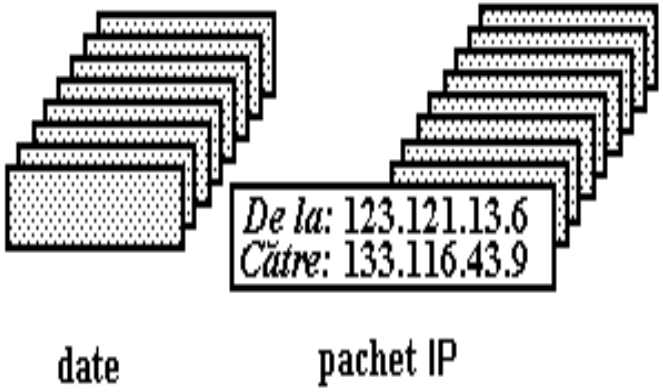
Nivelul retea

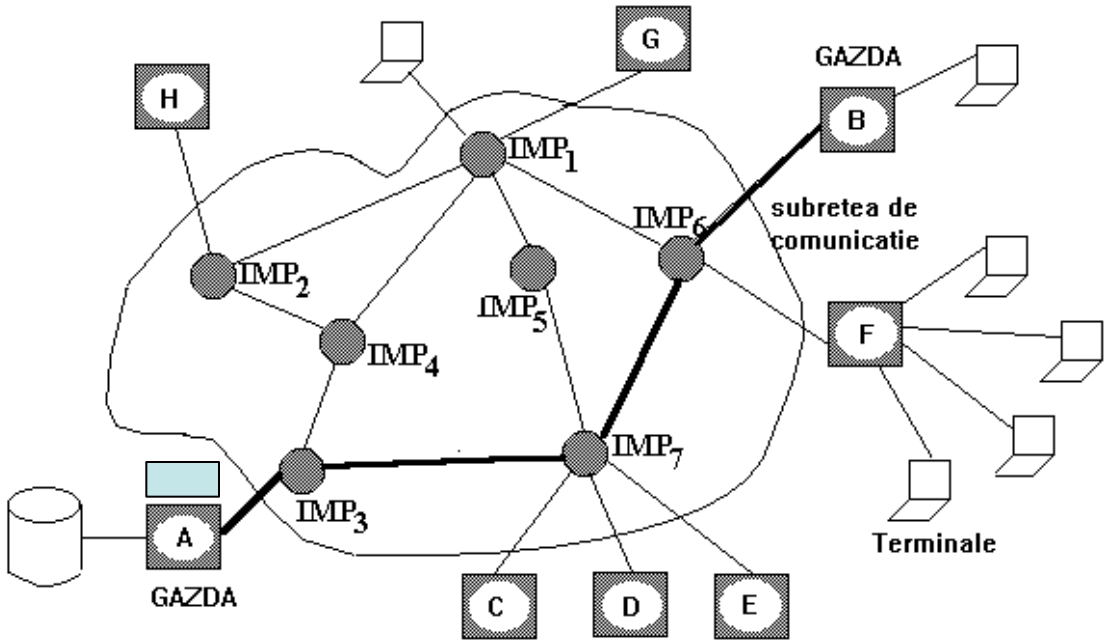
Principiul comunicării în Internet

Inspirat din sistemul poştal



Drumul între utilizatorii A si B trece prin ruterele IMP3, IMP7 şi IMP6

IMP = Interface Message Processor – denumire folosita in ARPANET



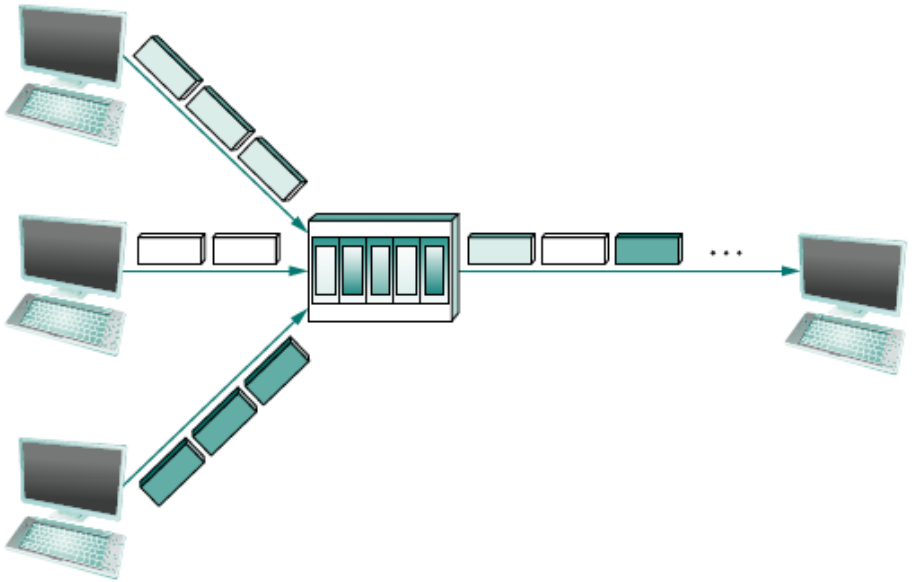
De ce este nevoie de pachete?

Legaturile unei retele pot fi folosite simultan de transmisii paralele ale mai multor perechi de noduri

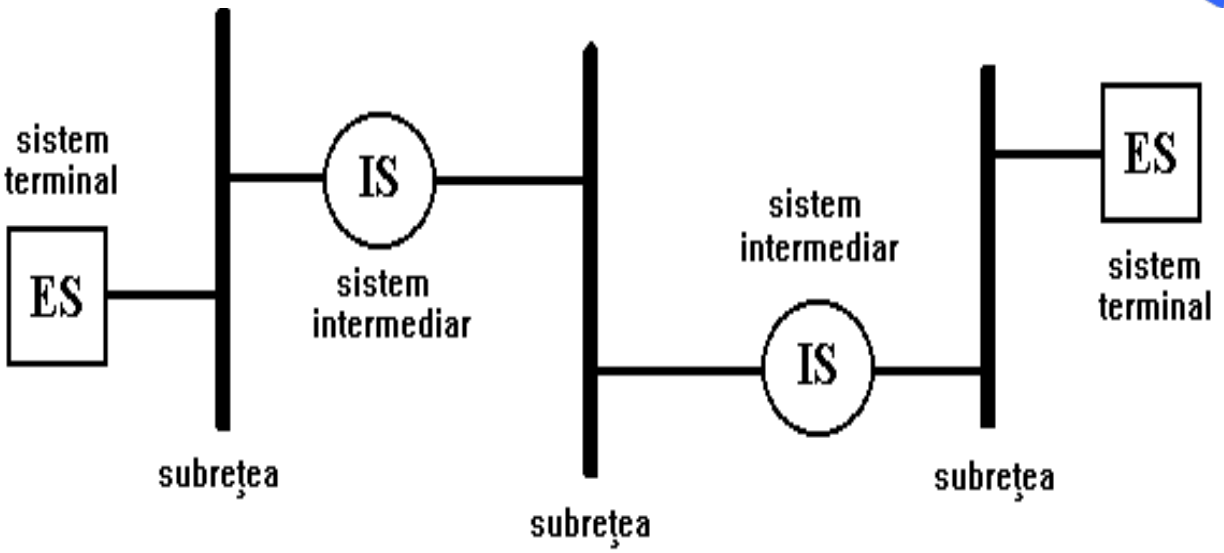
Folosirea partajata a legaturilor se face prin multiplexare

- sloturi de timp sunt alocate in proportii egale diverselor perechi - STDM – Synchronous Time Division Multiplexing
- subcanale de frecvente diferite sunt alocate diverselor transmisii – FDM – Frequency Division Multiplexing

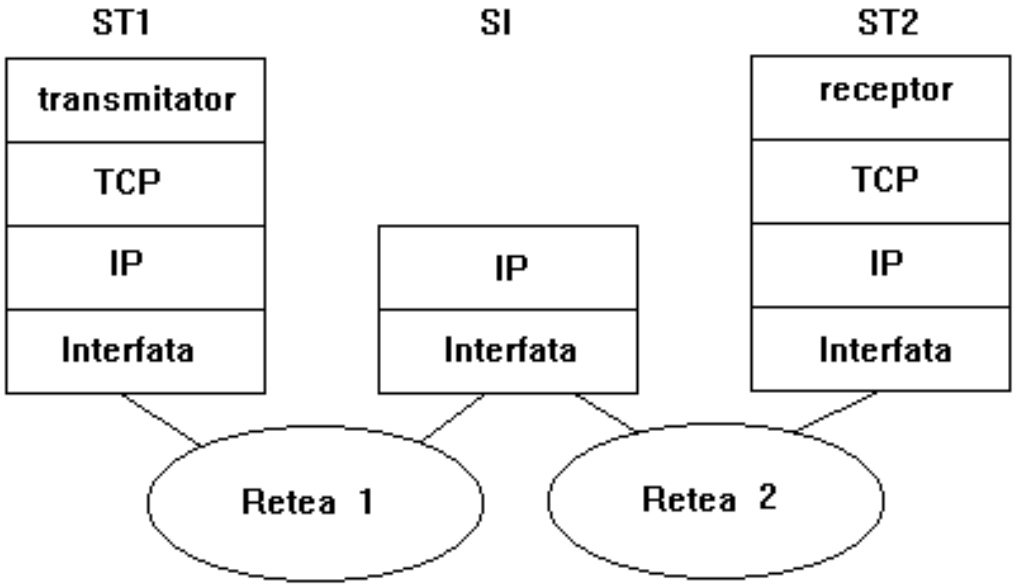
- statistic – legatura este alocata la cerere diferitelor transmisii;
- pentru a evita acapararea legaturii de o singura transmisie, dimensiunea blocului de date este limitata superior →pachet



Modelul Internetului



Nivelele străbătute de pachete



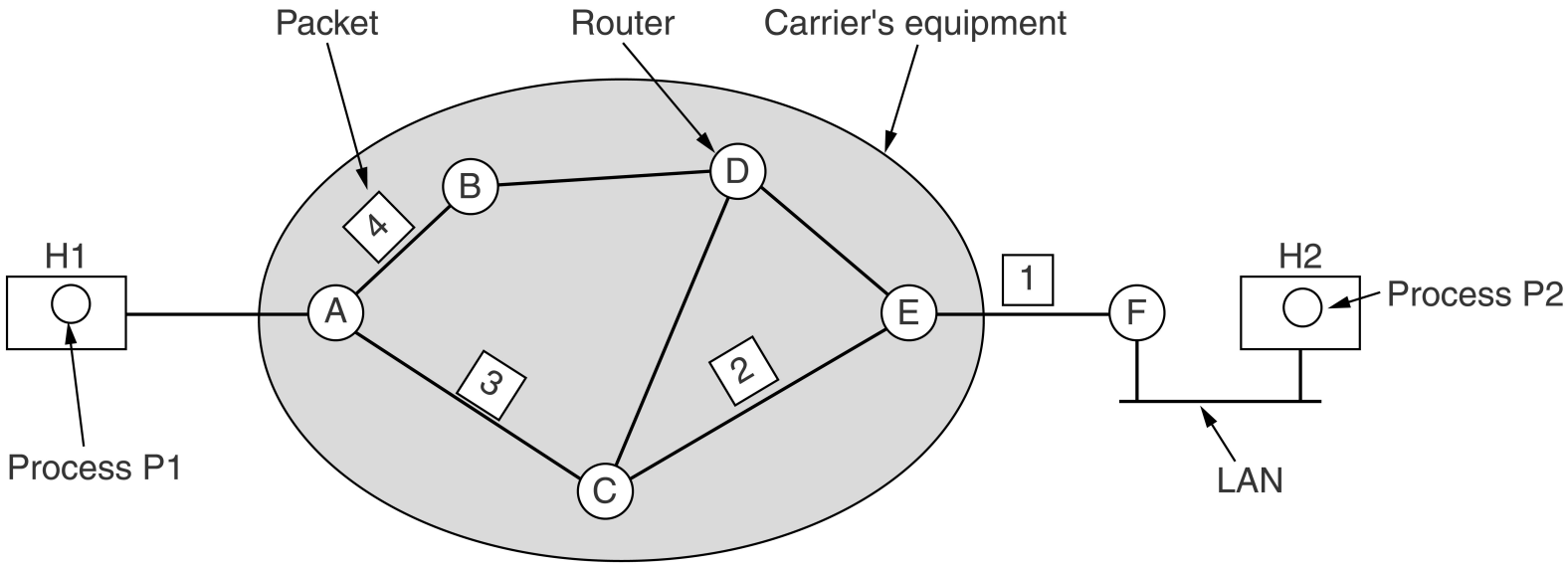
Funcţiile nivelului reţea

- dirijarea pachetelor
- adresarea

Aspecte principale

- servicii
  - orientate pe conexiune
  - ne-orientate pe conexiune
- organizarea internă
  - datagrame
  - circuite virtuale
- dirijarea
  - retransmiterea pachetelor – forwarding
  - algoritmi de dirijare actualizeaza tabelele de dirijare - rutare
  - politici

Organizarea internă - datagrame



Folosita de pachetele 1, 2 si 3

A's table

initially		later	
A	–	A	–
B	B	B	B
C	C	C	C
D	B	D	B
E	C	E	B
F	C	F	B

Dest. Line

C's table

A	A
B	A
C	–
D	D
E	E
F	E

E's table

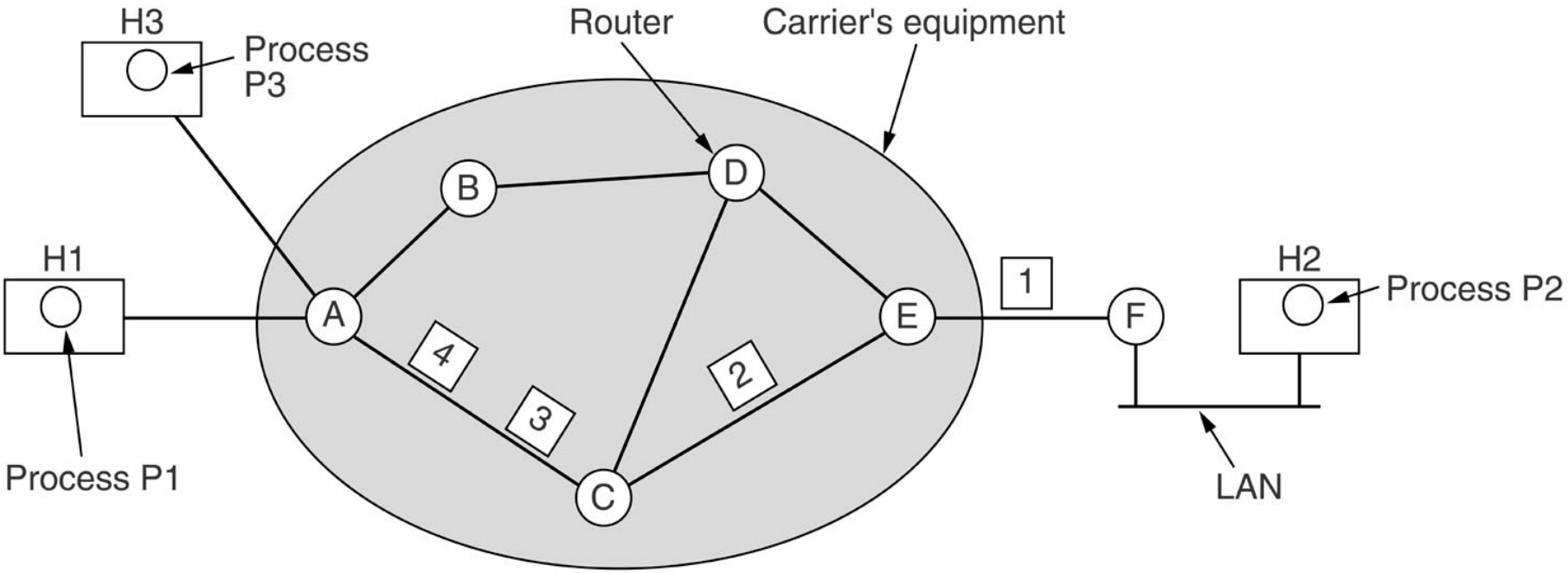
A	C
B	D
C	C
D	D
E	–
F	F

Folosita de pachetul 4

Caracteristici

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

Organizarea internă – circuit virtual



A's table

In		Out	
H1	1	C	1
H3	1	C	2

C's table

A	1	E	1
A	2	E	2

E's table

D	1	F	1
C	2	F	2

Nodul A re-numeroteaza circuitul virtual

Caracteristici

- Organizare bazata pe conexiune
  - Transferul incepe dupa stabilirea conexiunii
  - Se pot aloca resurse la stabilirea conexiunii (memorie tampon pentru pachete)
- Transmitatorul stie
  - ca exista o conexiune
  - ca receptorul este pregatit sa primeasca pachete
- Se pastreaza ordinea pachetelor
- Se poate controla fluxul

