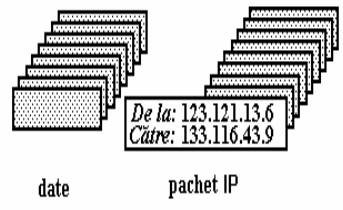
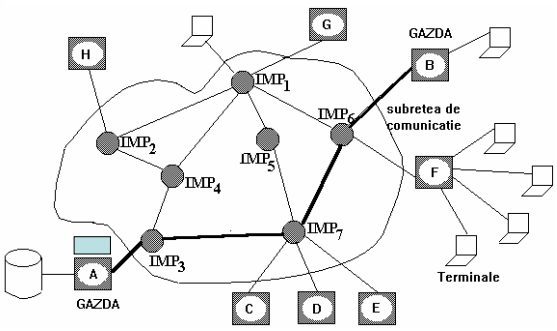
Nivelul retea

Principiul comunicării în Internet

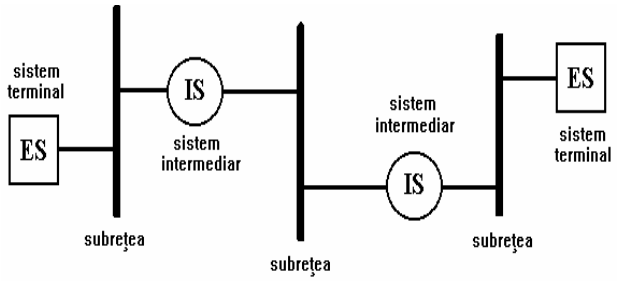


Inspirat din sistemul poştal

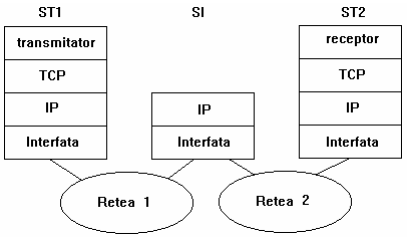
Drumul între utilizatorii A si B trece prin ruterele IMP3, IMP7 şi IMP6



Modelul unei reţele



Nivelele străbătute de pachete



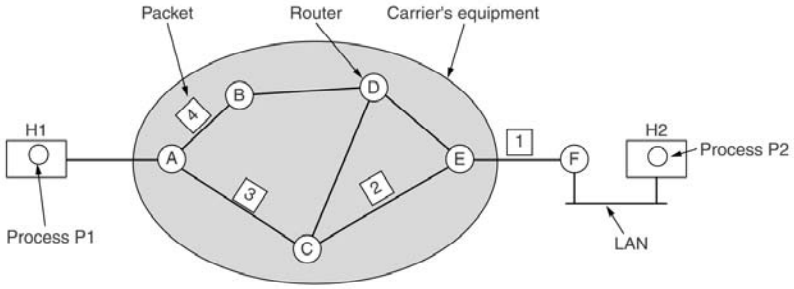
Funcțiile nivelului rețea

- dirijarea pachetelor
- adresarea
- evitarea congestionării rețelei

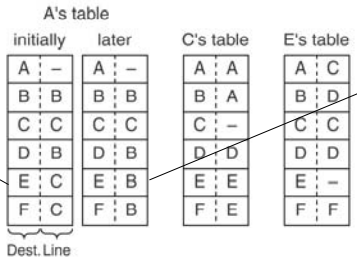
Aspecte principale

- servicii
  - orientate pe conexiune
  - ne-orientate pe conexiune
- organizarea internă
  - datagrame
  - circuite virtuale
- dirijarea (politici, algoritmi)

