea
- schimbul de chei

De asemenea:

- aria utilizarilor sunt in crestere pe masura ce creste conectivitatea
- semnatura digitala, banii digitali, autentificarea, biometrica se vor dezvolta mult
- un aspect important este cel social

Bibliografie:

- Crypto Policy Perspectives: S. Landau et al., Aug 1994 CACM
- Internet Security, R. Oppliger, CACM May 1997
- www.eff.org

Back | Top | Next

E-mail: engineering@teleactivities.net

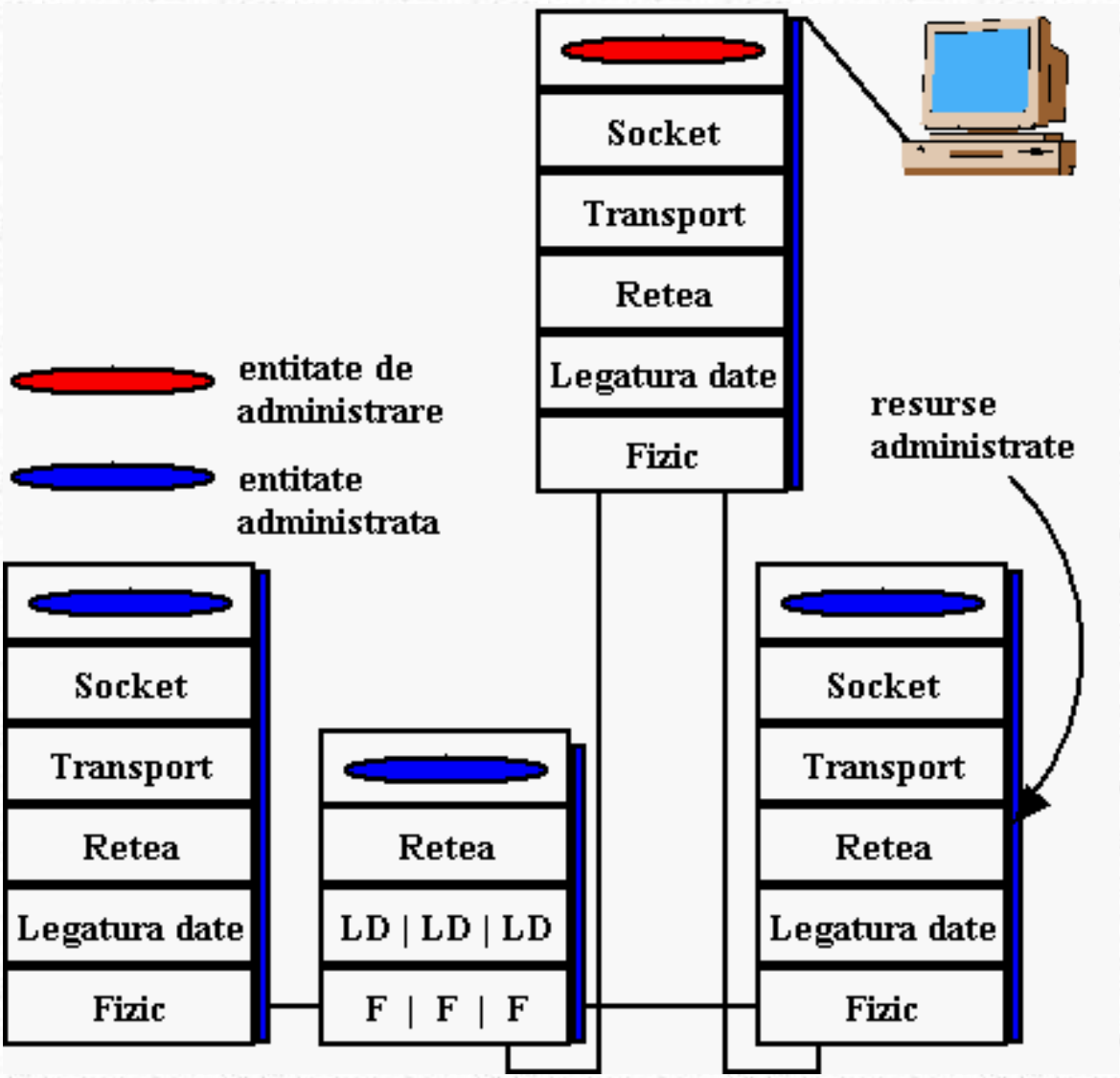
10. Managementul retelei

Managementul retelei: Introducere

Reteaua consta din multe resurse heterogene, realizate de mai multi fabricanti: rutere, punti, gazde, servere terminale, modemuri, legaturi, interfete.

Scopul manegementului retelei:

- identificarea si corectarea problemeleor de disfunctie la nivel hardware/software
- monitorizarea si optimizarea performantelor



Managementul retelei: Probleme

Abordarile managementului retelei trebuie sa:

- se situeze la o scara corecta: un mare numar de entizati care trebzuiesc administrate
- nu interfere cu operarea obisnuita (nivel scazut)
- poata lucra sub stres: cel mai important cand reteaua lucreaza sub stres

Probleme:

- ce resurse vor fi administrate
- cum se denumesc/descriu resursele administrate
- standarde:
  - Internet: SNMP: Simple Network Management Protocol
  - OSI: CMIP: Common Management Information Protocol

Administrare si entitati administrate

Entitatea de administrat:

- are o "privire de ansamblu" asupra retelei
- seteaza programele la nivel de aplicatie care controleaza/administreaza reteaua
  - cu interventie umana
  - cu asistenta de sistem AI (expert) pe baza unor reguli
- comunica cu entitatea administrata:
  - solicita date privind starea legaturilor, tabelele de rutare, numarul pachetelor esuate, etc.
  - schimbari efectuate, precum intreruperea unei legaturi, etc.

Entitatea administrata:

- proces la nivel de aplicatie localizat in fiecare loc de resurse pentru a comunica cu managerul retelei
  - raspunde la solicitarile managerului
  - notifica managerul asupra evenimentelor importante)de ex., picarea unei legaturi)

SNMP

Administrarea entitatii se realizeaza in statiile de management al retelei (NMS)

Entitatea administrata este numita agent SNMP.

MIB: Management Information Base (Baza de informare a manegementului)

- stocare logica a informatiei pentru managementul retelei
- mentinut local de catre agentul SNMP
- interogat si modificat de catre NMS
- 175 "obiecte" organizate in 10 grupe: sistem, interfete, translari de adrese, IP, ICMP, TCP, UDP, EGP, transmisie, SNMP

Variabile MIB legate de UDP:

nume	descriere
udpInDatagrams	# datagrame UDP dg livrat in proces
udpNoPorts	# datagrame UDP fara nicio aplicatie primita
udpInErrors	#alte erori UDP (de ex., checksum)
udpOutDatagrams	# datagrame UDP transmise
udpTable	adresele IP (interfata) si porturile pentru care sistemul va primi datagrame UDP, de ex. portul 520 pentru mesaje de rutare RIP

Variabile MIB

Variabile MIB pentru o interfata (hardware/software la nivel de legatura date):