- Folosit in
  - X.25, Frame Relay, ATM
  - retele virtuale private (VPN)

Protocolul IP

- Are doua parti
  - o schema de adresare care permite identificarea oricarui calculator din Internet
  - modelul de datagrama pentru livrarea datelor
    - datagrama este denumirea adoptată de IP pentru pachet
- Modelul de serviciu – best effort
  - rețeaua face toate eforturile sa livreze pachetele la destinație
  - nu face nici o încercare să corecteze erorile

Protocol IPv4 – formatul pachetului

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

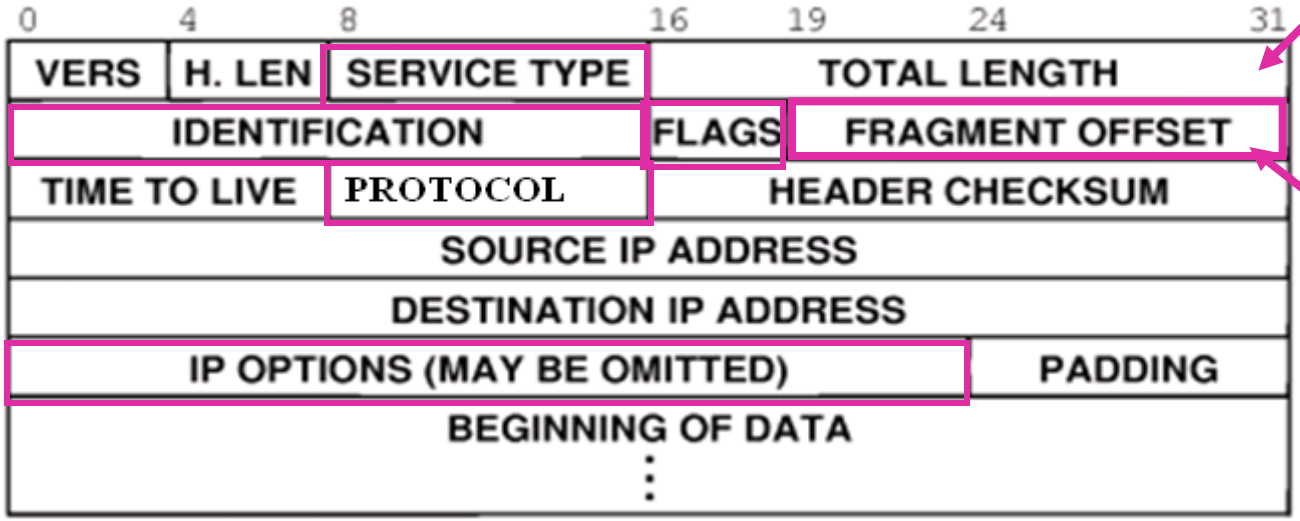
PROTOCOL = (TCP, UDP, etc.)

IDENTIFICATION datagrama de care aparţine fragmentul

FRAGMENT OFFSET pozitia fragmentului in pachet

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

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



max 65535 octeti

max 8192 fragmente a 8 octeti

Campurile de antet pentru fragmentare

(a) pachet ne-fragmentat

(b) pachetul fragmentat (3 fragmente)

(a)

Start of header				
Ident = x			0	Offset = 0
Rest of header				
1400 data bytes				

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

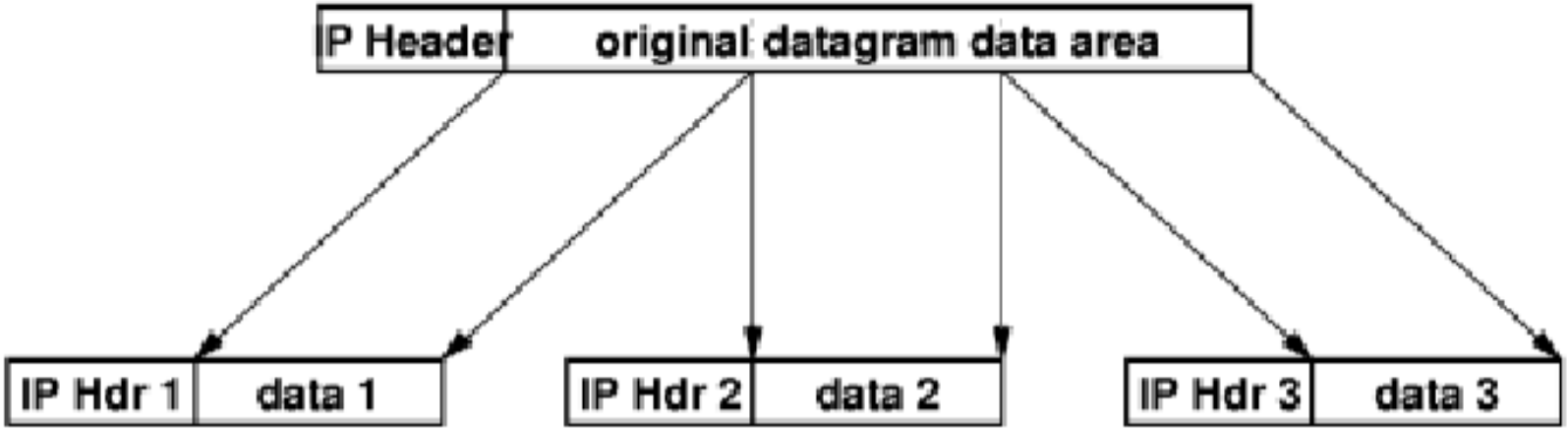
(b)

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

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

MTU – Maximum Transmission Unit

Fragmentarea se face la R_1



Reasamblarea se face la H_2

Adrese IP

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

Câteva adrese speciale

Prefix (rețea)	Suffix (gazdă)	Semnificație	Scop
toţi 0	toţi 0	acest calculator	Folosită la bootstrap
network	toţi 0	network	Identifică reţeaua
network	toţi 1	broadcast	broadcast în reţeaua specificată
toţi 1	toţi 1	broadcast	broadcast în reţeaua locală
127	orice	loopback	testare

Notații pentru adrese

binară 11000010 00011000 00010001 00000100

zecimală 194.24.17.4

Tabele de dirijare

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

<adresa_retea, adresa_nod>

Toate nodurile care au aceeași adresa_retea sunt situate in aceeasi retea fizica și pot comunica direct prin legătura de date (transmit cadre)

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

Rol ruter – primește un pachet și

- îl livrează gazdei de destinație (dacă este in aceeași rețea)
- altfel, il re-transmite (*forward*) către un alt nod NextHop

Folosește **tabela de** dirijare (**rutare**) care are intrări de forma

<adresa_retea, NextHop>

Algoritm de *forwarding* IP

Extrage <adresa_retea, adresa_gazda> destinatie din datagrama

Caută o intrare cu adresa_retea în tabela de dirijare

if adresa_retea apare in tabela de dirijare
    if adresa_retea indica o retea direct conectata **then**
        transmite datagrama direct la adresa_gazda
    else transmite datagrama urmatorului ruter (Next Hop)
else transmite datagrama unui ruter implicit

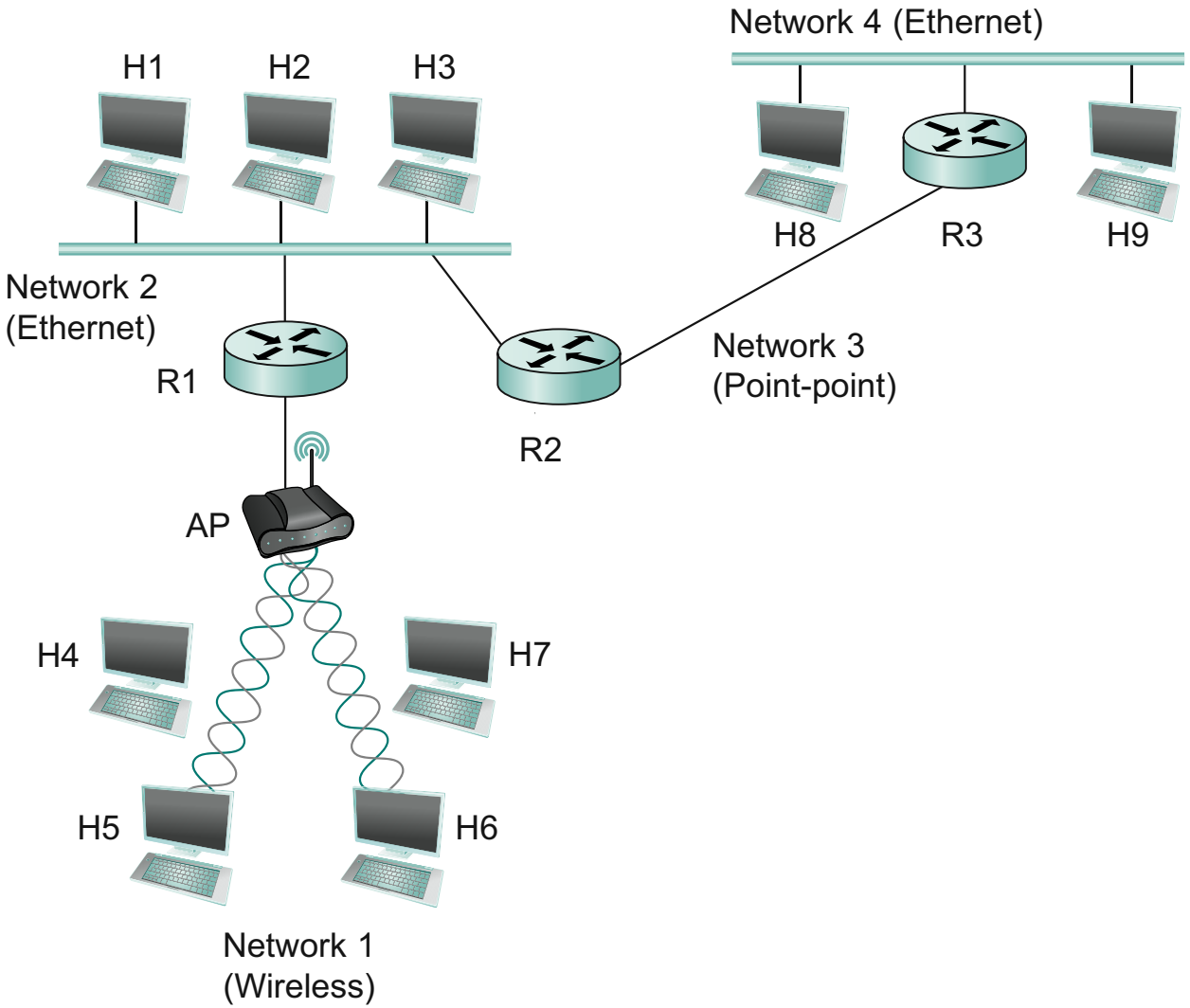
Adresarea ierarhica <adresa_retea, adresa_gazda> reduce numarul de intrari in tabela de dirijare (o intrare pentru o adresa_retea)

In practica, tabelele de rutare sunt separate pe clase de adrese
- Căutare prin: indexare (A şi B) sau hashing (C)

Exemplu

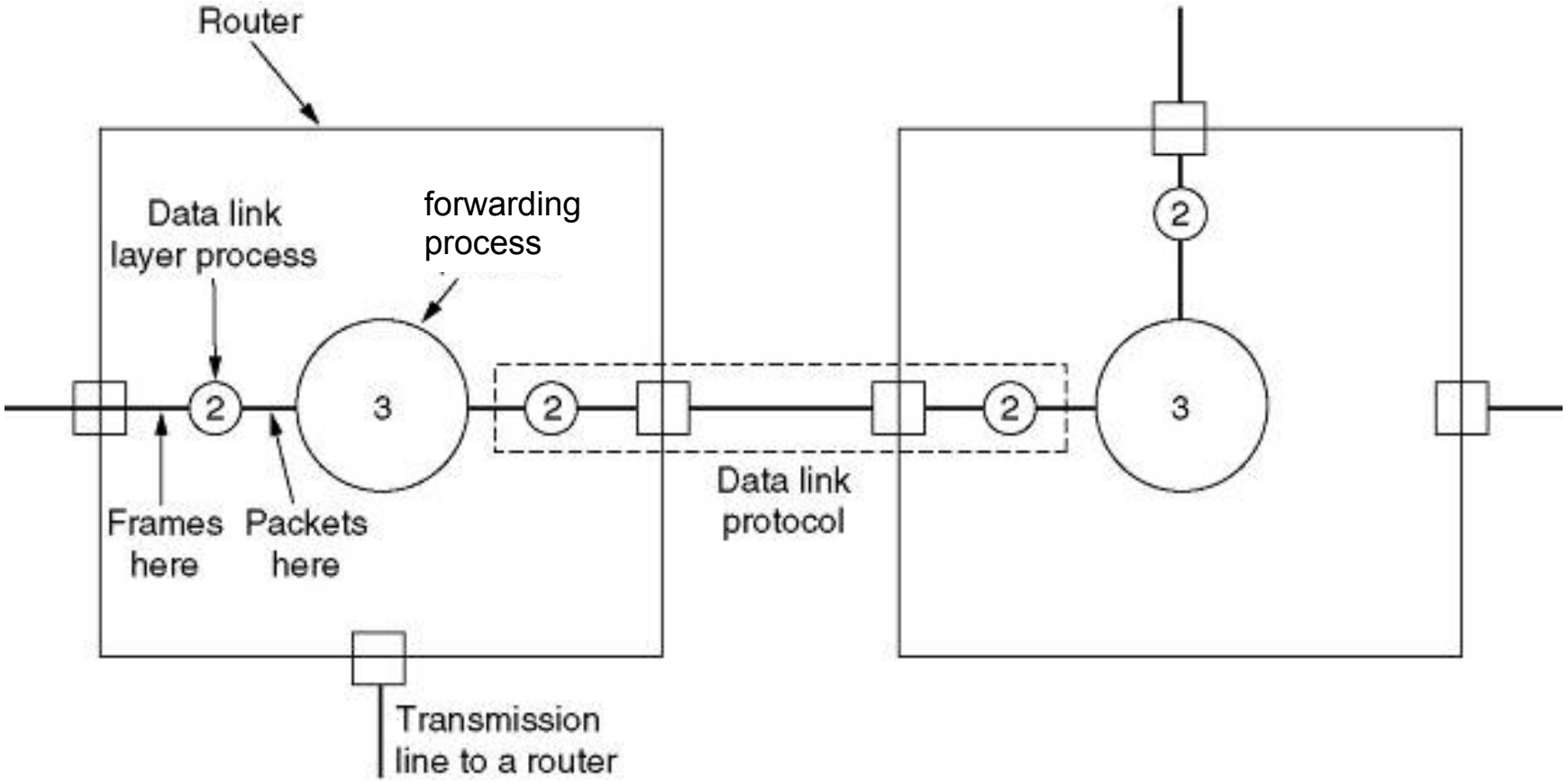
Transferul unui pachet intre H5 si H8 trece prin ruterele R1, R2 si R3