Organizarea internă - datagrame

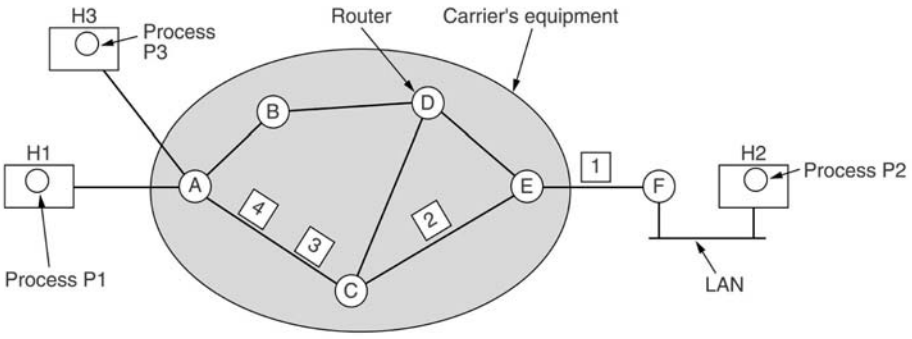


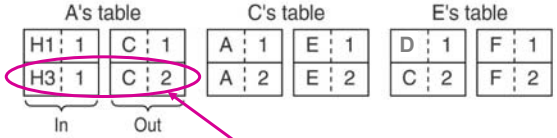
Folosita de pachertele 1, 2 si 3



Folosita de pachetul 4

Organizarea internă – circuit virtual





Nodul A re-numeroteaza circuitul virtual

Dirijarea - clasificare

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

Universitatea Politehnica Bucureşti - Facultatea de Automatica si Calculatoare

Algoritmi de dirijare - Calea cea mai scurtă

Algoritmul lui Dijkstra

nnod — **mulțimea nodurilor rețelei;**

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 — **mulțimea 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
                            }
     }
}
```


Dirijarea centralizată

Algoritmul lui Floyd

Utilizează tabloul distanțelor minime A :

A[i][j] distanța minimă de la nodul i la nodul j.

Initial, A[i][j] = l[i][j] pentru orice i si j.

Calculul drumurilor minime:

$$A[i][j]^k = \min (A[i][j]^{k-1}, A[i][k]^{k-1} + A[k][j]^{k-1})$$

Deoarece $A[i][k]^k = A[i][k]^{k-1}$

$A[k][j]^k = A[k][j]^{k-1}$

calculul se poate realiza cu o singură copie a tabloului A.

```
void Floyd()
 int i, j, k;
   for (i=1; i <= nnod; i++)
    for (j=1; j <= nnod; j++)
        { A[i][j] = l[i][j];
          if ( A[i][j] < lungmax)
             V[i][j] = j;
        }
   for (k=1; k <= nnod; k++)
     for ( i=1; i <= nnod; i++)
       for ( j=1; j <= nnod; j++)
          if (A[i][j] > A[i][k] + A[k][j])
                  {  A[i][j] = A[i][k] + A[k][j];
                          V[i][j] = V[i][k];
                  }
 }
```


Dirijarea distribuită (bazată pe starea legăturilor)

C tabloul distanțelor;

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

D tabloul distanțelor 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.

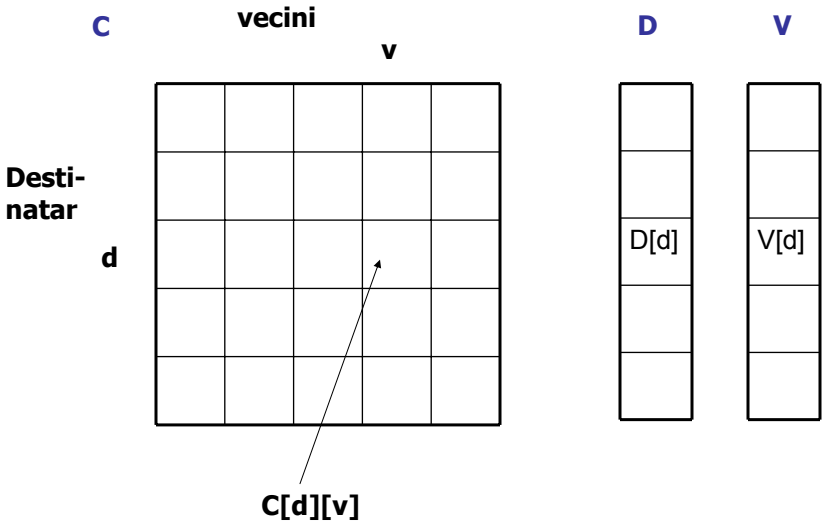
Evenimente tratate:

adăugarea unei noi legături;

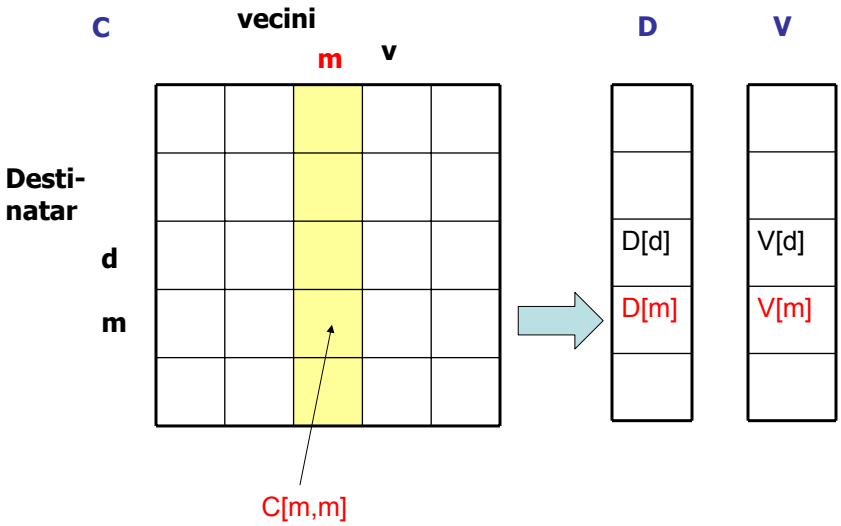
sesizarea modificării lungimii unei linii;

primirea unui mesaj de control de la un nod vecin.

Structuri de date pentru nodul crt



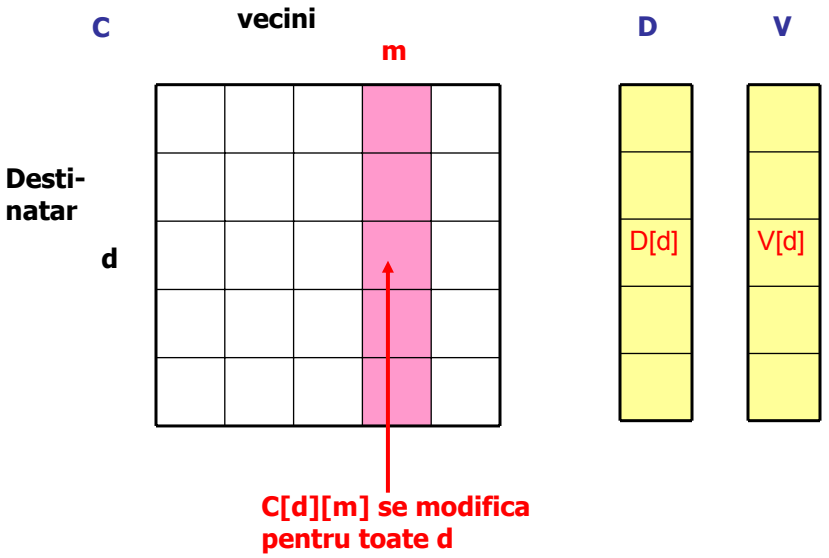
Adaugă legătura crt-m



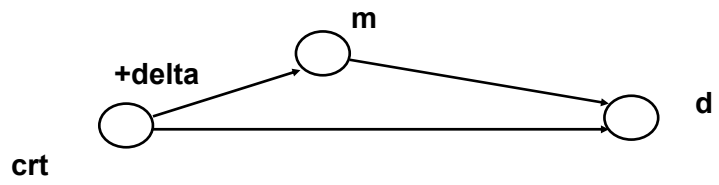
```
/* adauga legatura (crt,m), crt = nodul curent*/