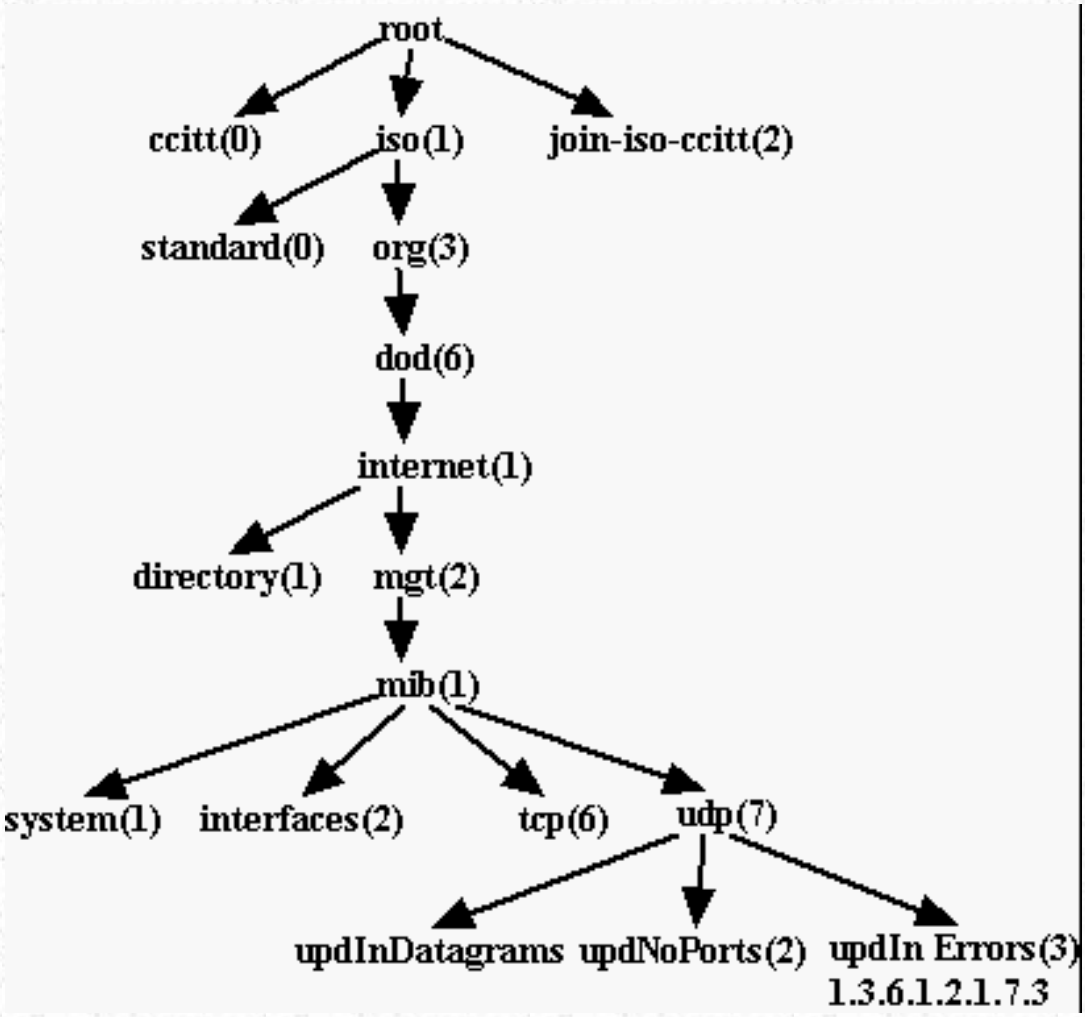
nume	descriere
ifIndex	indexul interfetei
ifDescr	descrierea textuala a interfetei

ifType	tipul interfetei (de ex., 7 pentru IEEE 802.3)
ifMTU	dimensiunea maxima a pachetului
ifSpeed	viteza in bit/sec
ifPhysAddress	adresa fizica (de ex., adresa 802.*)
ifOperStatus	1 pentru up, 2 pentru down, 3 pentru testare
ifInErrors	# pachetelor care sosesc si la care se renunta din cauza erorilor
ifInDiscards	# pachetelor la care se renunta datorita supraincarcarii buferelor
ifInUcastpkts	# pachetelor unidestinatie primite
ifOutQLen	# pachetelor aflate in asteptare

Variabile MIB de referinta

Tipul de reapelare ASN.1 OBJECT IDENTIFIER:

- ofera metode standard ISO structurate pentru denumirea obiectelor
- obiectele care se pot denumi includ protocoalele si variabilele MIB
- de ex., 1.3.6.1.2.1.7.1 spune numarul de pachete UDP livrate sa procesele utilizatorului:



Protocol SNMP

Comunicare intre entitatea de administrat si entitatea administrata transmisa prin porturile UDP 161, 162

Protocolul SNMP are 5 tipuri de mesaje:

- *get-request:* valoare provocata uneia sau mai multor variabile MIB
- *get-next-request:* pentru looping printre variabile si tabele
- *set-request:* spune agentului sa seteze valoarea variabilei MIB la valoarea specificata
- *get-response:* folosit de catre agent pentr a returna valoarea la manager
- *trap:* folosit de catre agent pentru a notifica managerul asupra "evenimentului"

Pachete de detectie:

numele detectiei	descriere
cold start	autoinitializarea SNMP a agentului
warm start	reinitializarea SNMP a agentului
link up	interfata schimbata din starea de jos in cea de sus
link down	interfata schimbata din starea de sus in cea de jos
authentication failure	pachet SNMP primit de la manager necunoscut

Bibliografie

- IEEE Network magazine, nov. 1993

Back Top Next

11. Garantii in Sistemul Calitatii (SC)

Garantii SC: Motivatie

Anumite aplicatii necesita un nivel minim al performantei retelei:

- telefonia Internet, teleconferinta, intarzieri > 500ms in cazul interactiei cu imperfectiunile umane
- sesiunea este garantata SC sau este blocata (este negata admisia in retea)

Nepotriviri fundamentale intre SC si comutarea de pachete:

- comutarea pachetelor: resurse partajate statistic in speranta ca cererile de varf a sesiunii nu coincid
  - este necesara o certitudine de 100% luand in calcul cazul cel mai rau posibil
- admisia/negarea sesiunii in cazul cel mai rau este echivalent cu comutarea de circuite!

Garantii SC: clase ale serviciului Internet

Modelul serviciului curent: "cel mai bun efort"

- trimite pachet si spera ca performanta este OK

Urmatoarea generasie de clase de servicii Internet:

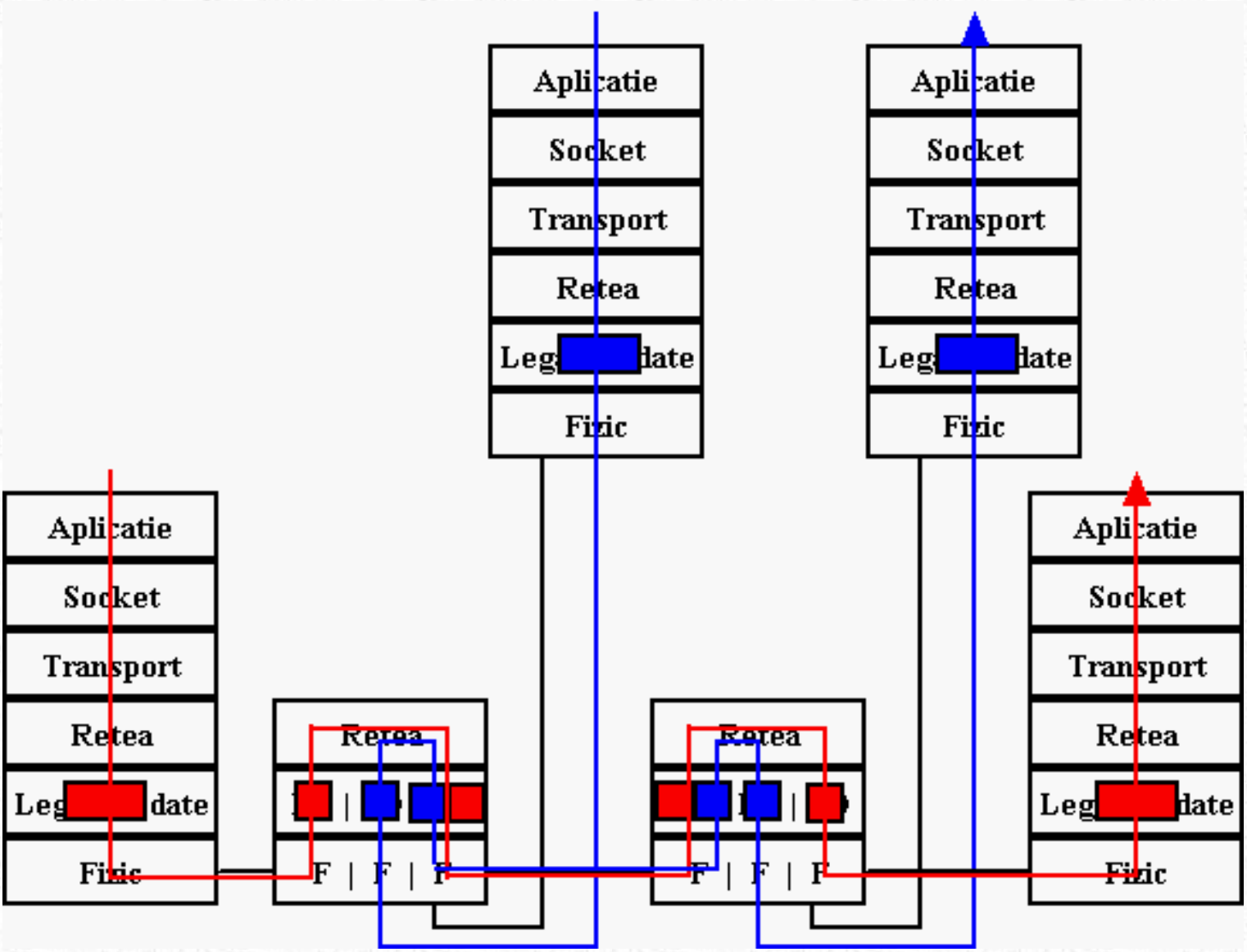
- *serviciu garantat:* "ofera limite clare (demonstrabile matematic) pentru datagrama cap-cap a intarzierilor in asteptare. Acest serviciu face posibila oferirea unui serviciu care garanteaza atat intarzierile cat si largimea de banda."
- *incarcarea controlata:* "un SC care aproximeaza strans SC pe care acelasi trafic l-ar primi de la un element de retea neincarcat, dar foloseste controlul capacitatii (admisiei) pentru a asigura ca acest serviciu este primit chiar cand elementul de retea este supraincarcat."
- *cel mai bun efort: modelul de serviciu curent*

Garantii SC: clase de serviciu ATM

Clase de serviciu ATM:

- *CBR:* viteza de bit constant (constant bit rate), largime de banda garantata, intarziere constanta cap-cap
- *ABR:* viteza (largimea de banda) minima garantata a celulei, mai posibila cand este disponibila (controlul congestiei via celule RM)
- UBR: viteza de bit nespecificata, niciun control al congestiei.

CS: sunt necesare schimbari radicale!



Problema admisiei apelului

Reteaua trebuie sa decida daca "admie" apelul oferit (sesiunea).

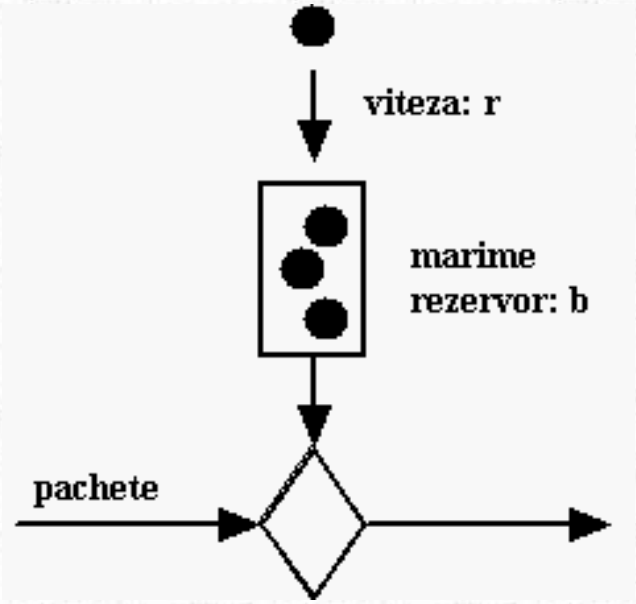
- *retele curente*: toate apelurile acceptate, performanta se degradeaza cand sunt transportate mai multe apeluri

Specificarea traficului: Tspec

Apelul trbuie sa descrie traficul pe care il va injecta in retea.

Propunerea rezervorului cu scapari: traficul care intra in retea este filtrat de rezervorul regulator cu scapari:

- B: marimea maxima a scaparilor
- r: viteza medie
- cantitatea de trafic care intra prin orice interval de lungime t, mai mic decat b + rt



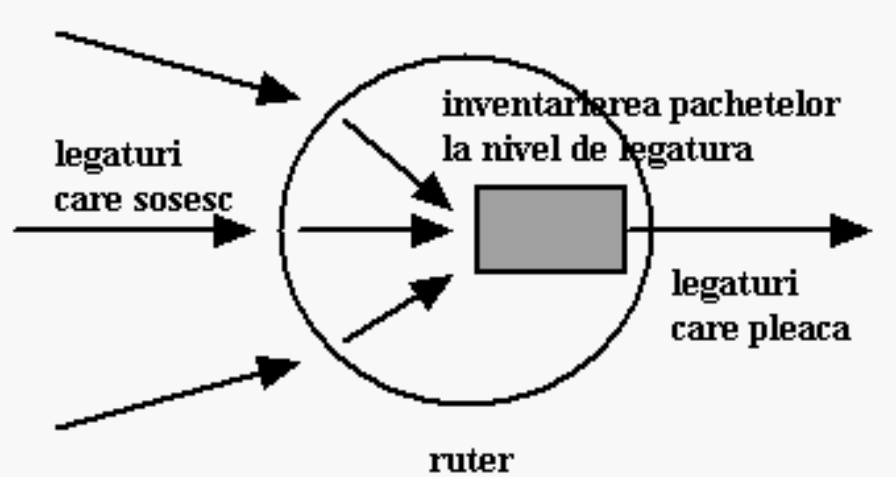
Rezervorul cu jetoane poate folosi: ajustarea, restrictionarea, marcarea

- intarzie pachetele la intrarea in tara (ajustare)
- lasa sa se scurga pachetele care sosesc fara jetoane (functia de restrictionare)
- lasa toate pachetele sa treaca, le marcheaza: cele cu jetoane, cele fara jetoane
- reteaua lasa sa se scurga pachetele fara jetoane in timpul congestiei (marcare)

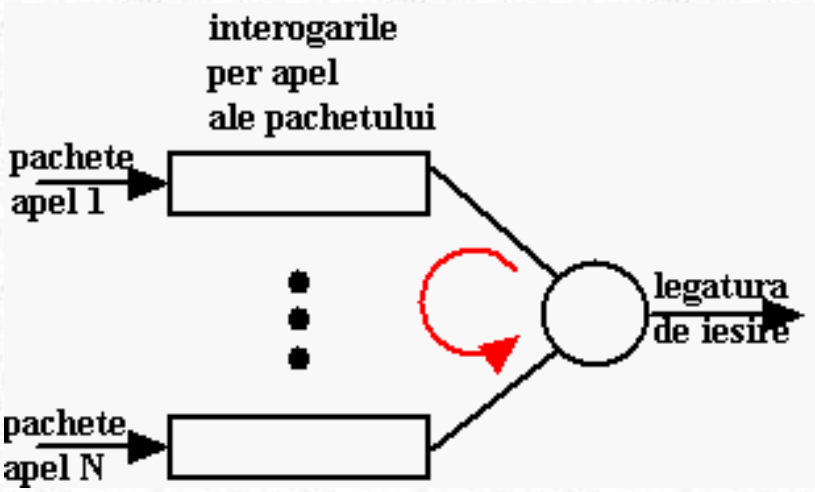
Nivelul Legatura: componenta critica a SC

Buferarea si largimea de banda; resursele critice

- cauza pierderii si intarzierii
- disciplina planificata de pachet, managementul buferului va determina pierderea, intarzierea vazuta de un apel
  - FCFS
  - Weighted Fair Queueing (WFQ)



Planificarea la nivelul legaturii: WFQ



Serviciu "in cerc":

- se considera pachete de lungime fixa, N sesiuni
- sesiunea i apuca sa transmita 1 pachet la fiecare "tura"
- daca sesiunea i nu are niciun pachet, i+1 preia sansa.