R3 livrează pachetul direct lui H8



Modelul unui ruter

Procesul de forwarding plaseaza pachetul într-o coadă asociată legăturii pe care trebuie să-l transmită



Conversie adresa IP – adresa fizică

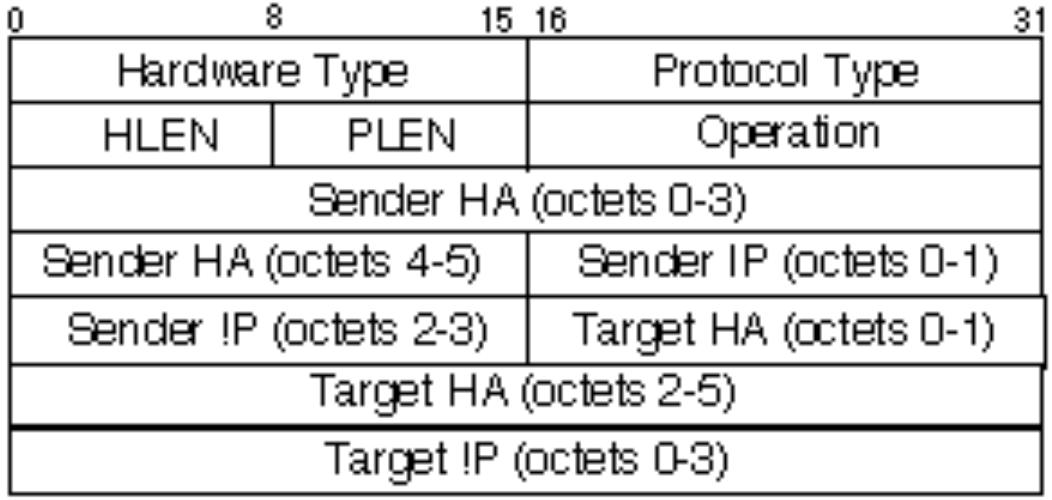
Tehnici

- tabele de corespondenţă
- formule de calcul
- schimb de mesaje

ARP - Address Resolution Protocol

Face maparea intre adresa de protocol şi adresa hardware

In figură:
livrare mesaj ARP



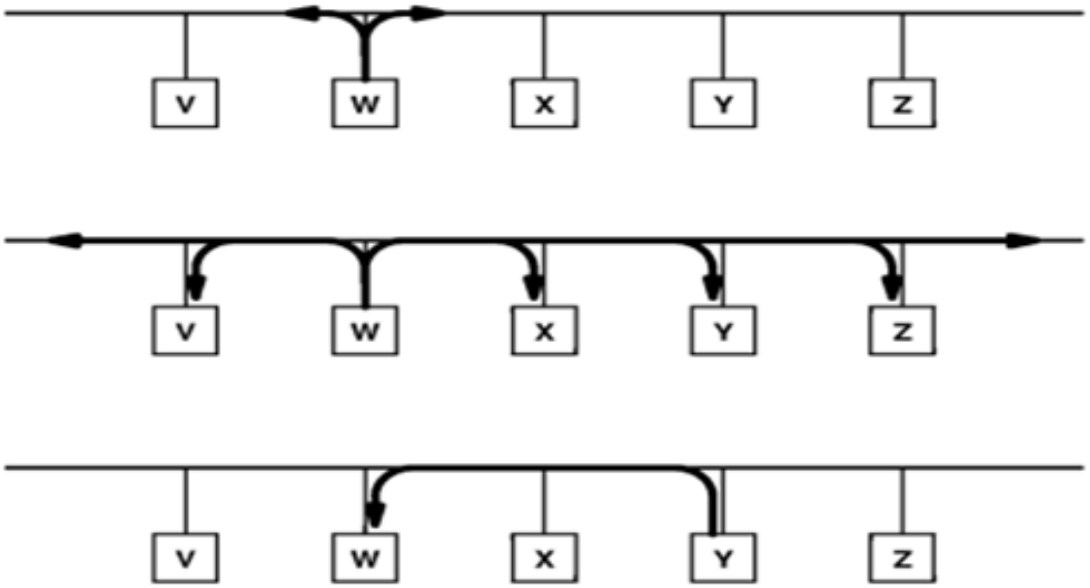


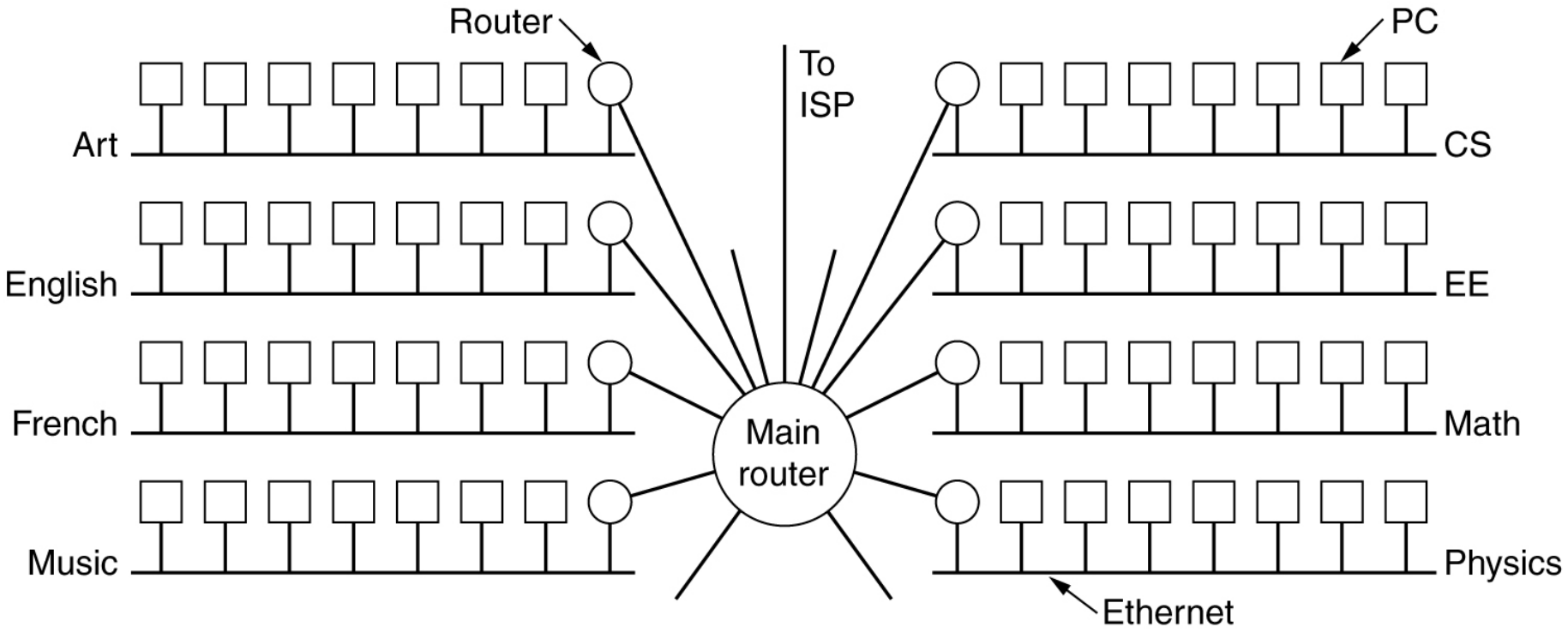
Tabela de *dirijare / forwarding*

- Denumirile se folosesc interschimbabil
- Utilizarile difera
  - tabela de dirijare este folosita de algoritmii de dirijare
  - cealalta este folosita de algoritmul de forwarding
- Uneori cele doua tabele au implementari separate
- Pentru eficienta, o intrare in tabela de forwarding trebuie sa indice
  - interfata pe care se trimite pachetul
  - adresa fizica a destinatarului
    - pentru retele locale Ethernet adresa MAC (48 biti) obtinuta cu ARP

Subreţele

- Regula: o adresa pentru fiecare retea fizica separata
  - consum mare de adrese
    - o retea clasa C cu doua noduri consuma inutil 253 de adrese de nod
    - o retea clasa B cu peste 255 noduri ocupa peste 64000 de adrese indiferent daca le foloseste pe toate
- solutia – subretele
  - de ex. o retea clasa B este impartita in mai multe subretele apropiate geografic
  - Organizarea este invizibilă în afara reţelei
    - Toate subretelele sunt vazute ca o singura retea cu o singura adresa
- adrese fara clase

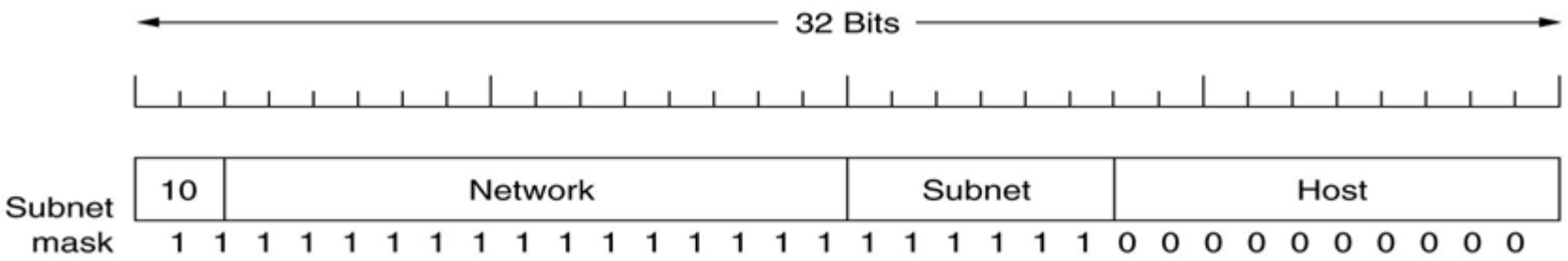
Subreţele - organizare (optional)



Ruterul principal dirijază pachetele spre ruterele de subreţea (cum?)

Ruterele de subreţea le livrează gazdelor

Adresarea în Subreţele (optional)



Exemplu: O reţea de clasă B impărţită în 64 subreţele

adresa Host împărţită în două: subreţea + Host

toate nodurile din subrețeaua fizică au aceeasi adresa retea+subretea

Pentru identificarea subretelei se foloseste o mască

pentru subreteaua de 64 adrese din fig. masca este 255.255.252.0

adresa_subretea = adresa_IP AND masca

Forwarding în Subreţele (optional)

Tabela de rutare pentru rutere de subretea are intrări de forma
(adresa_subretea, Masca, NextHop)

Algoritm rutare care suportă subrețele:

D = adresa_IP_destinatie
if exista intrare cu adresa_subretea = (D AND Masca)
if NextHop este o interfață (NextHop in aceeași subrețea)
transmite datagrama direct destinației
else transmite datagrama la ruter NextHop
else transmite datagrama la ruter implicit

CIDR – Classless InterDomain Routing

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

Notaţia specială pentru adresa de rețea CIDR

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

adresa reţea ocupă 21 biţi

adresa gazdă ocupă 11 biti

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

CIDR – Exemplu

Cambridge:	Adresă	11000010	00011000	00000000	00000000
	Mască	11111111	11111111	11111000	00000000
Edinburgh:	Adresă	11000010	00011000	00001000	00000000
	Mască	11111111	11111111	11111100	00000000
Oxford:	Adresă	11000010	00011000	00010000	00000000
	Mască	11111111	11111111	11110000	00000000

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

CIDR – reguli de alocare a adreselor

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

Reguli: o mască pentru un bloc de adrese

→ lungimea blocului trebuie sa fie o putere a lui 2

→ adresa de inceput a blocului de adrese alocat trebuie sa fie multiplu de dimensiunea acestuia

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

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

Cambridge	Edin.		Oxford

Algoritm *forwarding*

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

Algoritmul alege intrarea pentru care

(Adresa_IP AND Masca) = adresa_retea

Ex. Sosește pachet cu adresa_IP = 194.24.17.4

Cambridge /21 – adresa_rețea = 194.24.0.0

(Adresa_IP AND Masca) = 194.24.16.0 ➔ nepotrivire

Edinburgh /22 - adresa_rețea = 194.24.8.0

(Adresa_IP AND Masca) = 194.24.16.0 ➔ nepotrivire

Oxford /20 - adresa_rețea = 194.24.16.0

(Adresa_IP AND Masca) = 194.24.16.0 ➔ **potrivire**

Dacă nu sunt alte potriviri -> folosește intrarea pentru Oxford

Potriviri multiple

Prefixe de lungimi diferite

→ unele adrese IP se pot potrivi cu mai multe adrese_retea din tabela de dirijare

Ex.