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;
}
```

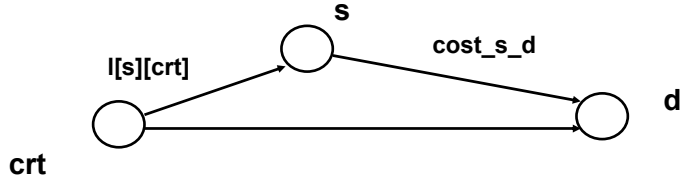

Schimbă cost crt-m cu delta_crt_m



```
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;
        }
    }
  }
```



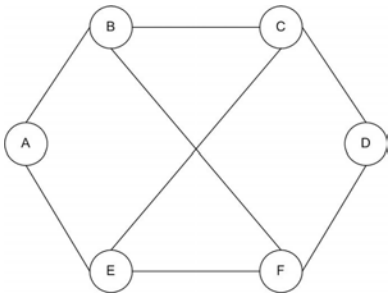
Starea legăturilor - variantă

Operații executate de fiecare ruter:

Descoperă vecinii şi află adresele de rețea (pachet HELLO)
Determină costul la fiecare vecin (pachet ECHO)
Alcătuieşte un pachet cu informațiile culese
- identitatea expeditorului
- număr de secvență
- vârsta
- listă de <vecin, cost legatura>

Transmite pachetul tuturor ruterelor (inundare)
Calculează cea mai scurtă cale la fiecare ruter (Dijkstra)

Dirijare folosind vectorul distanțelor



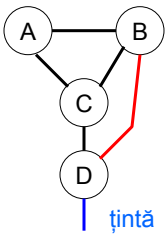
Următorii vectori au fost primiți de nodul C (lista include distanțele la nodurile A, B, C, D, E, F, în această ordine):

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).

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

La \ De 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

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



Problema numărării la infinit

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

Costurile **la țintă** 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
```

Cauza: C alege ruta prin A şi A alege ruta prin C.
In ultimul pas, C găseşte o cale mai ieftină prin D şi problema se rezolvă.
Pentru rețele deconectate, numărarea continuă la infinit.

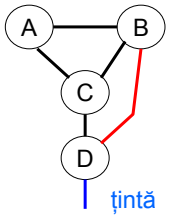
Soluții

Adoptate în RIP - Routing Information Protocol

"**simple split horizon**" omite rutele învățate 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

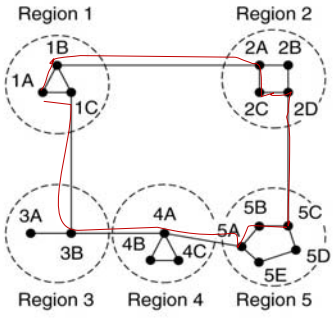


D:	dir, 1	dir, 1	dir, 1
B:	unreach	unreach	C, 12
C:	B, 3	D, 11	D, 11
A:	B, 3	unreach	C, 12

Protocoale care folosesc split horizon

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

Dirijare ierarhică



(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

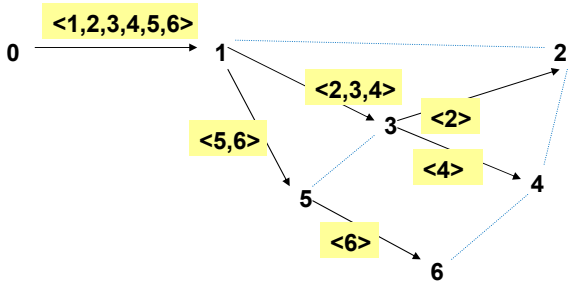
(c)