WFQ: Caracteristici bune

Fiecare sesiune obtine o cantitate minima de largime de banda garantata.

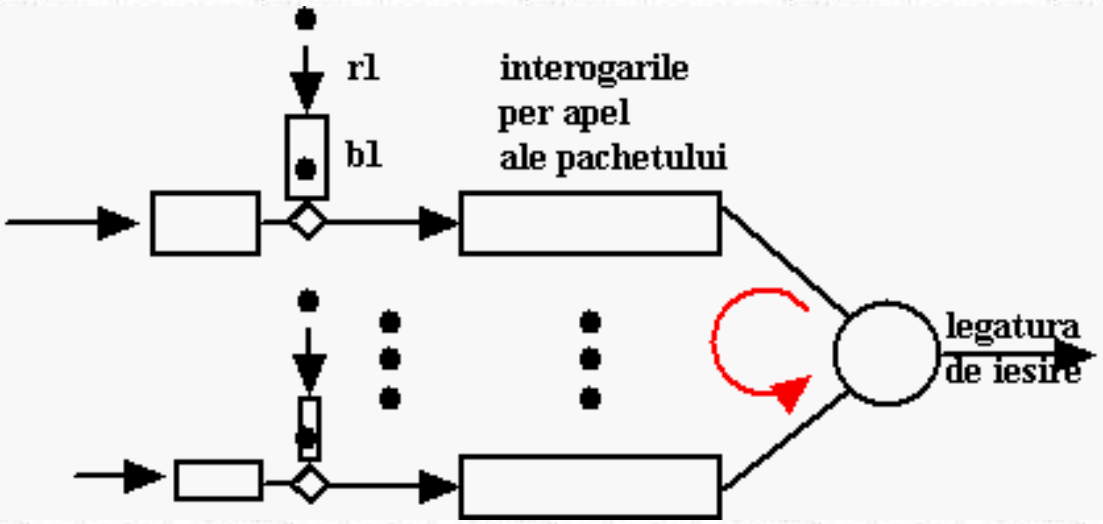
- partajare egala: 1/N din capacitatea legaturii la iesire, C
- partajare proportionala: fiecare apel i are numarul fi
  - i partajeaza C: fi/ sum(toate j cu date in asteptare, fj)

Protectie: WFQ *separa* manevrarea diferitelor sesiuni

- comportarea incorecta sau sesiunea lacoma nu fac decat sa se autopedepseasca

WFQ + rezervor cu jetoane = garantii privind intarzierea

Scenariu simplu: sursele controlate din rezervorul cu scapari alimenteaza multiplexorul WFQ.



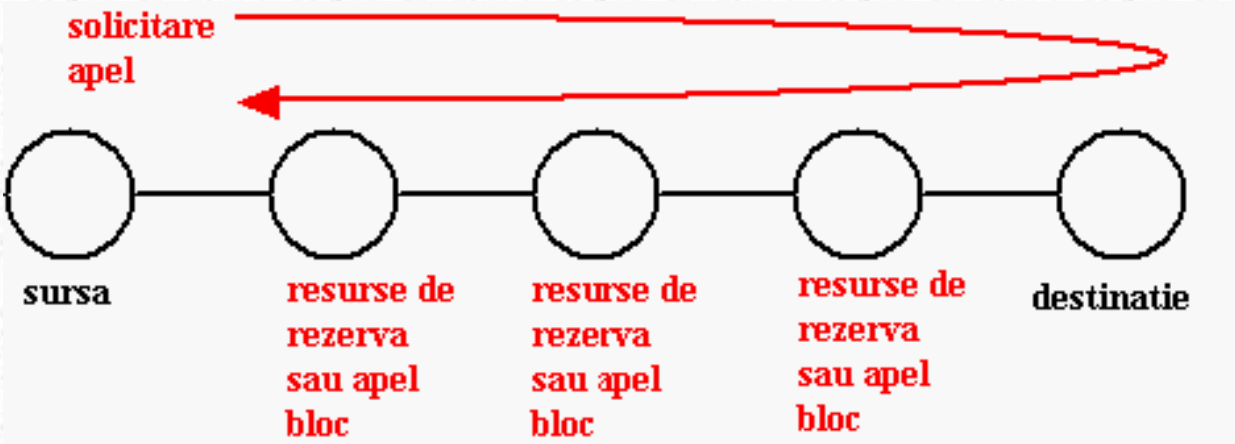
Reapel: f1/sum(toti j, fj) * C: largimea de banda minima garantata pentru sesiunea 1.

- cantitatea de trafic a sesiunii 1 < r1*t + b1
- distrugerea cantitatii b1 goleste rezervorul, introduce asteptarea 1
- timpul pina cand ultimul pachet este servit (intarzierea maxima a pachetului) este b1 / f1
- lungimea cozii scade de la b1 (considerand r1 < f1/sum(toti j, fj) * C)

Analizele pot fi extinse la retelele de multiplexoare.

Rezervarea resurselor

Protocolul de setare a apelului trebuie sa realizeze admisia apelului, sa rezerve resurse la fiecare ruter pe calea cap-cap



Q.2931: Protocolul de setare a apelului ATM

- expeditorul initiaza setarea apelului trecand Mesajul de Setare a Apelului prin limitele UNI (prin retea)
- reteaua returneaza imediat indicatii privind Procedura de Apel
- reteaua trebuie:
  - sa aleaga calea (rutarea)
  - sa aloce resurse in lungul caii (nota: cuplarea CS si a rutarii)
- reteaua returneaza indicatii despre conexiunea reusita/esuata expeditorului.

RSVP: Protocolul de rezervare a resurselor Internet (Internet Resource Reservation Protocol)

Consideratii despre un protocol Internet de rezervare:

- multistatia ca o categorie privilegiata
  - cantitati mari de destinatari heterogeni
  - heterogenitatea in largimea de banda disponibila expeditorului
  - heterogenitate in cerintele SC ale destinatarului (diferind de cerintele de intarziere)