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

Regula: se alege potrivirea “mai lungă”

Reducere dimensiune tabelă rutare

Soluție - agregarea intrărilor care au aceeaşi linie de ieşire

Consideram trei intrari in tabela, pentru retelele:

C - Cambridge: 194.24.0.0/21

E - Edingurgh: 194.24.8.0/21

O - Oxford: 194.24.16.0/20

Cambridge: Adresă 11000010 00011000 00000000 00000000

Edinburgh: Adresă 11000010 00011000 00001000 00000000

Oxford: Adresă 11000010 00011000 00010000 00000000

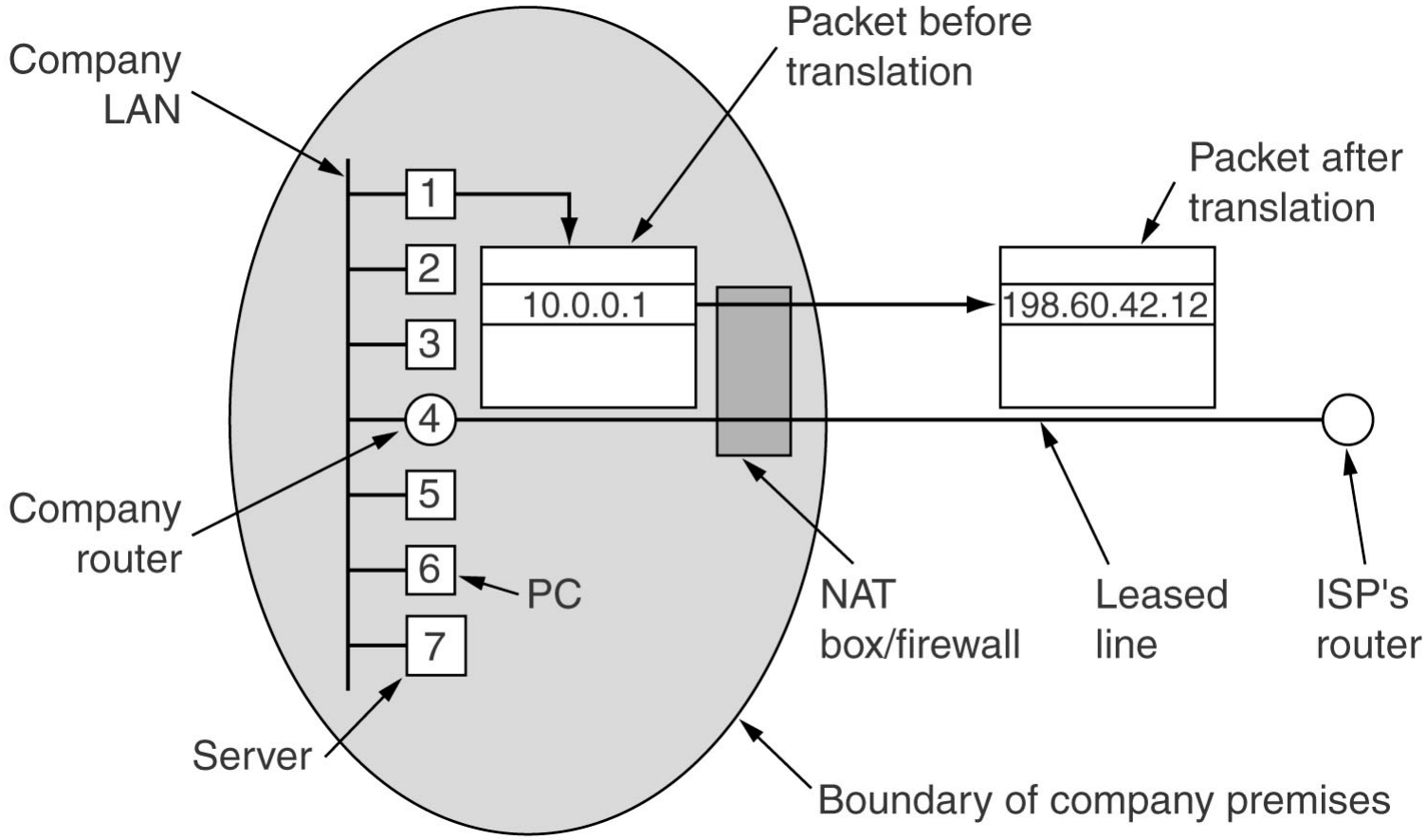
Presupunem pentru C, E, O, tabela de rutare are același NextHop

Inlocuiește 3 intrări cu una singură, având un prefix comun

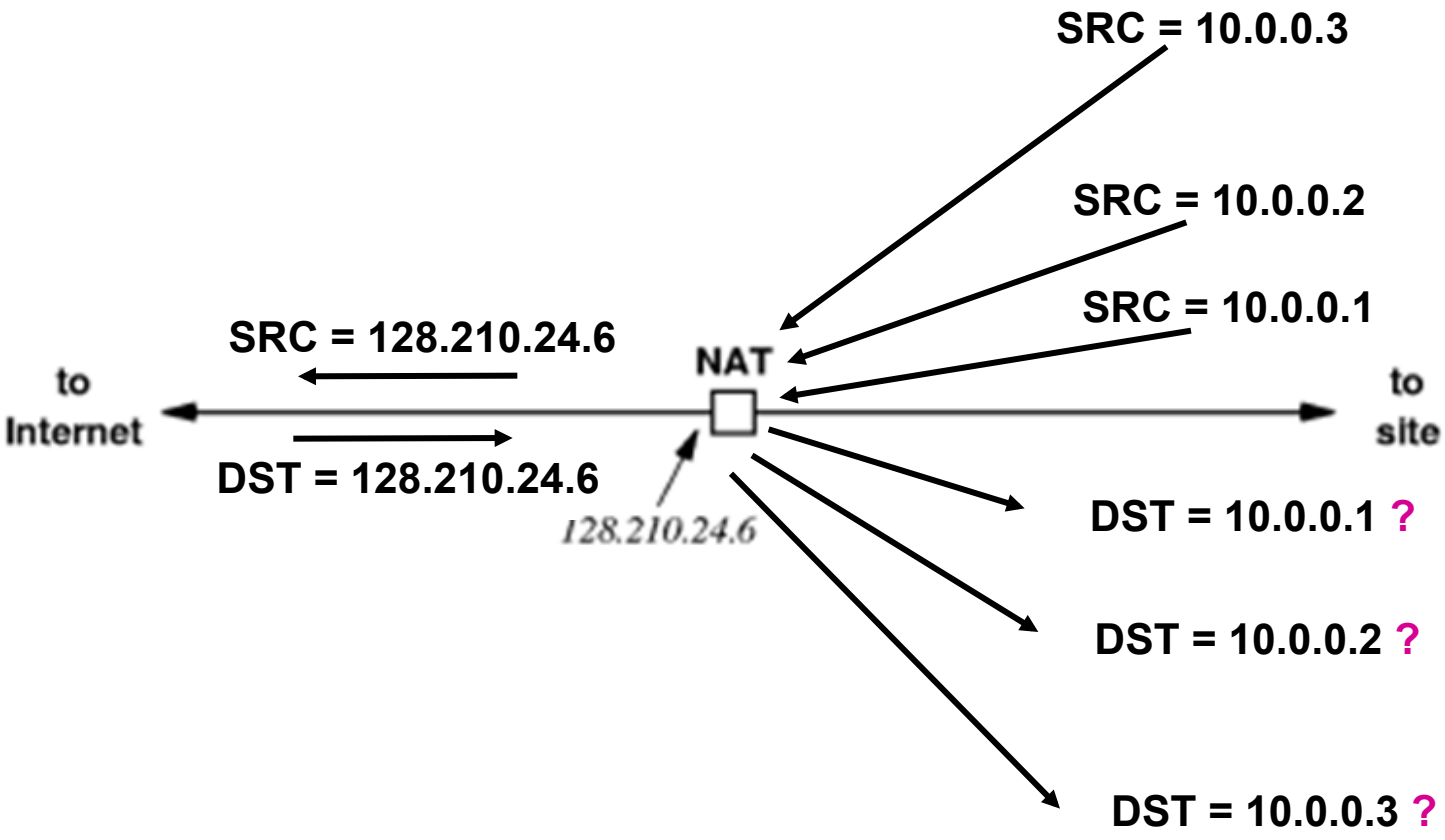
Corespunde cu 194.24.0.0/19

NAT – Network Address Translation

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



Translatarea adresa globală → adresa privată



Principiul NAT

Foloseşte

adresa IP + număr port transmitator

tabela de translatare

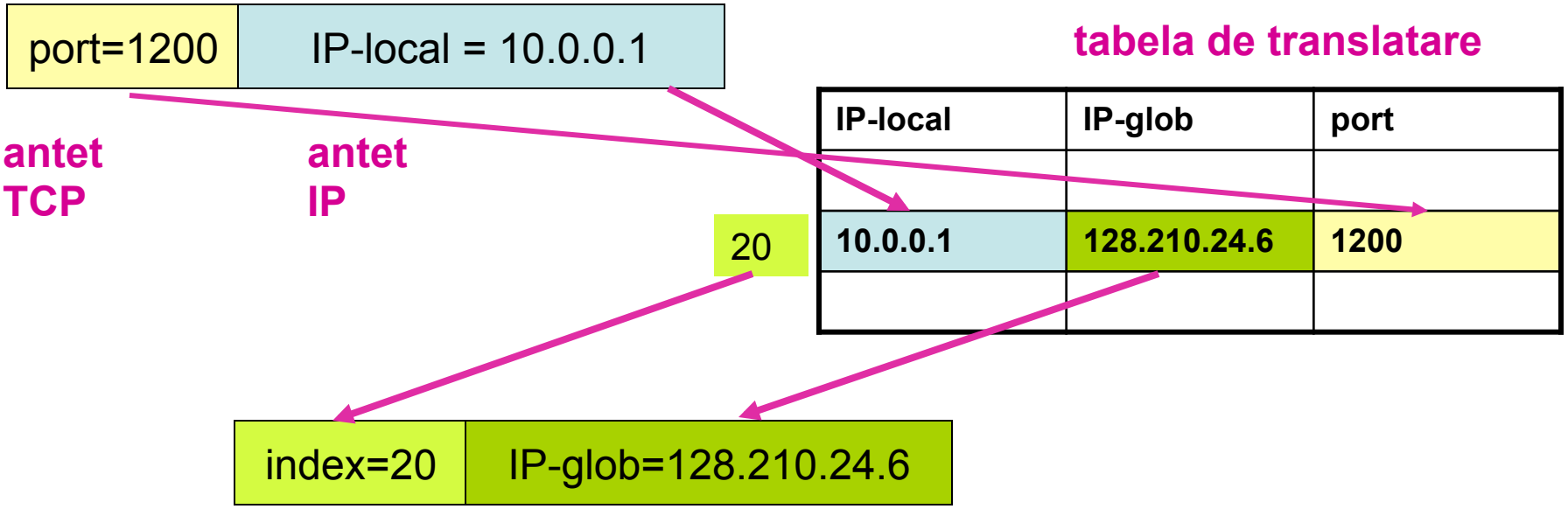
Transmisie

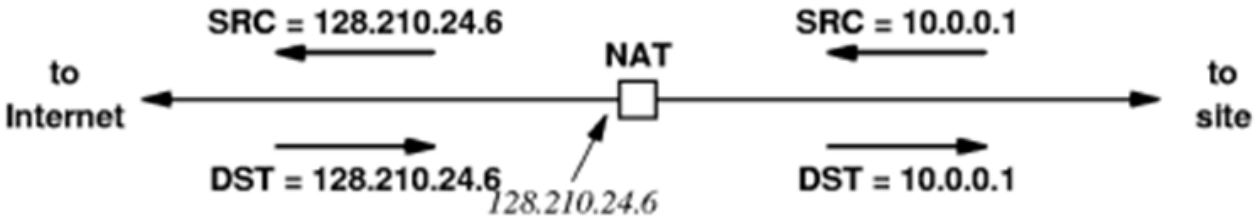
înlocuieşte adresa IP locală cu o adresă IP globală

memorează (in tabela de translatare) corespondenţa şi număr port

inlocuieşte număr port cu index în tabela translatare

re-compune sumele de control IP şi TCP





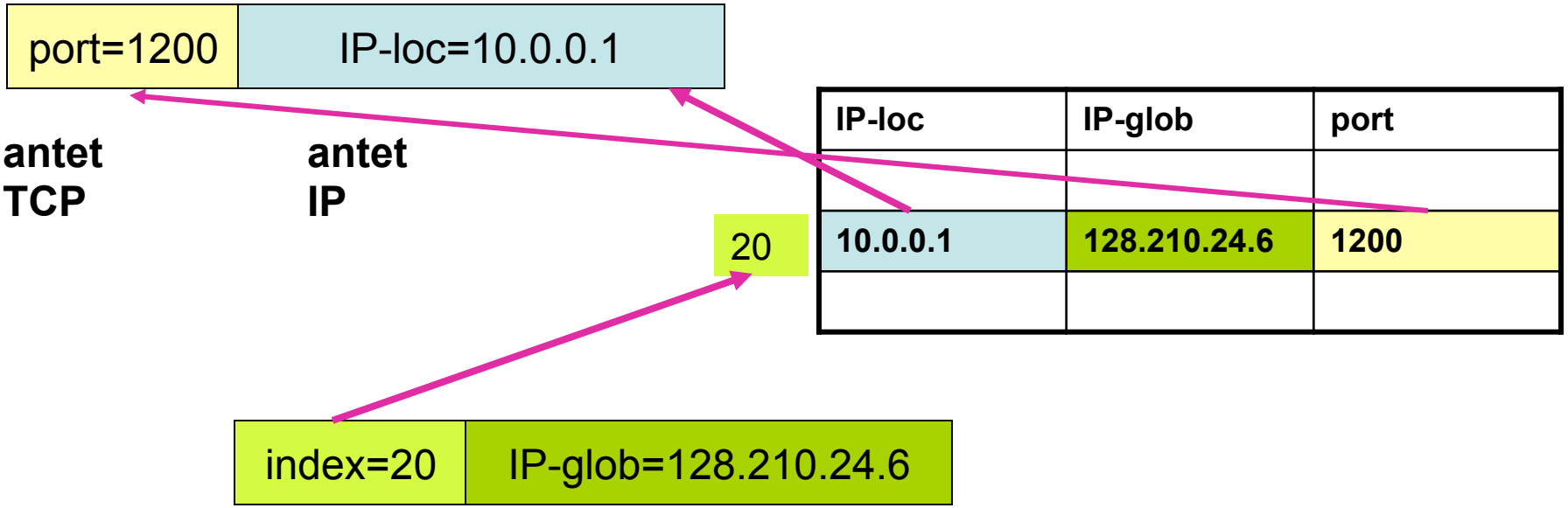
Receptie

obţine număr port din pachet (index în tabela translatare)

extrage adresa IP locală şi număr port

înlocuieşte adresa IP şi număr port din pachet

re-calculează sumele de control IP şi TCP



IP-loc	IP-glob	port
10.0.0.1	128.210.24.6	1200

ICMP- Internet Control Message Protocol

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

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

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

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

Folosire ICMP pentru aflare path MTU

Path MTU = Maximum Transmission Unit minimă pentru o cale

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

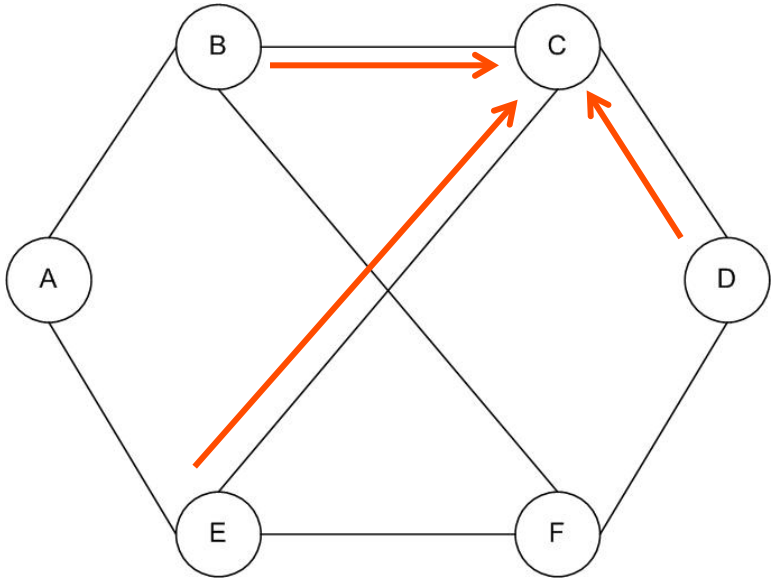
Dirijarea - clasificare

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

Dirijare în Internet

- Internet = număr mare de **Autonomous Systems**
- Două tipuri de protocoale de dirijare
  - IGP – Interior Gateway Protocols (în AS)
    - **RIP – Routing Information Protocol**
      - **Distance vector**
    - **OSPF – Open Shortest Path First**
      - **Link state**
  - EGP – Exterior Gateway Protocols (intre ASs)
    - **BGP – Border Gateway Protocol**

RIP - Dirijare folosind vectorul distanţelor



Algoritm distribuit !