Reducere număr intrări
17 -> 7

Penalizare la dest. 5C

Difuzare şi multicast

- Punct la punct - Trimite un pachet fiecărei destinații
- Inundarea
  - **Generează prea multe pachete**
  - **Copiile sunt distruse**
- Dirijarea multidestinație
  - **Pachetul conține lista adreselor de destinație**
- Arbore de acoperire



Difuzare – urmărirea căii inverse

(a) O subrețea **(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

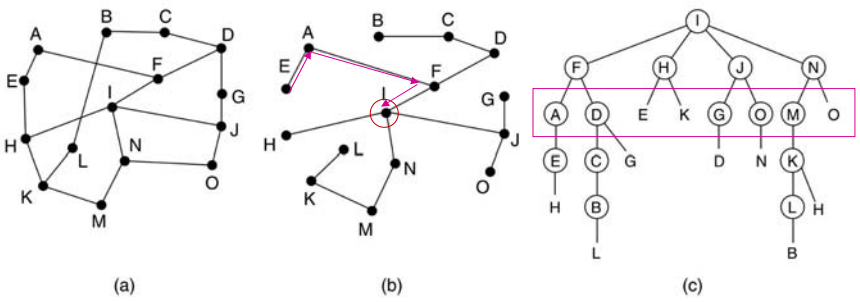
(c) functionarea algoritmului căilor inverse: când un pachet ajunge la un ruter:

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

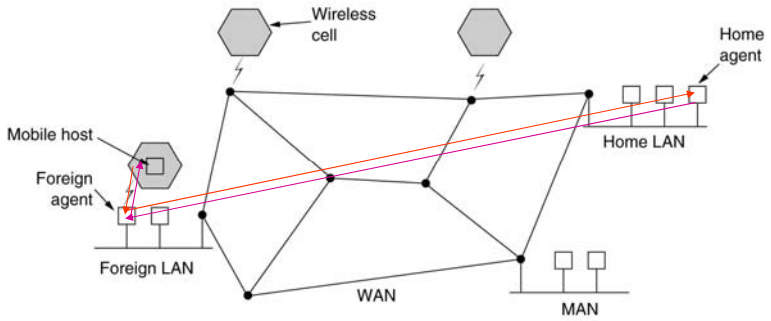
dacă da -> este trimis pe toate celelalte linii

altfel -> este distrus

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



Dirijarea pentru gazde mobile



Gazda mobilă cere să se înregistreze cu un agent străin (dă adresa de acasă)

Agentul străin contactează agentul de acasă (transmite adresa străină + info securitate)

Agentul de acasă validează

Agentul străin înregistrează gazda mobilă şi o informează

Dirijarea în rețele ad hoc

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

rețea ad hoc = graf

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

Fiecare nod = ruter + gazdă

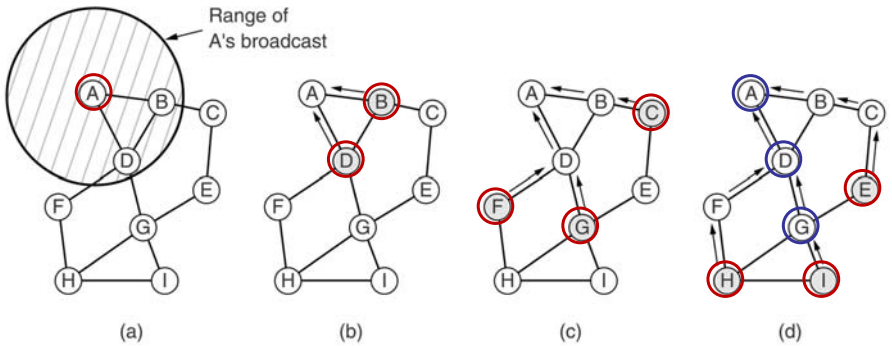
Conține

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

Cum functioneaza?

Exemplu: A vrea 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 informația 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, Dest address, Hop count sunt copiate**
- **Destination sequence # luat din contorul propriu**
- **Lifetime = cât timp rămâne valid**

Prelucrarea la alte noduri

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

Source address	Destination address	Destination sequence #	Hop count	Lifetime

Intreținerea rutelor

G cade

D descoperă (se folosesc mesaje Hello periodice)

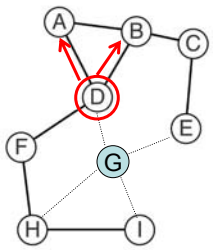
D află că G a fost utilizat pe rute către E, G şi I

D anunță 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)

Protocol IPv4

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

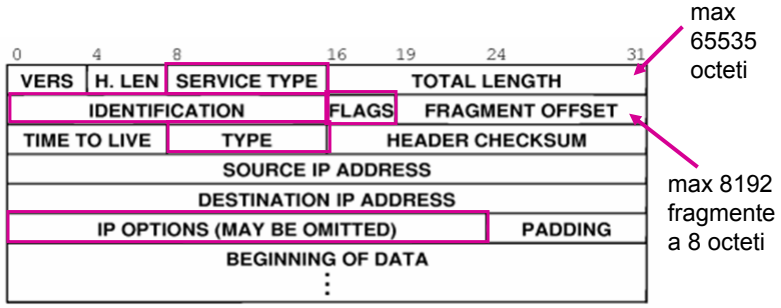
TYPE = protocol (TCP, UDP, etc.)