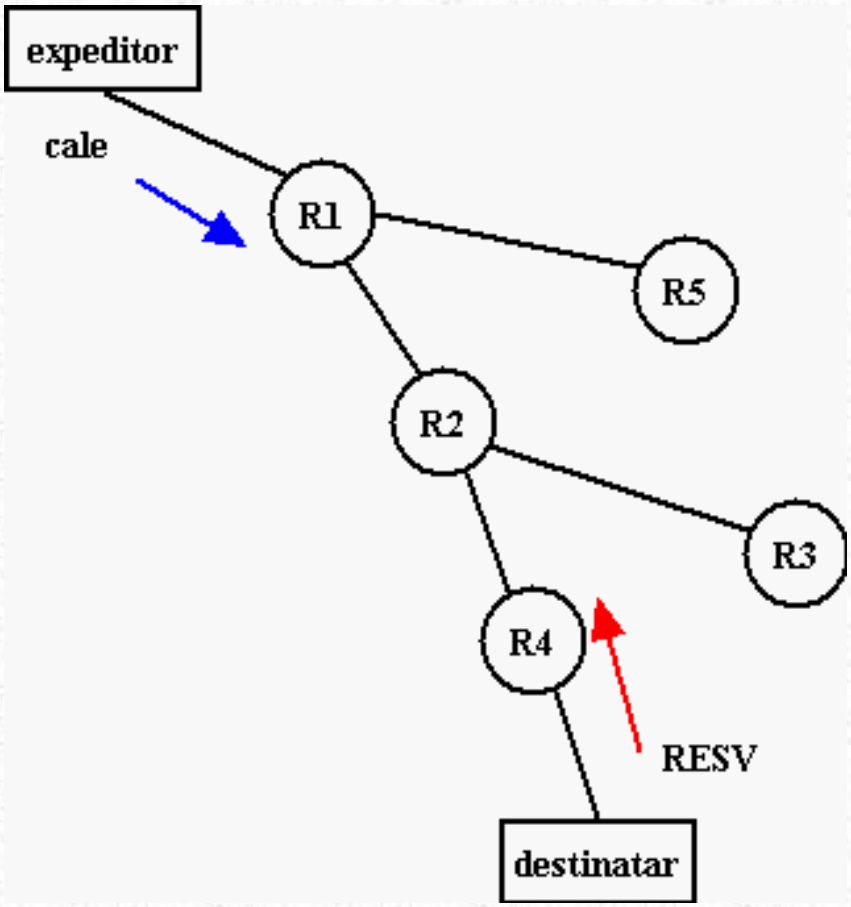
- orientarea destinatarului: lasa destinatarii sa conduca procesul de rezervare
  - scalabilitate
  - heterogenitate
- starea de apel (rezervatii) din rutere nu trebuie sa fie stearsa an mod explicit
  - va dispare daca nu este "improspatata" de destinatari.

Setarea apelului in RSVP

Expeditorul trimite Tspec in afara pe calea multistatie in mesajul PATH

Destinatarul trimite mesajul RESV anapoi in sus:

- contine cerinta (Rspec) perntru Tspec al expeditorilor si SC al destinatarului
- ruterele din calea inversa rezerva resursele necesare pentru a satisface SC al destinatarului

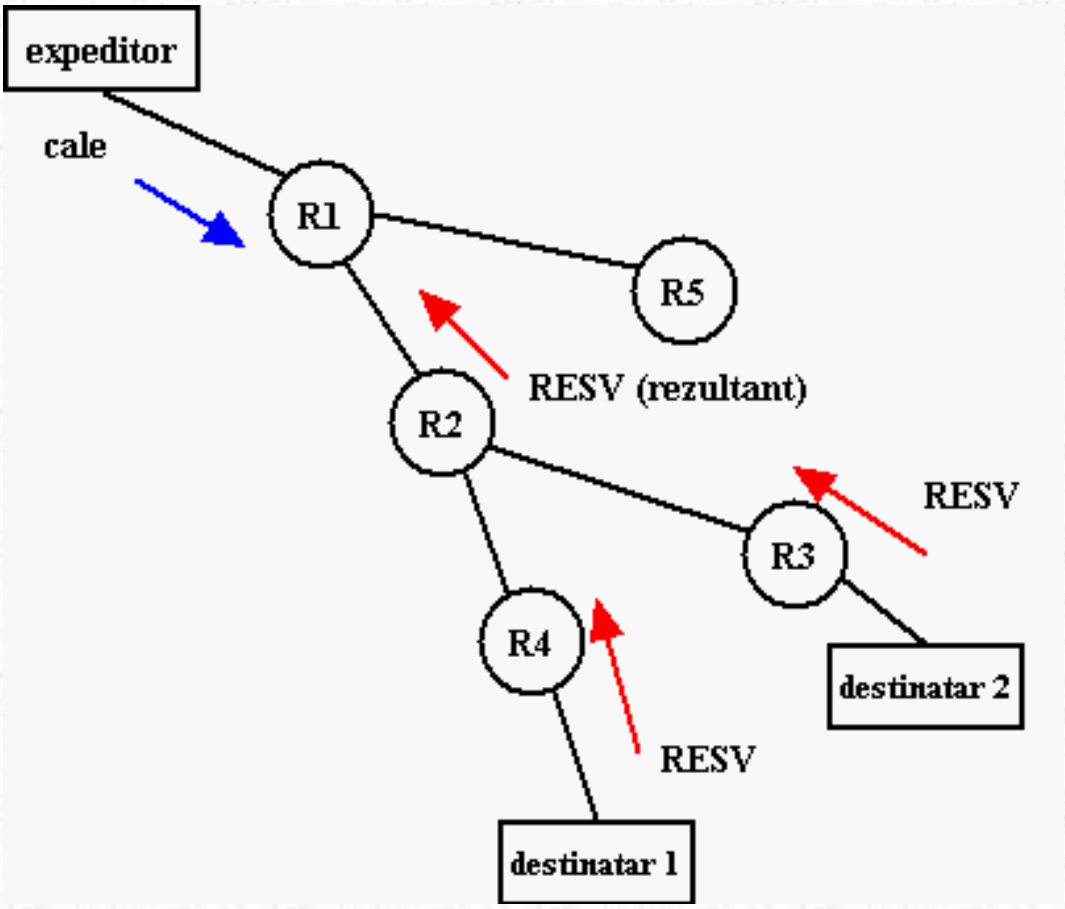


RSVP: Multistatie

Destinatari multipli:

- pot avea diferite cerinte SC
- rezervarile de resurse se cumuleaza in timp ce RESV urca

- resursele trebuiesc alocate pentru a satisface cerintele cele mai stricte ale destinatarilor de jos
  - de ex.: destinatarul 1 rezerva in primul rand resurse pentru max 100 intarzieri
  - daca SC al destinatarul 2 are intarzierea de max 200 ms, nu va fi nicio noua resursa la R2, R1
  - daca SC al destinatarului 2 este 50 ms, sunt necesare mai multe resurse la R2, R1



Se poate lucra cu mai multi expeditori ca si cu diferite "stiluri" de rezervare a resurselor

- de ex., rezerva suficiente resurse pentru cazul in care toti expeditorii actioneaza simultan
- suficiente resurse pentru cazul a doi expeditori simultan
- poate determina in mod dinamic care din cele N fluxuri sunt directionate in jos (comutare in retea).