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

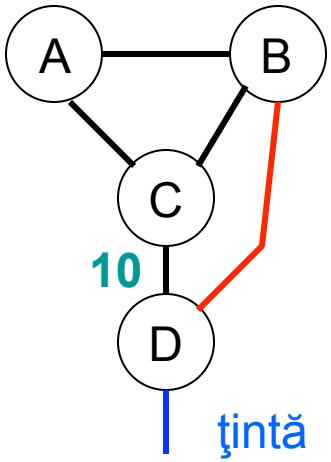
Următorii vectori au fost primiţi de nodul C (lista include distanţele la nodurile A, B, C, D, E, F, în această ordine):

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

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

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

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



Problema numărării la infinit

Toate legăturile au cost 1, exceptând (C,D) cu cost 10

Costurile **la ţintă** sunt:

D: direct conectată, cost 1
B: ruta prin D, cost 2
C: ruta prin B, cost 3
A: ruta prin B, cost 3

Legătura (B,D) cade.

timp -->

D:	dir, 1	dir, 1	dir, 1	dir, 1	...	dir, 1	dir, 1
B:	unreach	C, 4	C, 5	C, 6		C, 11	C, 12
C:	B, 3	A, 4	A, 5	A, 6		A, 11	D, 11
A:	B, 3	C, 4	C, 5	C, 6		C, 11	C, 12

C alege ruta prin A şi A alege ruta prin C.

B alege ruta prin C

In ultimul pas, C găseşte o cale mai ieftină prin D şi problema se rezolvă.

Pentru reţele deconectate, numărarea continuă la infinit.

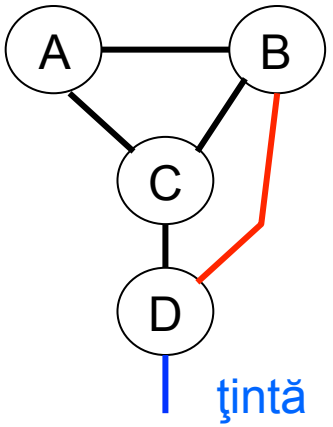
Soluţii

Adoptate în RIP - Routing Information Protocol

"**simple split horizon**" omite rutele învăţate de la un vecin în actualizările timise acestuia

"**split horizon with poisoned reverse**" include astfel de rute dar pune un cost infinit.

Ideea: în mesajul său către C, A trebuie să informeze că D nu mai este tangibil



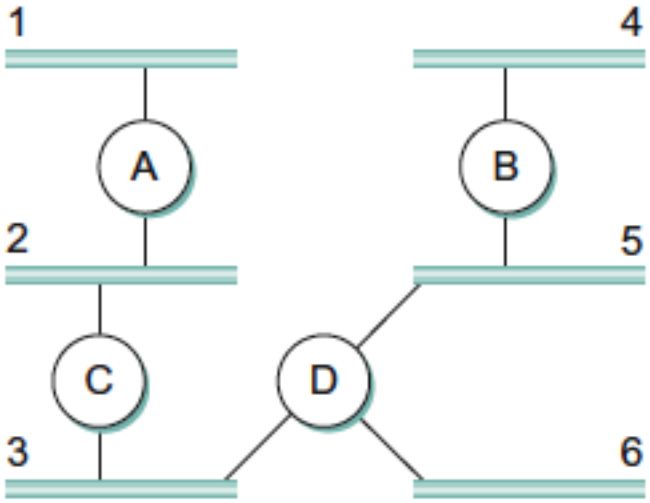
D:	dir, 1	dir, 1	dir, 1
B:	unreach	unreach	C, 12
C:	B, 3	D, 11	D, 11
A:	B, 3	unreach	C, 12

Protocoale care folosesc split horizon

- RIP - Routing Information Protocol
- IGRP - Interior Gateway Routing Protocol
- EIGRP – Enhanced Interior Gateway Routing Protocol

RIP - Routing Information Protocol

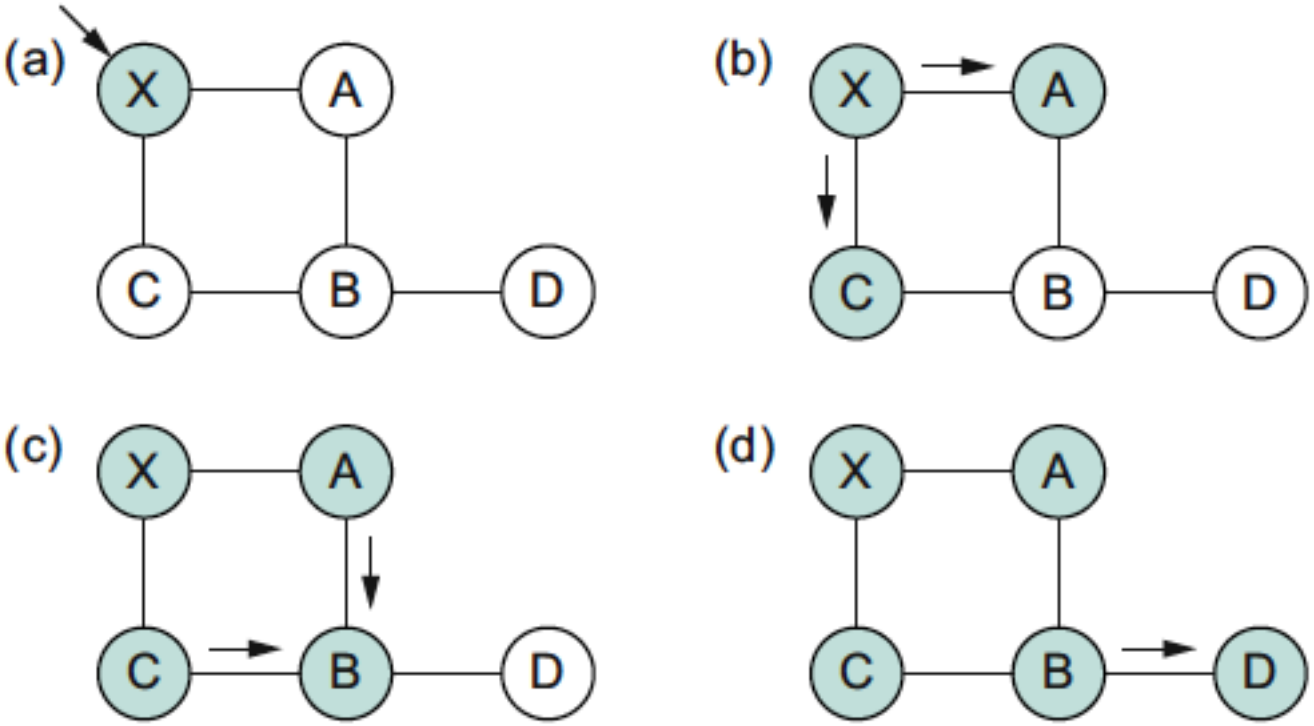
- foloseste distante la retele (nu la noduri)
  - ruterul C are distanta 0 la reteaua 2 si 2 la reteaua 4
- transmit vectorii distantelor la fiecare 30 secunde
- distante maxime de 15 hop-uri (16 inseamna infinit)
- fiecare legatura are cost 1



Starea legaturii

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

Transmiterea prin inundare



Algoritmi de dirijare - Calea cea mai scurtă

Algoritmul lui Dijkstra

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

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

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

Dirijare distribuită bazată pe starea legăturilor

C tabloul distanţelor;

C[d][v] este lungimea (sau costul) drumului de la nodul curent la nodul destinatar d, *prin nodul vecin* v;

D tabloul distanţelor minime;

D[d] este lungimea drumului minim de la nodul curent la nodul destinatar d;

V tabloul de dirijare;

V[d] este nodul vecin prin care se transmit datele, pe drumul minim, spre destinatarul d.

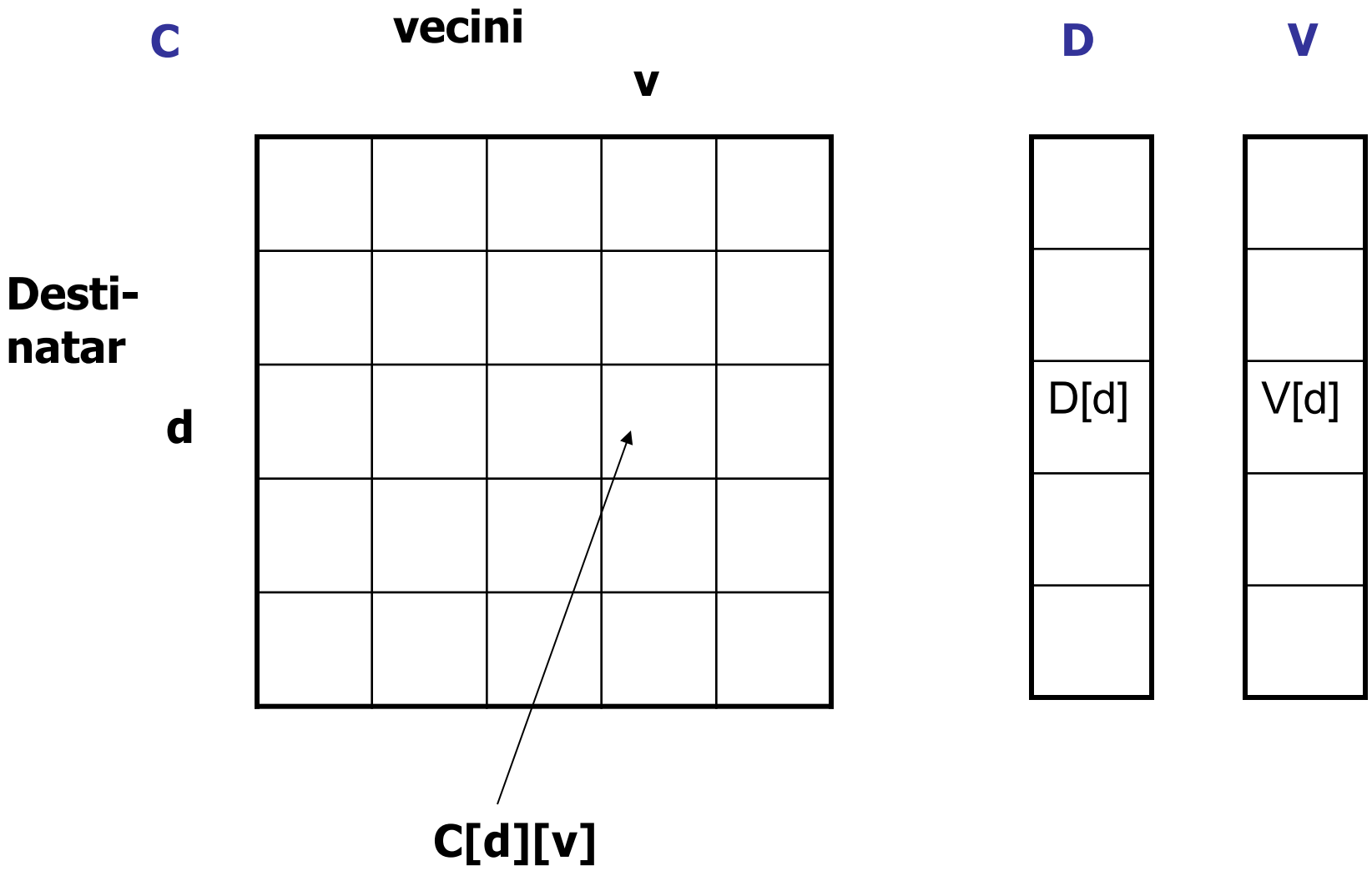
Evenimente tratate:

adăugarea unei noi legături;

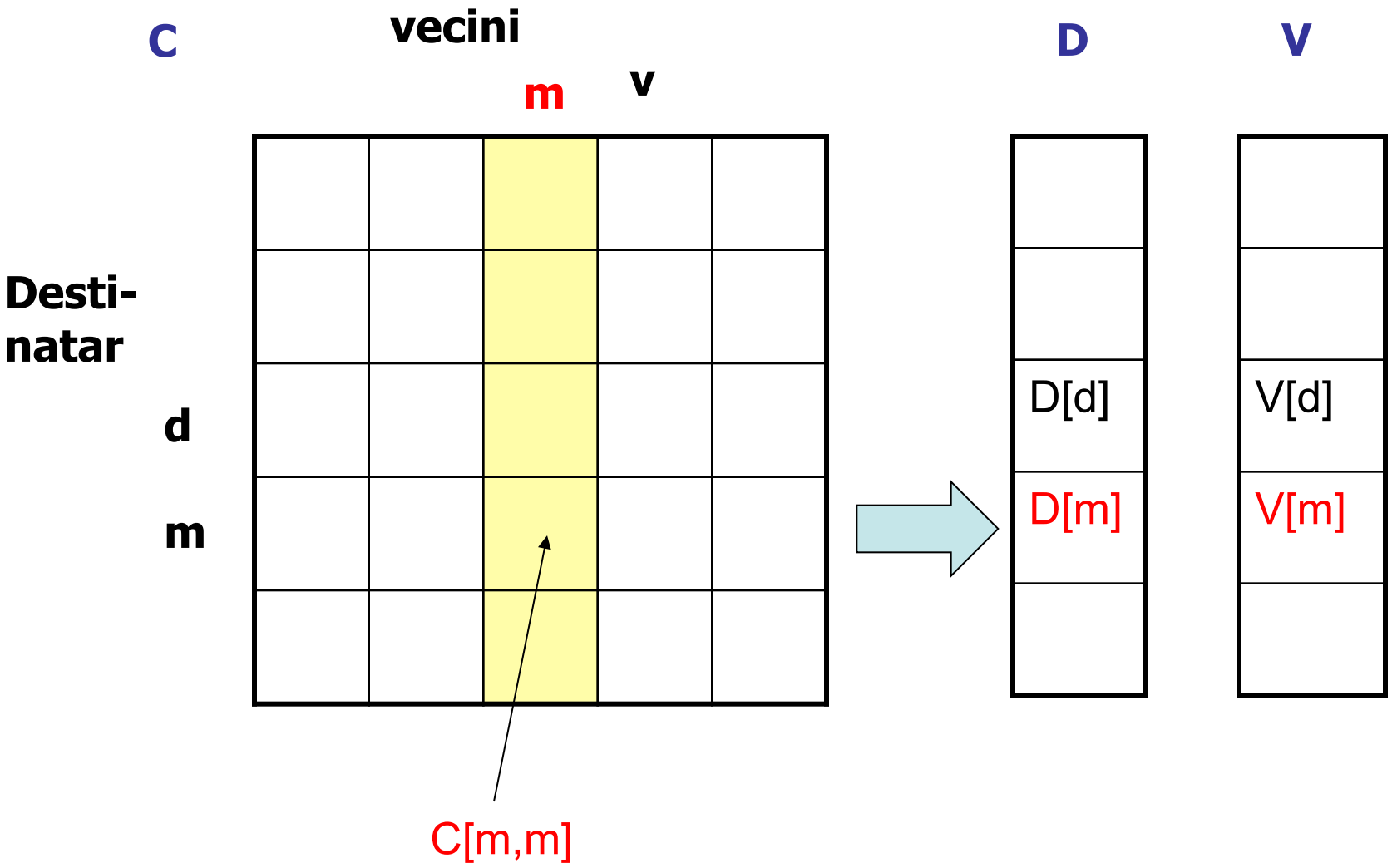
sesizarea modificării lungimii unei linii;

primirea unui mesaj de control de la un nod vecin.