IDENTIFICATION datagrama de care aparține fragmentul

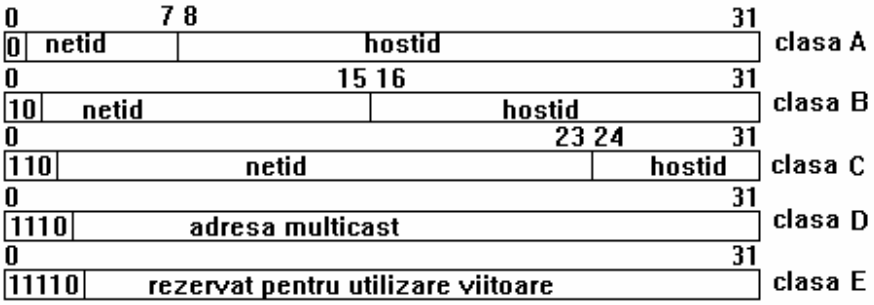
FLAGS DF = Don't Fragment

MF = More Fragments

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



Adrese IP



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

Adrese speciale

Prefix	Suffix	Tip de adresă	Scop
toți 0	toți 0	acest calculator	Folosită la bootstrap
network	toți 0	network	Identifică rețeaua
network	toți 1	broadcast	broadcast în rețeaua specificată
toți 1	toți 1	broadcast	broadcast în rețeaua locală
127	orice	loopback	testare

Algoritm de rutare IP

Tabela rutare – tipuri intrări

<network, 0> pentru rețele distante

<this-network, host> pentru gazde locale

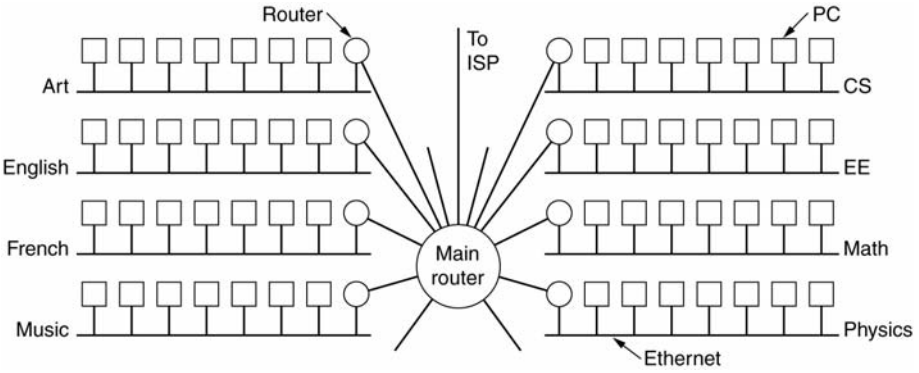
Tabele rutare separate pentru diferite clase de adrese

Căutare prin: indexare (A şi B) sau hashing (C)

Algoritm Rutare

Extrage adresa destinatiei DADR din datagrama
Calculeaza adresa de retea a destinatiei NADR
if NADR apartine unei retele direct conectate
    transmite datagrama gazdei de destinatie intr-un frame
else if DADR apare in tabela dirijare
    ruteaza datagrama conform tabelei
else if NADR apare in tabela dirijare
    ruteaza datagrama conform tabelei
else ruteaza datagrama la un ruter implicit

Subrețele

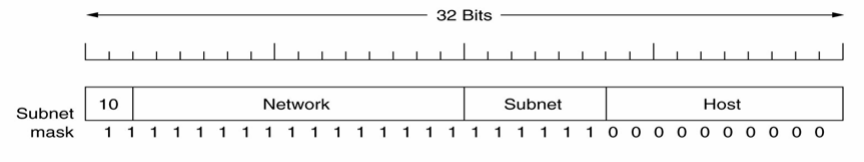


Organizarea în subrețele este invizibilă în afara rețelei

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

Ruterele de subrețea le livrează gazdelor

Adresarea în Subrețele



Exemplu: O reţea de clasă B impărţită în 64 subreţele

Tabela de rutare are intrări suplimentare pentru
(this-network, subnet, 0)
(this-network, this-subnet, host)

Modif algoritm rutare:

Foloseşte masca de subretea pentru a separa retea+subretea şi gazda
Dirijează spre subreţea (subnet)
Doar în subreţeaua curentă dirijează direct către gazdă (host)

Elimină limitările schemei de adresare bazată pe clase

CIDR – Classless InterDomain Routing

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

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

Notația CIDR 194.24.0.0/21 => din cei 32 de biți ai adresei

rețea+subrețea = 21 biți

gazda = 11 biti

Exemplu

	Adresă	Mască
C:	**11000010 00011000 00000000 00000000**	**11111111 11111111 11111000 00000000**
E:	**11000010 00011000 00001000 00000000**	**11111111 11111111 11111100 00000000**
O:	**11000010 00011000 00010000 00000000**	**11111111 11111111 11110000 00000000**

Alocare adrese: zona de adrese pentru Oxford incepe la o frontieră de 4096 octeți