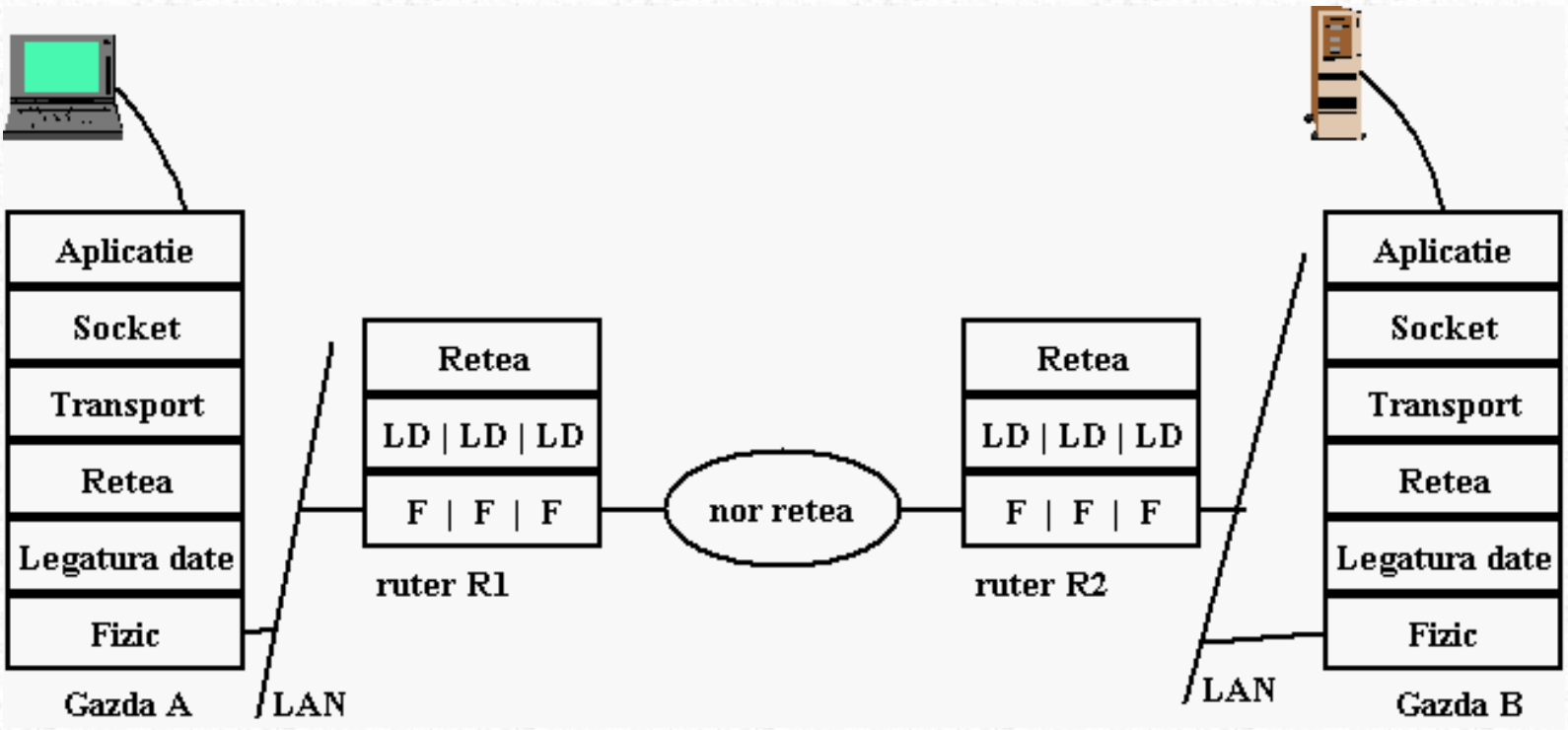
Back | Top | Next

Retele, protocoale si servicii asigurate prin retelele de calculatoare

Perspective

12. Perspective

Sumar: sa urmarim un pachet



Aplicatie

- utilizatorul introduce URL in browserul WWW
- URL poate fi text, audio, video, cu diverse cerinte
- browserul determina numele gazdei, foloseste DNS pentru a obtine adresa IP a serverului

IPA

- browserul (client) creaza socketul fluxului, socket()

- clientul apeleaza connect(), portul 80 al serverului
- dupa raspunsul de la connect(), va apela rcvfrom() pentru a citi datele returnate

Nivelul transport

- apelul connect() determina stabilirea conexiunii TCP
- se alege numarul initial de secventa
- se genereaza pachete SZN, adresa IP a serverului, port 80
- TCP formeaza pachetul, proceseaza checksum
- TCP apeleaza IP (fara control al congestiei pe SYN), trecand informatiile privind pachetul SYN si adresa IP

Nivelul Retea

- sursa Ip a adresei, adresa de destinatie in pachetul IP
- directionarea IP prin consultarea tabelei de rutare
  - tabela de rutare procesata de RIP, sau protocolul intradomeniu OSPF
  - tabela de rutare da adresa IP si interfata locala care trebuie obtinuta urmatorului ruter (din LAN-ul sau), R, pe ruta la destinatie
- ruleaza ARP pentru a obtine adresa fizica 802.3 corespunzatoare adresei IP a lui R
- ARP va genera pachetul de difuzare Ethernet pe LAN, solicitand lui R sa rpspunda cu adresa sa fizica
- R transmite adresa sa fizica

Nivelul Lagatura Date

- pachetul TCP SYN (in interiorul pachetului IP), precum in pachetul Ethernet transmis in LAN folosind protocolul Ethernet
- este posibil sa fie implicata si puntea transparenta (nearatata)

Nivelul Fizic

- pachetul Ethernet transmis la 10, sau 100 Mbps

La ruterul R1:

- nivelul fizic primeste pachetul Ethernet, il transmite in sus
- nivelul legatura date proceseaza checksum, OK, inlatura pachetul IP, il trece in sus
- nivelul retea consulta tabela de rutare
  - tabela de rutare a fost procesata folosind rutarea interdomeniu BGP
  - se trece pachetul IP in jos catre nivelul legatura date ...

La ruterul 2:

- pachetul DLC soseste din "net", este trecut in sus la nivelul retea
- nivelul retea determina interfata de iesire sa ajunga la gazda B
  - tabele procesate prin protocolul de rutare intradomeniu RIP sau OSPF
- pachetul Ethernet DLC transmis (R2 stia/aflase adresa nivelului fizic a lui B)

La gazda B:

- pachetul Ethernet soseste, se verifica sum (OK), se trece in sus la IP
- nivelul IP extrage pachetul TCP, demultiplexeaza pana la TCP (nota: nu pachete UDP)

- TCP vede pachetul SYN
  - serverul trebuie sa aiba deschis dinainte socketul si sa fi facut accept(), altfel SZN este picat
  - TCP determina fereastra de control al fluxului, alege numarul initial de secventa
  - transmite SZNACK inapoi

La gazda A:

- SYNACK eventual receptionat
- transmite ACK pentru nivelul transport la B
- se misca spre starea stabilita
- retur de la apelul de sistem connect()

Epilog:

- Gazda A poate acum sa efectueze read()
- Gatda B (dupa primire ACK) poate eventual sa transmita URL solicitat
  - va folosi TCP
  - controlul congestiei si al traficului
  - R1, R2 si A, B (nivel retea si mai jos) actioneaza exact la fel cu datele ca si cu pachetul SYN

Perspective

Tendinte generale:

- ubiguitatea comunicatiilor
  - aplicatii disponibile pentru retea (de ex., termostat IP)
  - probleme importante cu dimensiunea globala: sute de milioane de dispozitive conectate in retea
- mobilitatea este importanta: oamenii se misca si au nevoie sa comunice
- multimedia este importanta: modalitate de comunicare a oamenilor
- viteze ale legaturilor in crestere, dar largimea de banda este limitata an viitorul apropiat
  - creste numarul "utilizatotrilor"
  - necesitati crescute ale largimii de banda pentru aplicatiile disponibile
- securitatea, un aspect critic
- necesitatea unei largimi mari de banda pentru acasa (ADSL, modemuri cablu), o problema majora pentru viitor
  - jocuri, VR, educatie, informare, distractie
  - unirea retelisticii cu telefonia
  - distractie in retea cu difuzie (TV) si WWW
- agenti, alte tehnologii pentru afaceri necesitand mari cantitati de date distribuite si intr-o continua schimbare

Retelistica va juca un rol central in toate sistemele viitoare de calcul.