Structuri de date pentru nodul crt



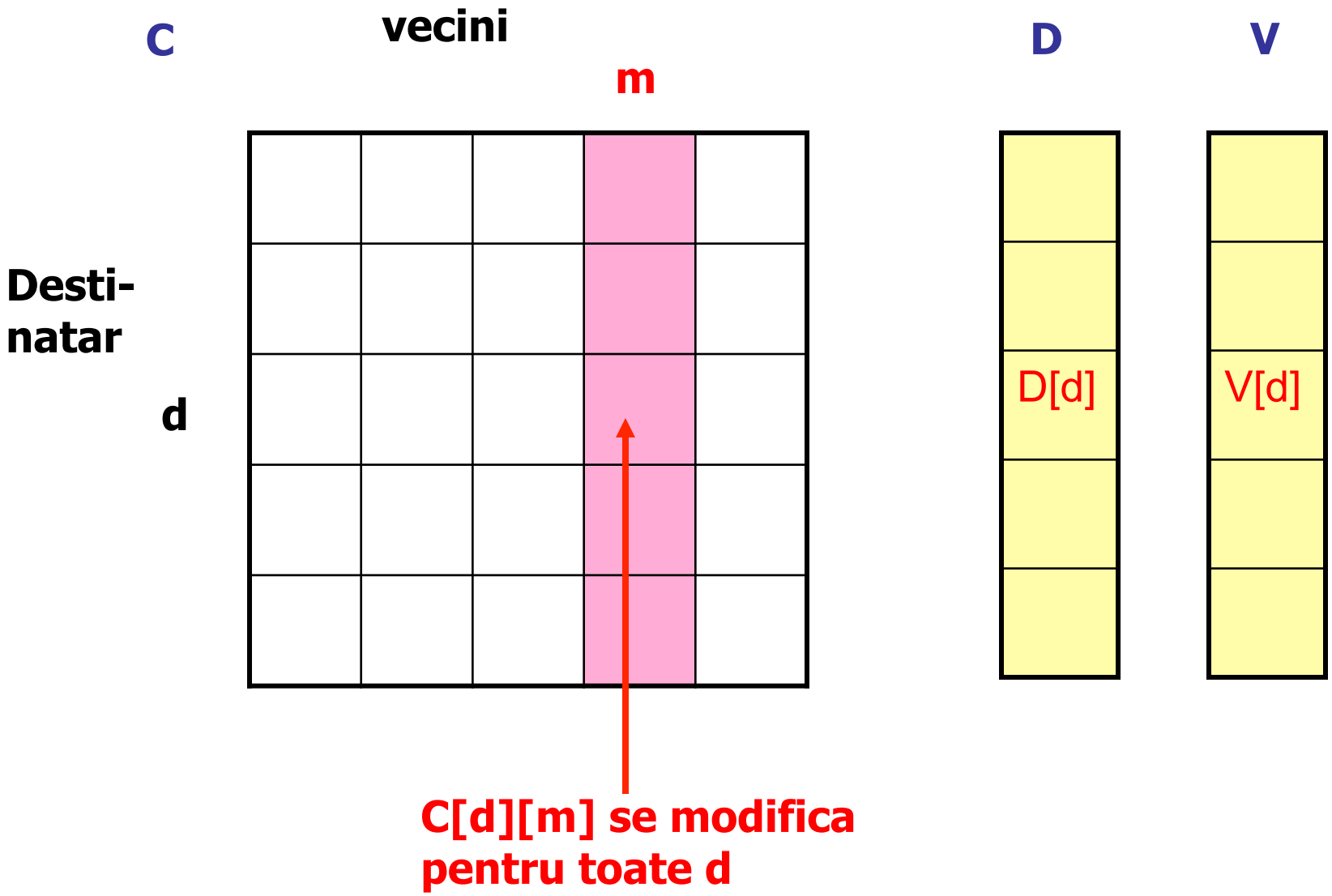
Adaugă legătura crt-m



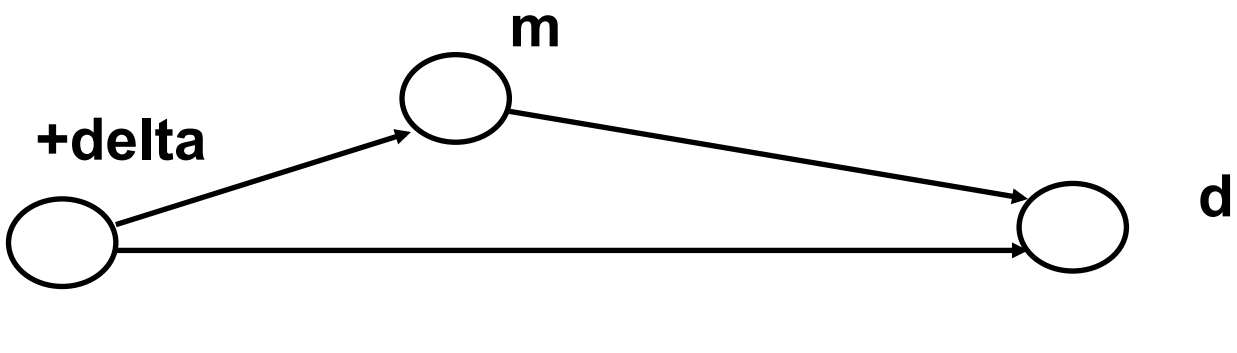
```
/* adauga legatura (crt,m), crt = nodul curent*/

void adauga_legatura (int m)
{
   C[m][m] = l[crt][m];
   calculeaza p ptr care C[m][p]=min C[m][w], dupa w;
   V[m]=p;
   if (C[m][p] != D[m])
     {D[m] = C[m][p];
      transmite mesaj (crt,m,D[m]) tuturor vecinilor;
     }
   transmite mesajele (crt,a,D[a]),...,(crt,z,D[z]) nodului m;
}
```

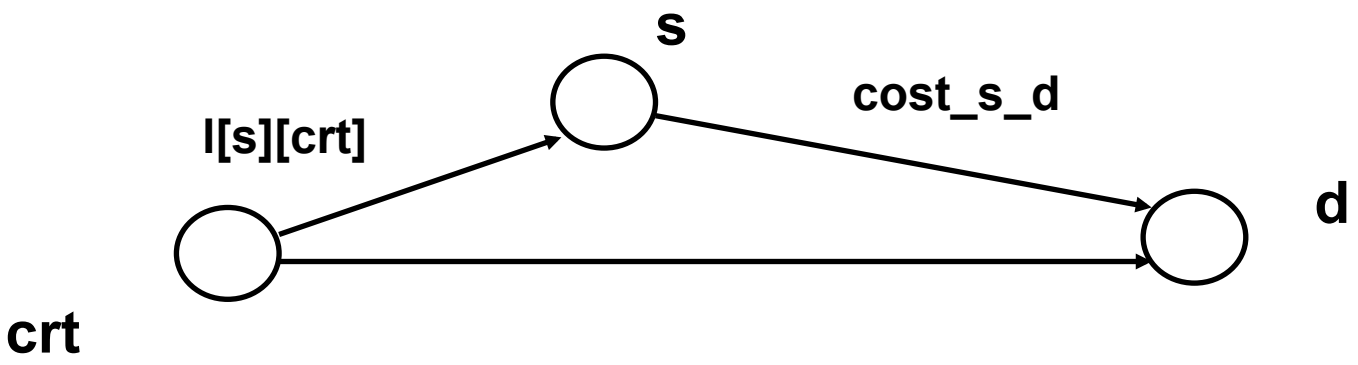
Schimbă cost crt-m cu delta_crt_m



```
void schimba_cost (int m, int delta_crt_m)
{
  for (toate destinatiile d)
   { C[d][m] += delta_crt_m;
     calculeaza p a.i. C[d][p]=min C[d][w], dupa w;
     V[d] = p;
     if (C[d][p] != D[d])
        { D[d] = C[d][p];
          transmite mesaj (crt,d,D[d]) tuturor vecinilor;
        }
   }
}
```



```
void receptie_mesaj (int s, int d, int cost_s_d)
{
   if (d != crt)
    { C[d][s] = cost_s_d + l[s][crt];
      calc p a.i. C[d][p] = min C[d][w], dupa w;
      V[d] = p;
      if (C[d][p] != D[d])
         {D[d] = C[d][p];
          transmite mesaj (crt,d,D[d]) tuturor vecinilor;
         }
     }
  }
```



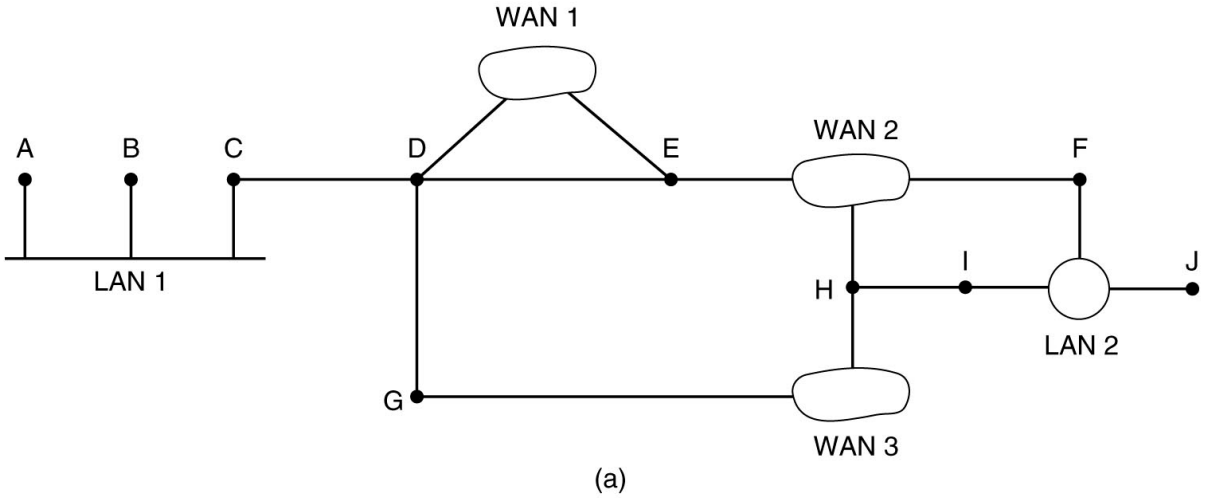
OSPF

Suportă:

Linii punct la punct intre două rutere

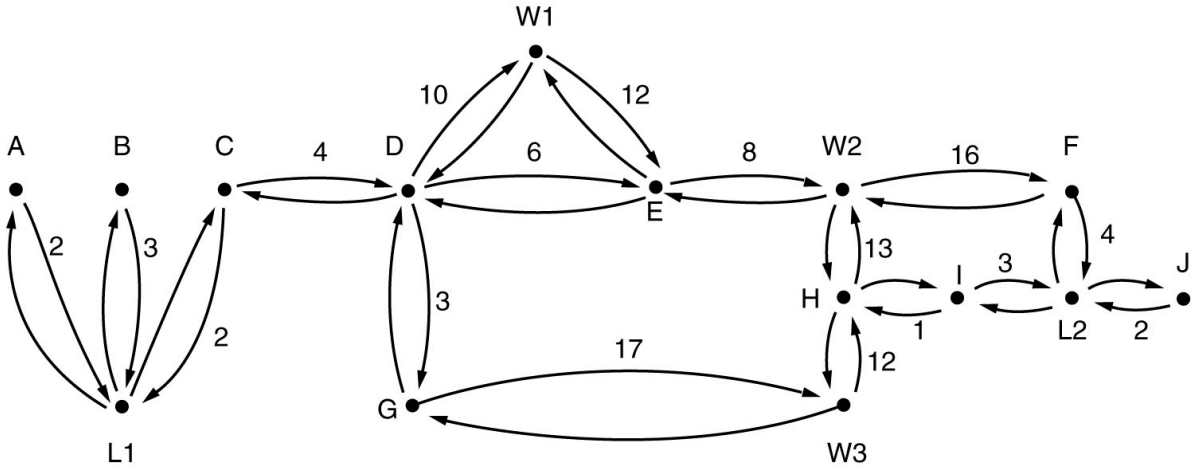
LANs

WANs



(a)

Modelul de graf



(b)

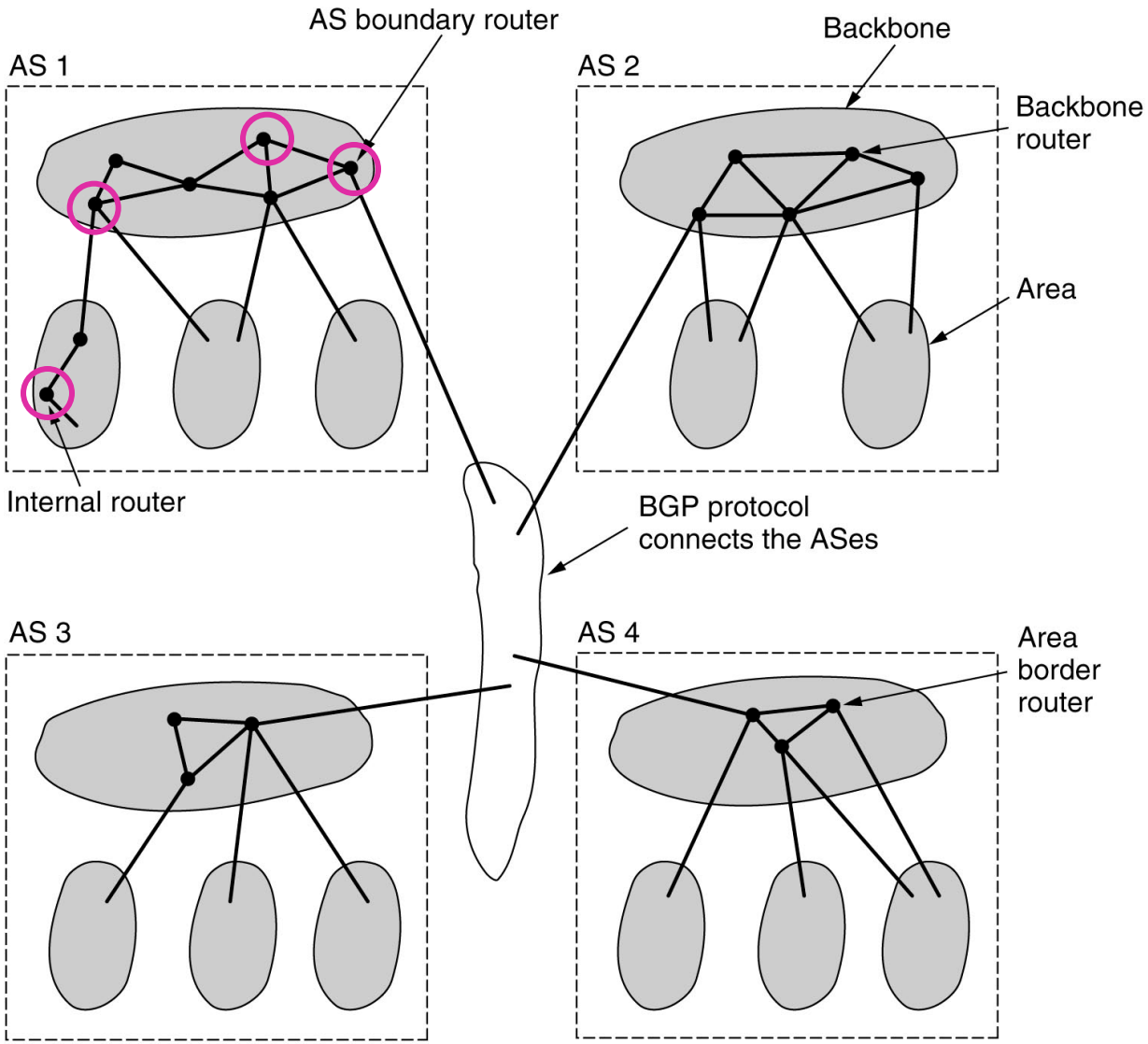
OSPF

Fiecare AS are mai multe zone (Areas)