Algoritm rutare

(Adresa IP AND masca) comparată cu primul câmp al fiecărei intrări în tabelă

La mai multe coincidențe se alege masca cea mai lungă

Universitatea Politehnica Bucureşti - Facultatea de Automatica si Calculatoare

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

Soseste pachet cu adresa 194.24.17.4

11000010 00011000 00010001 00000100

AND cu masca de la Cambridge

11000010 00011000 00010000 00000000 = 194.24.16.0 ➔ nepotrivire

AND cu masca de la Edinburgh

11000010 00011000 00010000 00000000 = 194.24.16.0 ➔ nepotrivire

AND cu masca de la Oxford

11000010 00011000 00010001 00000000 = 194.24.16.0 ➔ **potrivire**

➔ daca nu sunt alte potriviri -> foloseste intrarea pentru Oxford

Reducere dimensiune tabelă - agregarea intrărilor pentru o aceeaşi linie de ieşire

	Adresă	Mască
C:	11000010 00011000 000|00000 00000000	11111111 11111111 11111000 00000000
E:	11000010 00011000 000|01000 00000000	11111111 11111111 11111100 00000000
O:	11000010 00011000 000|10000 00000000	11111111 11111111 11110000 00000000

Pentru toate adresele din C, E, O, un ruter trimite pe aceeaşi interfață

Intrare agregată

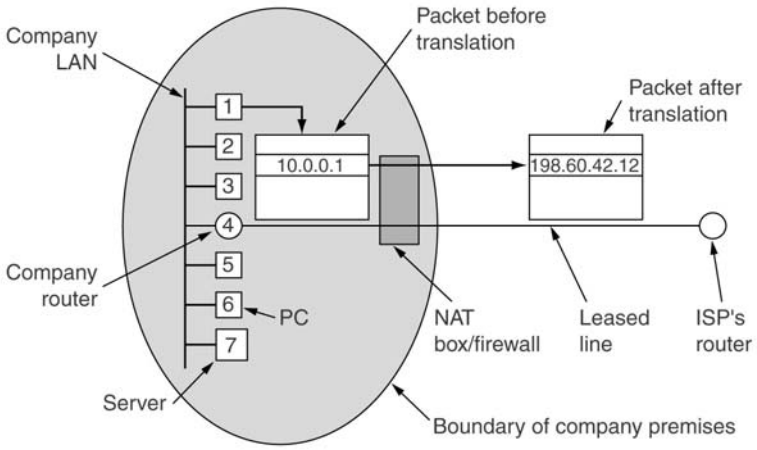
Adresă	Mască
11000010 00011000 000|00000 00000000	11111111 11111111 11100000 00000000

Corespunde cu 194.24.0.0/19

Universitatea Politehnica Bucureşti - Facultatea de Automatica si Calculatoare

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



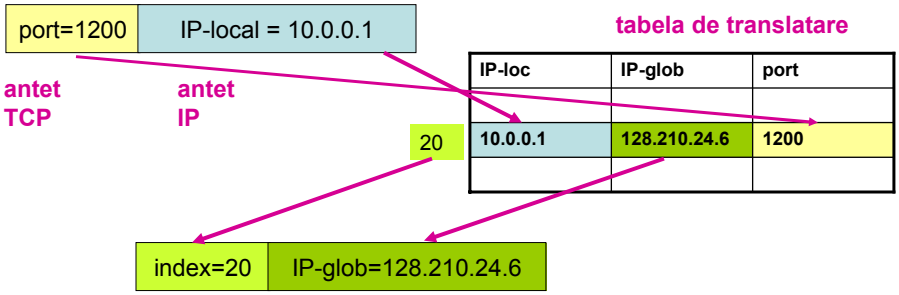
Principiul NAT

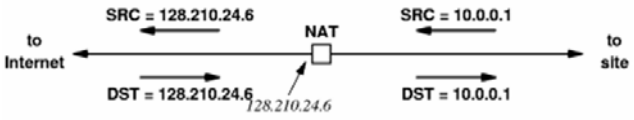
Foloseşte

adresa IP + număr port transmitator
tabela de translatare

Transmisie

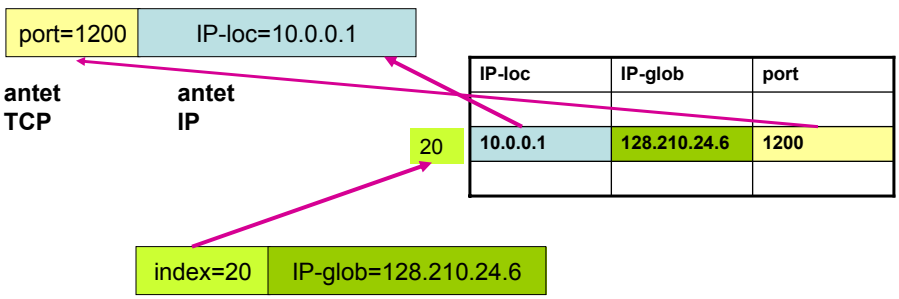
înlocuieşte adresa IP locală cu o adresă IP globală
memorează (in tabela de translatare) corespondența şi număr port
inlocuieşte număr port cu index în tabela translatare
re-compune sumele de control IP şi TCP