E-mail: engineering@teleactivities.net

13. Bibliografie

Carti

1. Ion Banica, "Retele de comunicatii intre calculatoare", Editura Teora, 1998
2. Andrew S. Tanenbaum, "Retele de calculatoare", Computer Press Agora, Editia a treia, 1998
3. Jim Kurose, "Computer Networks", Department of Computer Science, Universitz of Masachusetts, USA
4. Crypto Policy Perspectives: S. Landau et al., Aug 1994 CACM
5. Internet Security, R. Oppliger, CACM May 1997
6. D. Flanagan, "Java in a Nutshell", O'Reily and Associates, 1996
7. John Larmouth's book "Understanding OSI" : chapter 8: ASN.1 , role of ASN.1 in next generation http
8. Neufeld and Y. Yang, "An ASN.1 to C compiler," *IEEE Trans. Software Engineering*, Oct. 1990
9. C. Huitema and A. Doghri, "Defining Faster Transfer Syntaxes for the OSI Presentation Protocol," *ACM Computer Communication Rev.* Oct. 1989
10. Olaf Kirch, "The Network Administrators' Guide"
11. Stephen Northcutt, Donald McLachlan, Judy Novak, "Network Intrusion Detection: An Analyst's Handbook", New Riders Publishing, 2000
12. David Groth, "Network+ Study Guide" (3rd Edition), Sybex, 2001
13. Jeff Doyle, "Routing TCP/IP (CCIE Professional Development)", Cisco Press, 1998
14. Craig Hunt, Gigi Estabrook, "Tcp/Ip Network Administration", O'Reilly & Associates,

1998

Resurse Internet

1. Understanding Networking Technologies, http://www.netguru.net/courses/ntc
2. www.eff.org
3. Ericsson, Wireless LANs, http://www.ericsson.com/wlan
4. Symbionics, Wireless Networking, http://www.symbionics.com/
5. IEEE Computer.org A Short Tutorial on Wireless LANs and IEEE 802.11, http://computer.org/students/looking/summer97/ieee802.htm
6. Andrew S. Tanenbaum, www.cs.vu.nl/~ast/
7. Computer Networks, www.math.grin.edu/~rebelsky/Courses/CS364/2000S/
8. Computer Networks, www.comp.nus.edu.sg/~cs3103/
9. Computer Networks, www.cs.utexas.edu/users/lam/cs356/
10. Data Communication and Computer Network, www.cs.purdue.edu/homes/park/cs536.html
11. Computer Networks. A Systems Approach, www.cs.virginia.edu/~zaher/classes/CS457/
12. Ethernet: Distributed Packet Switching for Local Computer Networks, www.acm.org/classics/apr96/
13. Introduction to Computer Networks, www.cs.technion.ac.il/~cs236334/
14. Data Communications, Computer Networks, and Open Systems, www.aw.com/catalog/academic/product/1,4096,020142293X,00.html

Back Top

Retele, protocoale si servicii asigurate prin retelele de calculatoare

- **Teleactivities Web Portal**
  - **Internet Business Services**: Servicii de afaceri pe Internet (RO), Internet Marketing, Web Design, Word Processing, Translation, Secretarial Services, Desktop Publishing, Consulting, Order Form
  - **Shop**: Cartea "Telelucru (Inovare - Internet - Oportunitati) (RO)
  - **Romania Business Opportunities**
    - **Introduction**: Transitions, Challenges, Relations, EU Integrations
    - **Economic Summary**: Investment, EU Convergence
    - **Investment Climate**: Policy and Incentives, Privatisation, Markets and Trade
    - **Major Sectors**: Automotive, Telecommunications, Oil and Gas, Energy, Transport, Tourism, Agriculture and Food
    - **Financial**: Non-bank Institutes
    - **The Friends of Romania Forum**: Post a message, Search for messages
  - **Romania Business Directory**
    - Add URL
    - Adult
    - **Arts**: Animation, Artists, Literature, Music, Painting, Photography
    - **Business**: Agriculture, Business Services, Companies, Consulting, E-Commerce, Employment, Finance, Industries, Insurance, Management, Marketing, Opportunities, Real Estate, Small Business, Trade, Training and School, Transport
    - **Computers**: CAD, Companies, Hardware, ITC, Internet, Multimedia, Programming, Publications, Software, Security
    - **Health**: Beauty, Cabinets, Healthcare, Hospitals and Clinics, Magazines, Management, Mental Health, Nutrition, Products and Shopping
    - Home
    - Kids & Teens
    - **News**: Ezines, Magazines, News Agencies, Newspapers, Television

- **Recreation**: Autos, Competitions, Foods, Games, Humor, Movies, Postcards, Restaurants and Bars, Travel and Tourism, Underground
- **Reference**: Directories, Education, Museums
- **Regional**: Countries
- **Science**: Archeology, Physics, Research, Technology
- **Shopping**: Books
- **Society**: Astrology, Culture, Ethnicity, Genealogy, Government, History, Law, Military, Organizations, Paranormal, People, Politics, Relationship, Religion, Traditions
- **Sports**: Martial Arts, Soccer
- New Sites

- **e-Business**
  - **Afaceri / Internet** (RO): Editoriale, Interviuri, Articole, Stiri, Comunicate, Evenimente, Proiecte, Ghiduri e-Business, Anunturi Gratuite
  - News, Events, Studies
  - **e-Business Guides**: Online Business Guide, Guerilla Web Promotion and Marketing, Web Design Guidelines
  - **Online Calculators**: Foreign Currency Converter, Loan Calculator, Savings Calculator
  - **The Virtual Romanian Market**: Post a message, Search for messages
- **Companies**: SC AmaCom SRL
- **Jobs**
  - **Telework**: Telelucru (RO), Opportunities, Events
  - **Telejobs**: Homework
  - **Telejobs Forum**: Post a message, Search for messages
- **Personal Home Pages**: Nicolae Sfetcu
- **The Virtual Friends**: Post a message, Search for messages
- **Dracula Zone**: Zona Dracula (RO), Background, The Youth, The Reign, The Death, The Castle, The Myth, Chronology, Links, The Vampires, Dracula's Forum
- Afaceri / Internet Ezine (RO)
- The Virtual Office Ezine
- **Resources**: Web Sites, Directories and Links, Web Forums, Mailing Lists, Web Rings, Ezines, E-books, Awards, Contests, Links
- Contact

- **Business Center**
  - **AmaCom Internet Center SRL: Internet Business Services**
    - Centre Internet AmaCom (FR)
    - **Centrul Internet AmaCom** (RO): Despre noi, Servicii de afaceri, Solutii TI, Clienti, Contact
    - About us
    - **Business Services**: Internet Marketing, Web Design, Desktop Publishing, Word Processing, Secretarial Services, Bookkeeping, Consulting, Order
    - **IT Solutions**: Hardware, Software, Computers Networks, Training
    - Customers, Contact
  - **AmaNav SRL: The leader of the Lower Danube shipping companies**
    - **The Company**: Management, Customers, Careers
    - Danube
    - **Services:** Customer value, Technology, Partners
    - Information, Contact
    - **Online Trade Forum**: Search messages, Post messages
- **DeskTop Services**
  - **Marketing**: Links
  - **Documentation**: Romanian Links, Business Links, How to info
  - **Desktop Publishing & Word Processing**
  - **Web Design**: Links
  - Translation
  - **Ask me! - You post questions, I post answers**: Post answer, Search messages
- **Business Links Directory**
  - **Business Links Directory**: Accounting, Agriculture, Art, Companies, Construction, Employment, Entertainment, Finance, Health, Industries, IT&C, Marketing, Media, Opportunities, Regional, Small Business, Science, Services, Shopping, Society, Sport, Trade, Training, Transport, Travel
  - **Free Advertising Forum**: Post messages, Search messages
  - **Bad Companies Forum**: Post messages, Search messages
- **Romania Trade Forum**: Post messages, Search messages
- **Forumul Afaceri / Internet** (RO): Posteaza mesaje, Cauta mesaje
- **Home Business Links Forum**: Post messages, Search messages, Favorite Links
- **European Telework Online: Romania**