Tipuri de rutere:

- **interne**
- **de coloană vertebrală**
- **de graniţă zonală** (conecteaza mai multe zone)
- **de graniţă AS**

OSPF foloseşte schimb inf între rutere adiacente



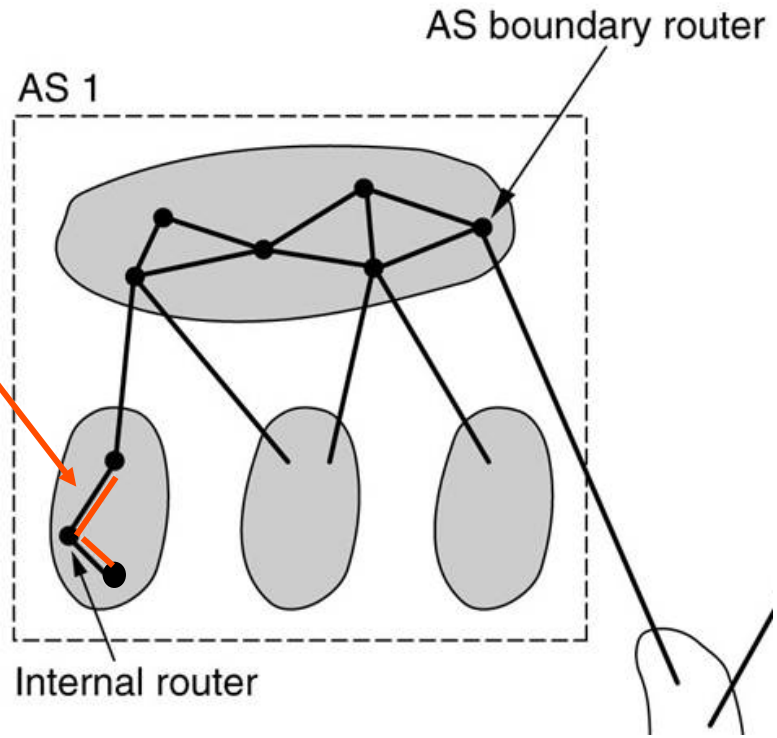
Calcul rute

Nivel 1 (zona)

Fiecare ruter din **zonă** calculează separat căile cele mai scurte catre ruterele din aceeasi zona (la fel fac ruterele din coloana vertebrala)

Mesaje OSPF

- **Hello - descoperă vecinii**
- **Actualizare stare legătură –** furnizează costul unei legături + nr secv (mai multe costuri intr-un pachet)
- **Confirmare stare legătură –** confirmă primirea
- **Descriere bază de date –** furnizează toate costurile (vecin nou)
- **Cerere stare legătură –** cere info de actualizare



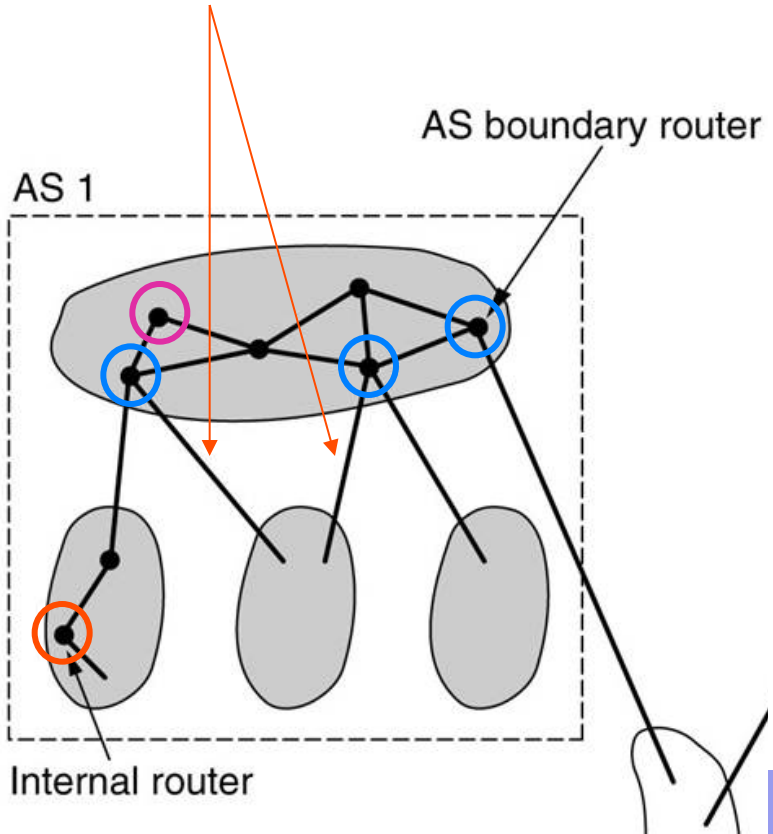
Nivel 2 (AS)

Ruterele de **coloana vertebrala** (backbone)

- **acceptă info de la ruterele de granita zonale**
- **calculeaza cele mai bune rute intre orice ruter backbone şi toate celelalte rutere**
- **propagă info înapoi la ruterele de granita zonale**

Ruterele de granita zonale avertizează ruterele din zonă

Fiecare ruter selectează cea mai bună ieşire spre backbone



BGP – Border Gateway Protocol

Algoritmi orientaţi pe aspectele politice, de securitate, economice

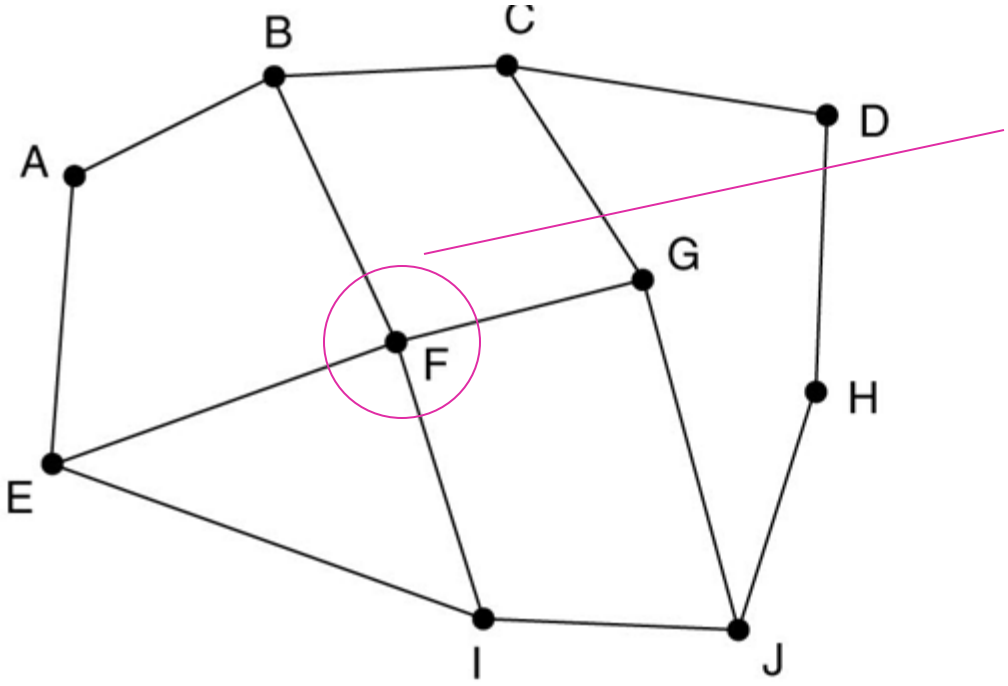
Reţea = ASes şi conexiunile

Protocol = vectorul distanţelor

Tabelele de dirijare conţin şi rutele spre destinaţie

Comunică vecinilor căile utilizate efectiv

pp. F foloseste calea FGCD la D



(a)

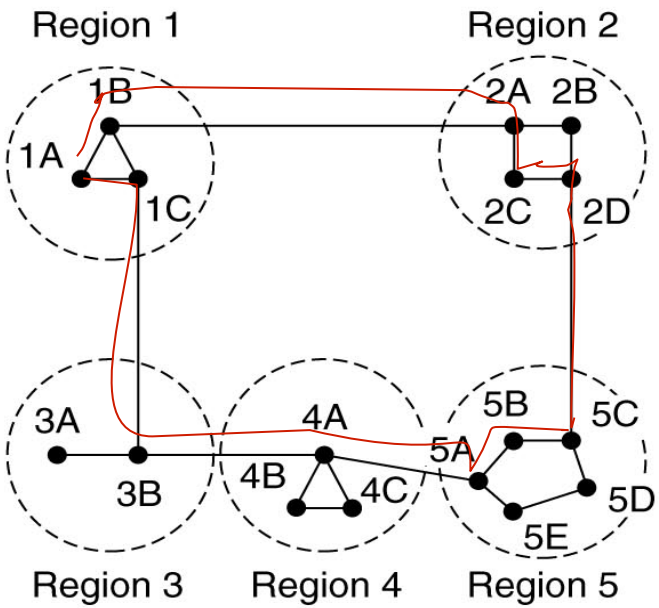
Information F receives from its neighbors about D

From B: "I use BCD"
From G: "I use GCD"
From I: "I use IFGCD"
From E: "I use EFGCD"

(b)

cand G cade, F alege calea FBCD

Dirijare ierarhică (optional)



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

1C mai bun ptr majoritatea dest din grupul 5

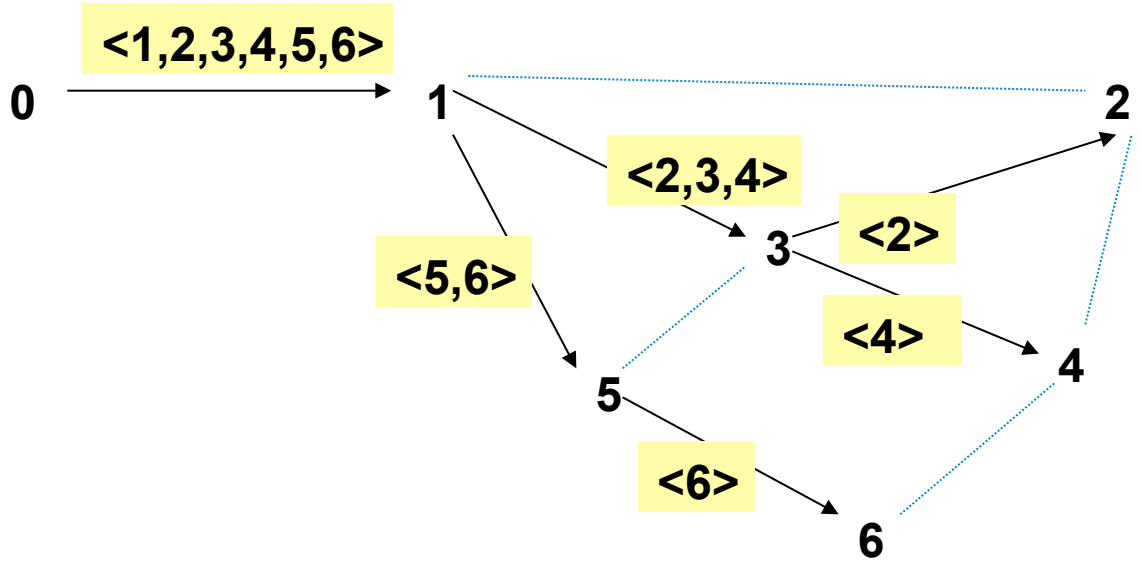
Reducere număr intrări
17 -> 7

Penalizare la dest. 5C

(c)

Difuzare şi multicast (optional)

- Punct la punct - Trimite un pachet fiecărei destinaţii
- Inundarea
  - **Generează prea multe pachete**
  - **Copiile sunt distruse**
- Dirijarea multidestinaţie
  - **Pachetul conţine lista adreselor de destinaţie**
- Arbore de acoperire



Difuzare – urmărirea căii inverse (optional)

(a) O subreţea **(b)** un arbore de acoperire pentru nodul I (caile preferate catre I)

calea preferată între nodurile I şi E este cea pe care E trimite pachete lui I

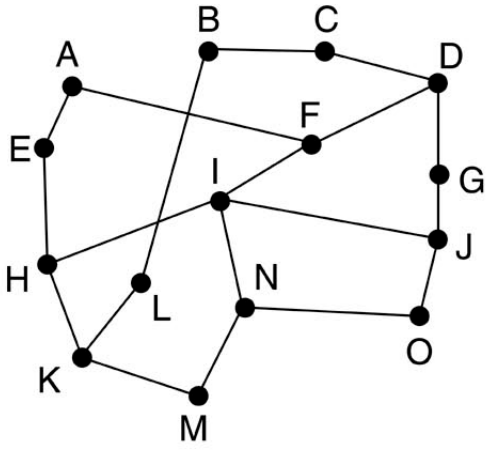
(c) functionarea algoritmului căilor inverse: când un pachet ajunge la un ruter:

verifică în tabela sa de dirijare dacă a sosit pe **calea preferată**

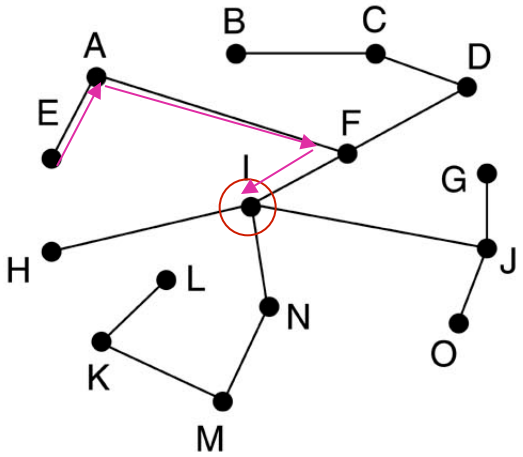
dacă da -> este trimis pe toate celelalte linii

altfel -> este distrus

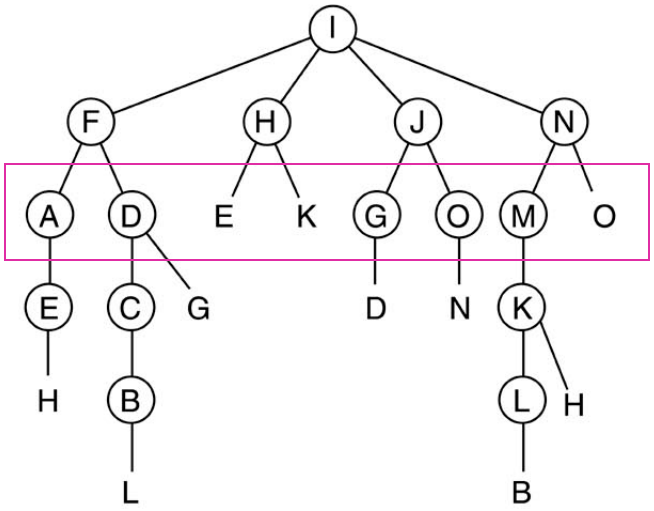
– la pasul 2 doar 5 din cele 8 pachete ajung pe calea preferata si sunt difuzate in continuare