Receptie

obține 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



Protocoale de control in Internet

Rezolvarea adreselor

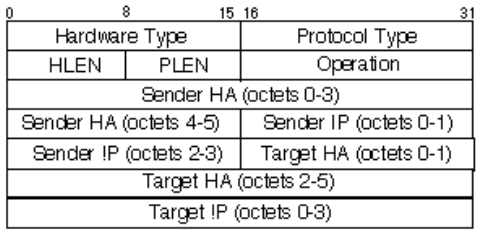
Mapare adresa de protocol şi adresa hardware

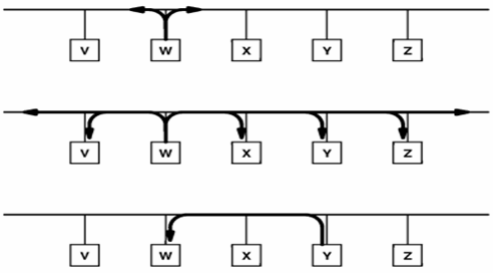
Tehnici

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

Address Resolution Protocol

livrare mesaj ARP





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

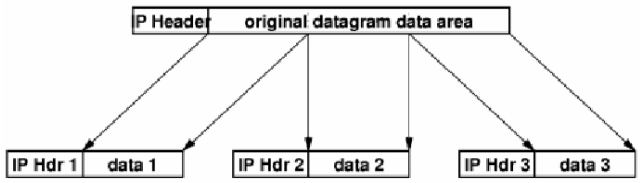
ICMP foloseşte IP ptr transmisie <=> IP foloseşte ICMP pentru raportare de erori

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)

MTU – Maximum Transmission Unit

Fragmentarea



Reasamblarea

Universitatea Politehnica Bucureşti - Facultatea de Automatica si Calculatoare

Folosire ICMP pentru aflare path MTU

- **Path MTU = MTU 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**

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**

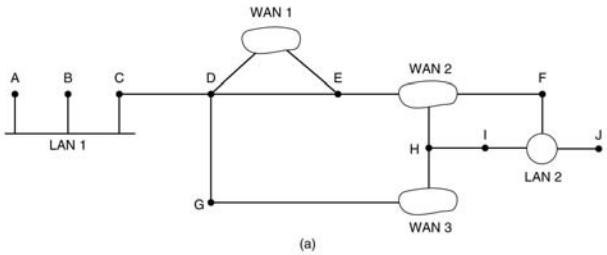
OSPF

Suportă:

Linii punct la punct intre două rutere

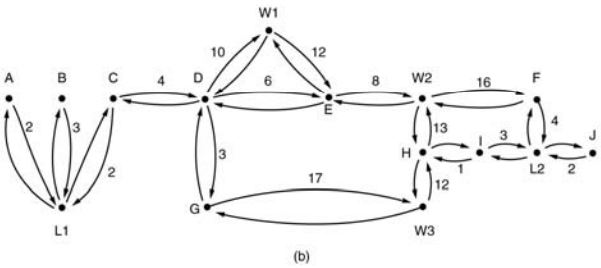
LANs

WANs



(a)

Modelul de graf



(b)

OSPF

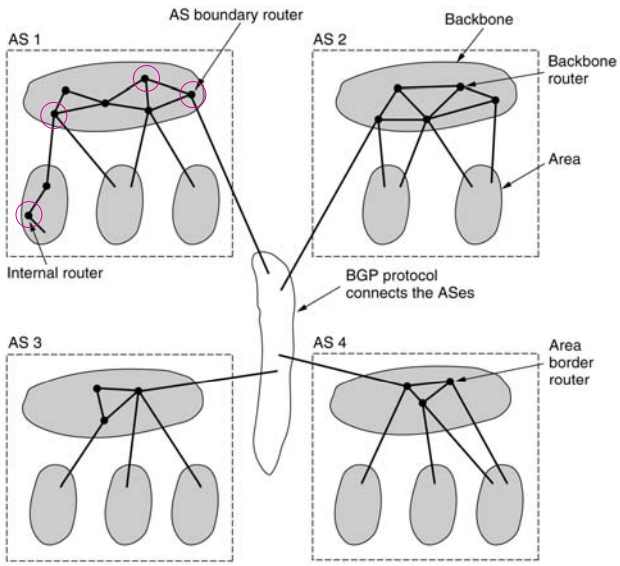
Fiecare AS are mai multe zone

Tipuri de rutere:

- **interne**
- **de coloană vertebrală**
- **de graniță zonală**
- **de graniță AS**

OSPF foloseşte schimb inf între rutere adiacente

In LAN, un ruter desemnat este adiacent cu toate celelalte



Calcul rute

Nivel 1 (zona)

Fiecare ruter din zonă calculează separat căile cele mai scurte pe baza info de la celelalte

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 backbone

acceptă info de la area border routers
calculateaza cele mai bune rute intre orice ruter backbone şi celelalte rutere
propagă info înapoi la area border routers