- **Telework**
  - Ghidul telelucratorului (RO)
  - Articole (RO): Telelucru si teleprotectie, Marketingul pe Internet, Tips & Tricks pentru Outlook Express, Impactul asupra mediului a telelucrului si teleactivitatilor
  - **Telework**: Opportunities, News
- **Jobs Forum:** Post messages, Search messages
- **Best Clipart**: Just for fun!, Images, Sounds, Movies, Clipart Forum
- **Free Classified Ads Forum**: Post messages, Search messages
- **Nicolae Sfetcu - Multimedia Business Card**
  - (RO): Introducere, Orasul, Compania, Resume, Contact
  - Introduction, The City, The Company, Resume, Contact
  - **Useful**: MathTimeFunction, Area Equivalents, Foreign Currency Convertor, Loan Calculator, Savings Calculator, Biorhythm, Tetris Game
- **Engineering**
  - **Inginerie** (RO)
    - **Calculatoare**: Networks, protocols and services provided through the computers networks
    - **Electrochimie**: Cinetica
    - **Electronica**: Cartela electronica programabila (Smart Card)
    - Software pentru ingineri
  - **Computers**
  - **Electrochemistry**: Preview, Corrosion, Ellipsometry, Software
  - **Electronics**
    - **Design**: Microstrips, Core Inductors, Operational Amplifiers, Telephones, Calculators
    - **Circuits**: Amplifiers, Analogue Circuits, Automotive, Comparators, Converters, Detectors, Digital Circuits, Drivers, Filters, Interfaces, Measuring Circuits, Oscillators, Protections, Radio, Sound, Supplies, Telephone, Timers
    - **Catalogues**: PIC16C84
    - **For Sale**: Condensers, Diodes, Integrated Circuits, Optoelectronics, Potentiometers, Rectifier Bridges, Resistors, Thyristors and Triacs, Transistors
  - **Telecommunications**: International Conference on Telecommunications 2001, Tutorials
  - **Software for engineers**: Chemistry, Metallurgy, Miscellaneous
  - **Online Calculators**: Area Equivalents, Ohm's Calculator, Parameter Functions
  - **Engineers Forum**: Post messages, Search messages
- **Free Books**: Business, Computers, Industry. Marketing, Science, Free Books Forum,

- **The Environmental Site: Remat SA Mehedinti, The Recycling Company**
  - **Remat SA Mehedinti, Compania de reciclare a materialelor** (RO)
    - Despre noi, Colectare, Procesare
    - **Protectia mediului**: Sistemul de Management al Mediului, Resurse, ISO14000, Prelucrare otel, Proiecte
    - **Dezvoltarea Durabila**: Constructii ecologice, Dezvoltarea ecologica, Planificarea folosirii terenului, Masurarea progreselor, Anticiparea calamitatilor, Energia comunitara, Transportul durabil, Afaceri durabile, Finante, Valori rurale, Eficienta materialelor, Eficienta apei
    - Ghidul ONG: Facilitati globale de mediu, Stiri
  - About us, Collecting, Processing
  - **Environmental Protection**: Environmental Management System, Resources, Energy, ISO14000, Steel Scrap
  - **Sustainable Development**: Green Buildings, Green Development, Land Use Planning, Measuring Progress, Disaster Planning, Community Energy, Sustainable Transportation, Sustainable Business, Finance and Sustainability, Rural Issues Introduction, Materials Efficiency, Water Efficiency
  - NGO Guide: The Global Environment Facility, News, Links
  - **Forumul Bursa Materialelor Reciclabile** (RO): Posteaza mesaje, Cauta mesaje
- **Non-Governmental Organizations Romania (NGO Romania)**
  - **ONG Romania** (RO)
    - **Societatea Romana pentru Telelucru**: Despre SRT, Statutul, Membrii, Inscriere, Contact
    - **ONG Romania**: Alianta Romanilor din Ukraina (ACDR), Asociatia Dusmanii Hemiblegieie Bihor Romania, Asociatia Generala a Inginerilor din Romania, Centrul de Consultanta Ecologica Galati,. Fundatia pentru Armonizarea si Optimizarea Relatiilor Interumane, Gnomes'Land, Fundatia Pentru Dezvoltare Comunitara "RURAL FORUM", Asociatia de Tineret pentru Invatamant si Stiinta Solaris, Comunitatea Rurala Todireni, Asociatia "Millennium Center", Formular solicitare pagina gratuita ONG
    - **Activitatea ONG**: Programe, Proiecte, Granturi, Parteneriate, Comunicate, Evenimente, Training, Competitii, "Lista de discutii", Anunturi
    - Societatea Informationala
    - **Ghiduri ONG**: Ghidul finantarii si participarii in programele UE a organizatiilor neguvernamentale din Europa Centrala si de Est (ECE) si a Noilor State Independente (NSI), Ghidul organizatiilor de mediu. Facilitati globale de mediu. Ghid pentru organizatiile neguvernamentale

- Noutati, De la cititori, Oportunitati, Recunoastere, Design, Contact, Confidentialitate
- **Romanian Telework Society**: Statute, About SRT, Mission, SRT Board, FAQ, Membership, Media/Press, SRT Awards, SRT Branches, Events, SRT Publications
- **Romanian NGOs**: The General Association of Engineers in Romania (AGIR), The Ecological and Tourism Club "Floarea reginei" Sinaia, ICAR Cultural Association, Application form for a free Romanian NGO page
- **NGOs Activity**: Programmes, Projects, Grants, Parteneriat, Training & Events, Press Releases, Announcements
- Information Society
- **NGO Guides**: Guide to Funding and Participation in European Union Programs for Non-Governmental Organizations in Central and Eastern Europe and the Newly Independent States, Concise Guide to Human Rights on the Internet, The Global Environment Facility - A Guide for NGOs
- Jobs Opportunities, News from NGO, ONG Romania Ezine (RO)
- **Organizations**: The Inspectorate for the Environmental Protection Mehedinti, "Fituica": Foaia elevilor de la Scoala cu clasele I-VIII "Octavian Goga" din Jibou (RO)
- Privacy Statement

Romanian Telework Society

- About us, SRT Statute
- (RO): Despre noi, Membrii SRT, Statutul SRT, Inscriere in SRT

The General Association of Engineers in Romania (AGIR)

- **Asociatia Generala a Inginerilor din Romania (AGIR)** (RO): Prezentare, Statut, Filiale, Propuneri, Anunturi, EuroRecord, Inscriere
- Preview, Statute, Branches, Subscribe
- **Forumul AGIR** (RO): Posteaza mesaje, Cauta mesaje

Balkan Forum: Post messages, Search messages

Romanian Forum: Post messages, Search messages

The Official Site of Drobeta Turnu Severin City

- **Drobeta Turnu Severin**: Severin Forum, Feedback, Guest Book, Geography, History, Economy, Education, Culture, Travel, Maps, Entertainment
- Drobeta Turnu Severin (FR)
- (RO): Geografia, Istoria, Economia, Educatia, Cultura, Turism, Harti, Proiecte, Ezine, Editoriale, Stiri, Anunturi gratuite, Locuri de munca, De la cititori, Umor, Severin on-line Ezine

"Nadejdea" Newspaper: latest news from RIA NOVOSTI, comments and analyses from Russia (RO)

- **Online Security**
  - **Hacking**: Harmless hacking, Cracking, IRC hacking, Phreaking, Social engineering
  - **Attacks**: Denial of attacks
  - **Intelligence Services**: Organizations, Economic intelligence, Documentation, News, Miscellaneous
  - **Security**: Cryptography, Unix security
  - **Online Security Forum**: Post messages, Search messages
  - Online Services
- **Forumul Impotriva Abuzurilor pe Internet (FIAI)** (RO): Posteaza mesaje, Cauta mesaje
- **Proiectul Anticoruptia** (RO):
  - **Directorul persoanelor corupte din Romania**: Adauga o noua persoana, Administratie, Economie, Justitie, Ministere, Sanatate
  - **Directorul cazurilor de coruptie din Romania**: Adauga un nou caz, Contrabanda, Economie, Justitie
  - **Top 100 Coruptia**: Voteaza, Top 100 a celor mai corupte persoane din Romania, Top 100 a celor mai corupte institutii din Romania
  - **Forum web Coruptia**: Transmite mesaj, Cauta mesaje
  - Legislatie, Structuri specializate, Studii, rapoarte si analize, Articole, Diverse, Forum FAQ, Linkuri
- **Free gifts for all**
  - **Cards**: Greeting Cards
  - Just for fun!, Tests, Adults only, Online games
- **Confession Forum, The Virtual Friendship**: Post messages, Search messages
- **Ezines, Newsletters, Mailing Lists**
- **Web Rings**

Top

E-mail: engineering@teleactivities.net