(a)



(b)



(c)

Dirijarea în reţele ad hoc

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

reţea ad hoc = graf

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

Fiecare nod = ruter + gazdă

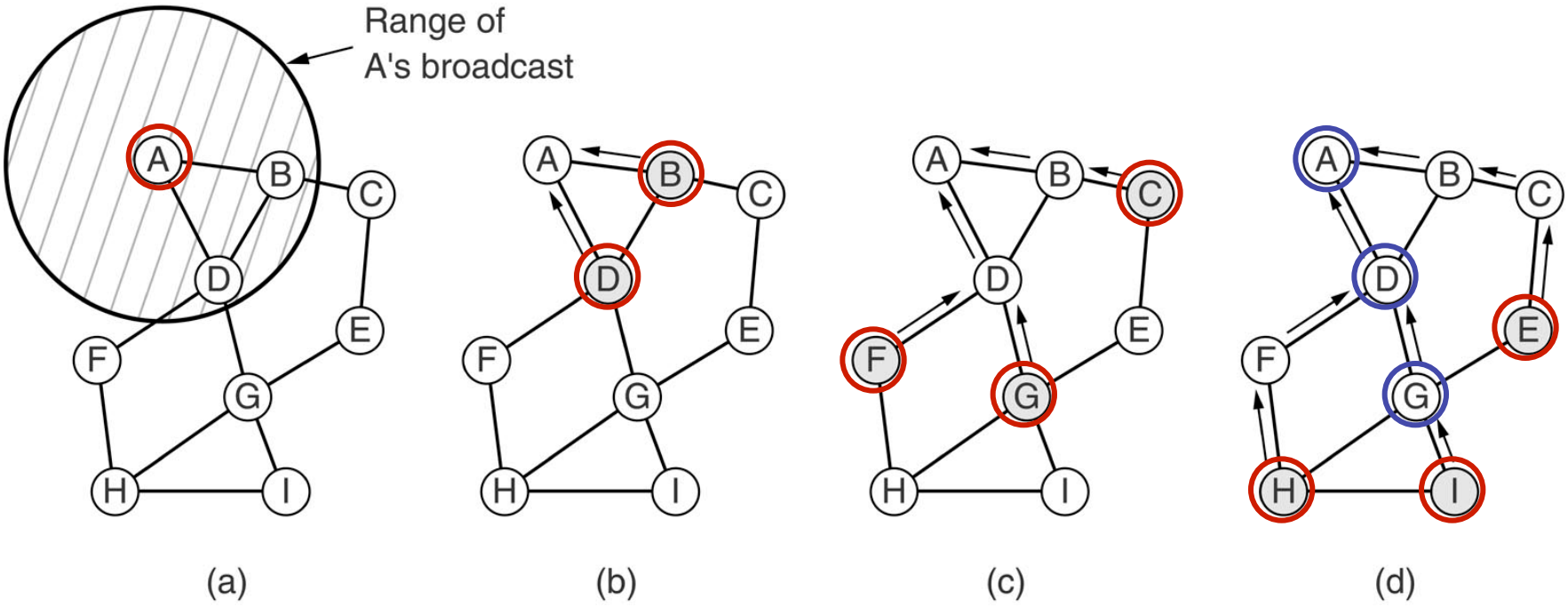
Conţine

- **Tabela dirijare**
  - **destinaţie,**
  - **pas următor,**
  - **distanţă,**
  - **nr secv destinaţie**
  - **altele**
- **Tabela history**
  - **identitatile cererilor precedente**
- **Tabela reverse route**
  - **calea spre sursa unui pachet de cerere**

Cum functioneaza?

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

-> trebuie să descopere ruta



Pachete ROUTE REQUEST

A difuzează un pachet ROUTE REQUEST

- Identificat unic prin Source address + Request ID
- Foloseşte Sequence # pentru a deosebi rutele noi de cele vechi

Prelucrarea ROUTE REQUEST în fiecare nod

- Verifica duplicat în tabela history locală (Source address + Request ID)
- Transmite ROUTE REPLY dacă găsit ruta nouă, adică
  - Dest sequence # în routing table > Dest sequence # în packet
- Altfel,
  - incrementează Hop count şi re-difuzează ROUTE REQUEST
  - memorează informaţia în reverse route table
- Source sequence # folosit pentru actualizare tabela dirijare locala

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

Pachete ROUTE REPLY

I construieşte ROUTE REPLY şi-l trimite pe legătura inversă

- **Source address**, **Destination address** **sunt copiate**
- **Hop count** **pus pe zero**
- **Destination sequence #** **luat din contorul propriu**
- **Lifetime** **= cât timp rămâne valid**

Prelucrarea la alte noduri

- **Actualizează tabela dirijare locală**
- **Transmite pe legătura inversă**
- **Trece prin anumite noduri – celelalte şterg intrarea în reverse route table**

Source address	Destination address	Destination sequence #	Hop count	Lifetime

Intreţinerea rutelor

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

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

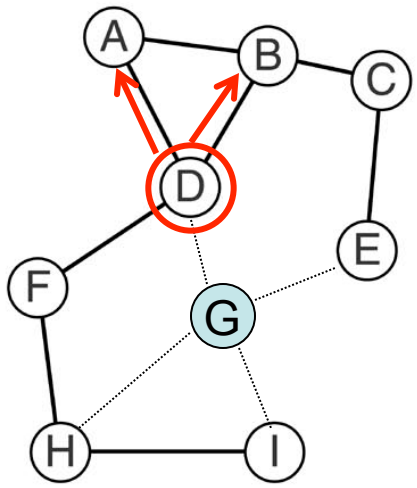
D află că G a fost utilizat pe rute către E, G şi I

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

D goleşte intrările pentru E, G şi I din tabela de rutare

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

(a)



(b)

IPv6

Motivaţii

Spaţiul de adrese

32 biţi = peste un milion de reţele

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

214 adrese de reţea Clasa B, multe folosite

Tip servicii

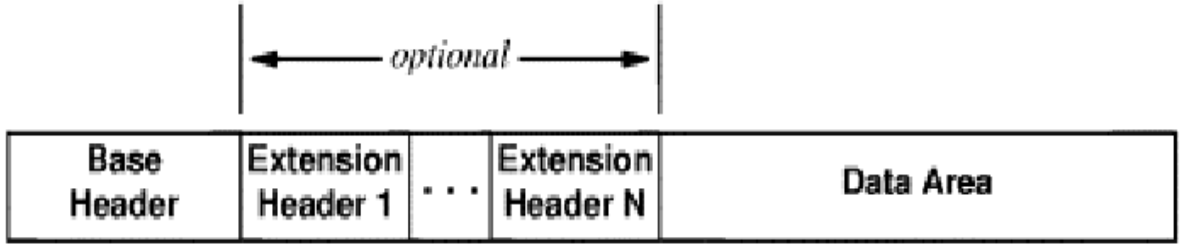
Aplicaţii diferite au cerinţe diferite de livrare, siguranţă şi viteză

IPv4 are tip de serviciu dar adesea nu este implementat

Caracterizare IPv6

- **format antet**
- **antete extensii**
- **suport audio şi video**
- **protocol extensibil**
- **spaţiu adresa**
- **multicast**

IPv6 - format datagrama



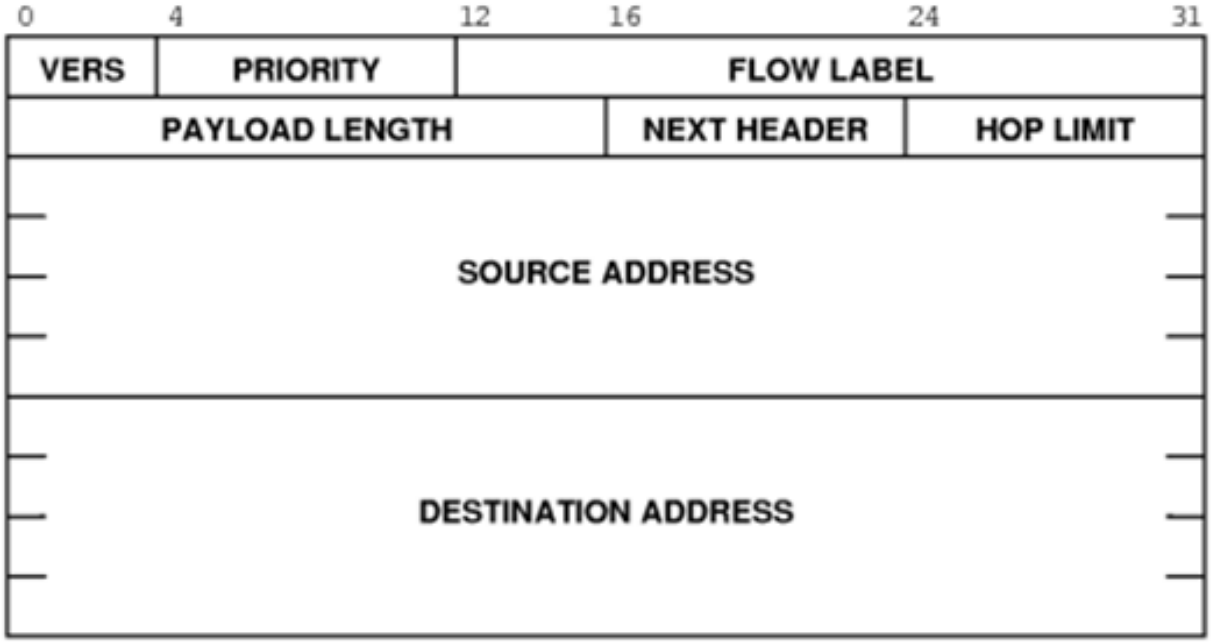
IPv6 format Base header

Base header
lungime fixă = 40 octeti

Prioritate - clasa de trafic

FLOW LABEL - asociază datagramele unui flux

Diferenţe circuit virtual
două fluxuri cu aceeaşi etichetă se dif prin adr sursă + adr dest
aceeaşi pereche sursă+dest poate avea mai multe fluxuri



Conţine mai puţine info decât antet IPv4

Restul de info în extensii

NEXT HEADER defineşte tipul datelor (ex. TCP)

NEXT HEADER defineşte tipul antetului de extensie (ex. route header)

(a)

(b)

IPv6 – antete extensie

Hop-by-hop header – info pentru rutere

- suport datagrame excedând 64K (jumbograme)
- specifica lungimea;
- campul de lungime din antetul de baza este 0

Destination header – info aditionale pentru destinaţie

- nefolosit

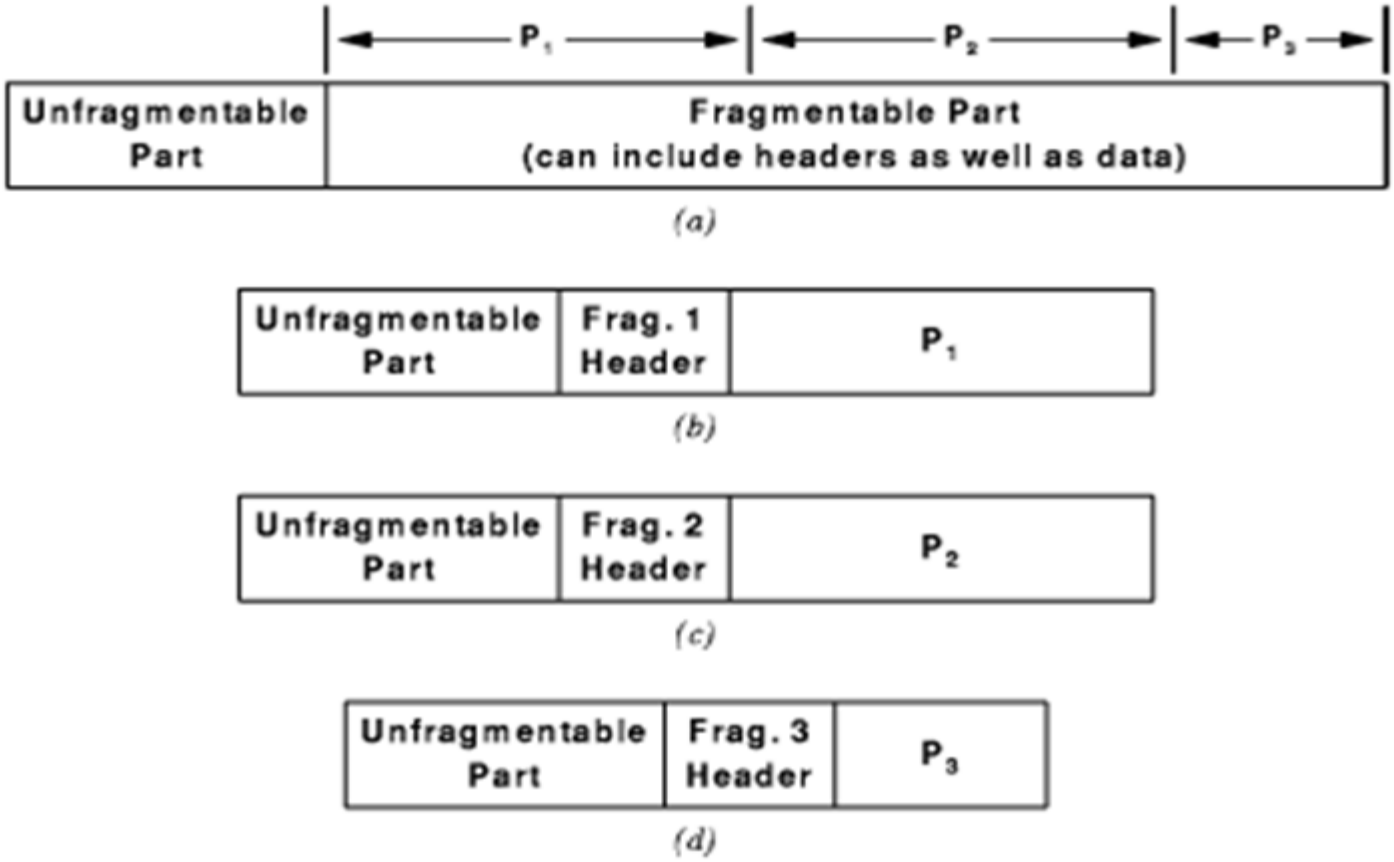
Routing – lista rutere de vizitat

Fragmentation – identificare fragmente

Authentication – verificare identitate transmiţător

Encrypted security payload – info despre conţinut criptat

Fragmentarea



fragmentare IPv6– la sursă

Ruterele ignoră datagramele mai lungi decât MTU

Sursa

Fragmentează pachetele

Descoperă path MTU

Caracter dinamic

- calea se poate schimba

Eficienţa – antet nu are spaţiu pierdut

Flexibilitate – noi antete pentru noi caracteristici

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

adrese 128-bit

Includ prefix reţea şi suffix gazdă

Fără clase de adresă – limita prefix/suffix oriunde

Tipuri speciale de adrese:

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

Notaţia de adresă

16 numere

105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255

Notaţie hexazecimală

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

Compresie zerouri

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

FF0C::B1

adrese IPv6 cu 96 zerouri prefix sunt interpretate ca adrese IPv4