Area border routers avertizează ruterele din zonă

Fiecare ruter selectează cea mai bună ieşire spre backbone

BGP – Border Gateway Protocol

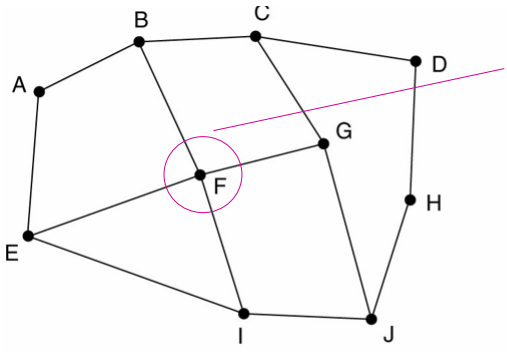
Algoritmi orientați pe aspectele politice, de securitate, economice

Rețea = ASes şi conexiunile

Protocol = vectorul distanțelor

Tabelele de dirijare conțin şi rutele spre destinație

Comunică vecinilor căile utilizate efectiv



(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)

IPv6

Motivaţii

Spațiul de adrese

32 biți = peste un milion de rețele

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

214 adrese de rețea Clasa B, multe folosite

Tip servicii

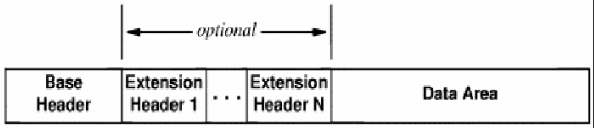
Aplicații diferite au cerințe diferite de livrare, siguranță şi viteză

IPv4 are tip de serviciu dar adesea nu este implementat

Caracterizare

- **format antet**
- **antete extensii**
- **support audio şi video**
- **protocol extensibil**
- **spațiu adresa**
- **multicast**

IPv6 - format datagrama



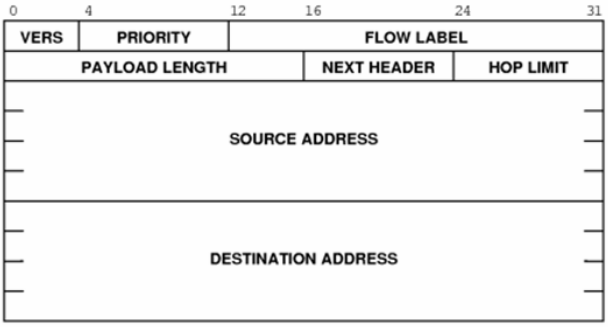
IPv6 format Base header

Base header
lungime fixă = 40 octeti

Prioritate - clasa de trafic

FLOW LABEL - asociază datagramele unui flux

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

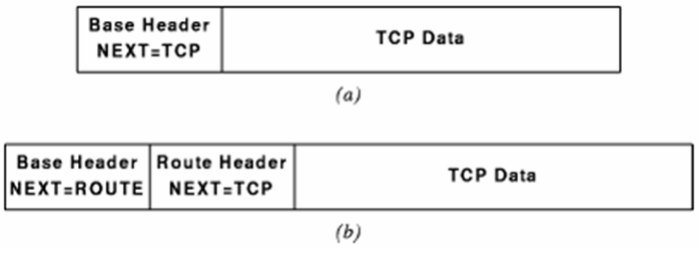


Conţine mai puţine info decât antet IPv4

Restul de info în extensii

NEXT HEADER defineşte tipul datelor (ex. TCP)

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



IPv6 – antete extensie

Hop-by-hop options – info pentru rutere
- suport datagrame excedând 64K (jumbograme)

Destination options – info aditionale pentru destinație
- nefolosit

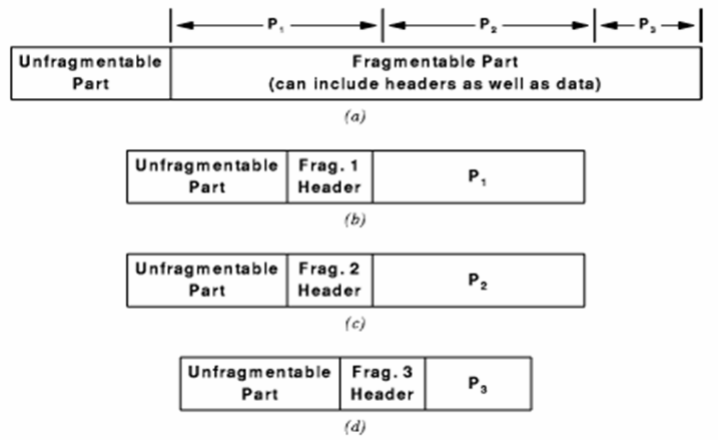
Routing – lista rutere de vizitat

Fragmentation – identificare fragmente

Authentication – verificare identitate transmițător

Encrypted security payload – info despre conținut criptat

Fragmentarea



fragmentare IPv6– la sursă

Ruterele ignoră datagramele mai lungi decât MTU

Sursa

Fragmentează pachetele

Descoperă path MTU

Caracter dinamic

- calea se poate schimba

Eficiența – antet nu are spațiu pierdut

Flexibilitate – noi antete pentru noi caracteristici

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

adrese 128-bit

Includ prefix rețea şi suffix gazdă

Fără clase de adresă – limita prefix/suffix oriunde

Tipuri speciale de adrese:

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

Notația de adresă

16 numere

105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255

Notație hexazecimală

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

Compresie zerouri

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

FF0C::B1

adrese IPv6 cu 96 zerouri prefix sunt interpretate ca adrese IPv4