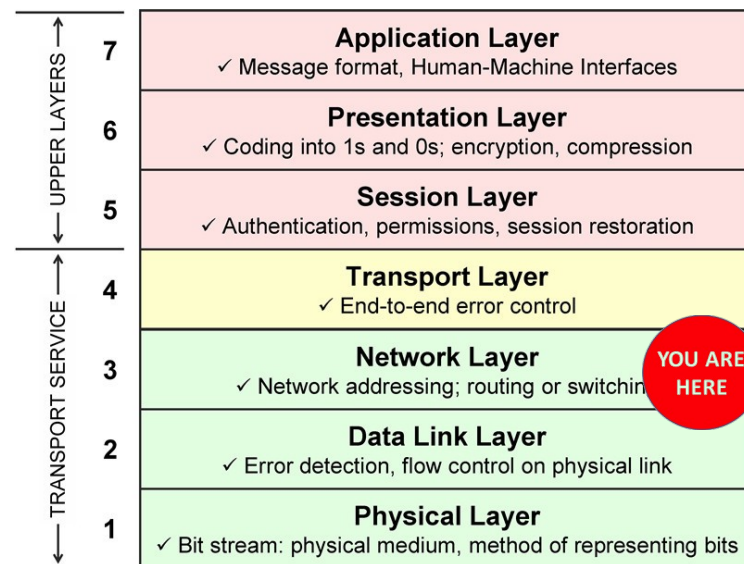




# Nivelul retea

# Cuprins

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



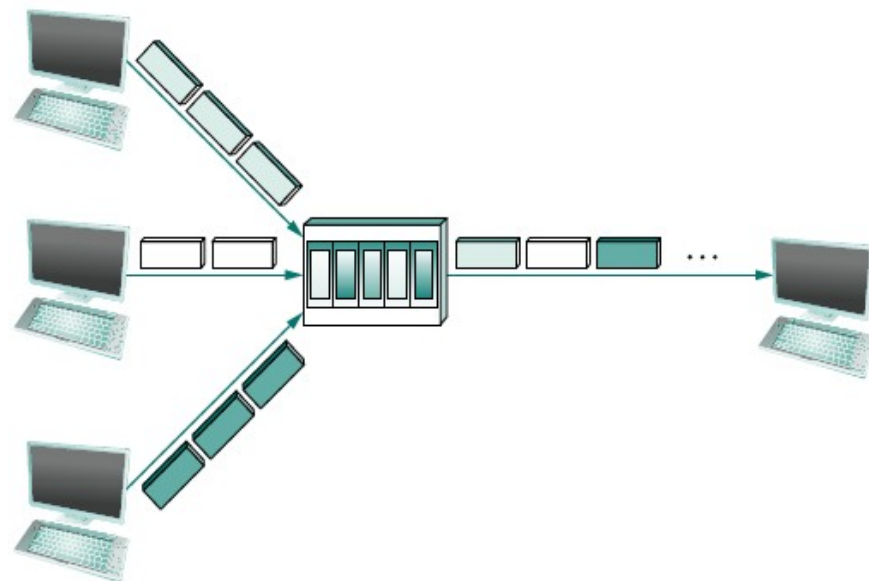
# De ce este nevoie de pachete?

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

**Multiplexare** – se alocă transmisiilor

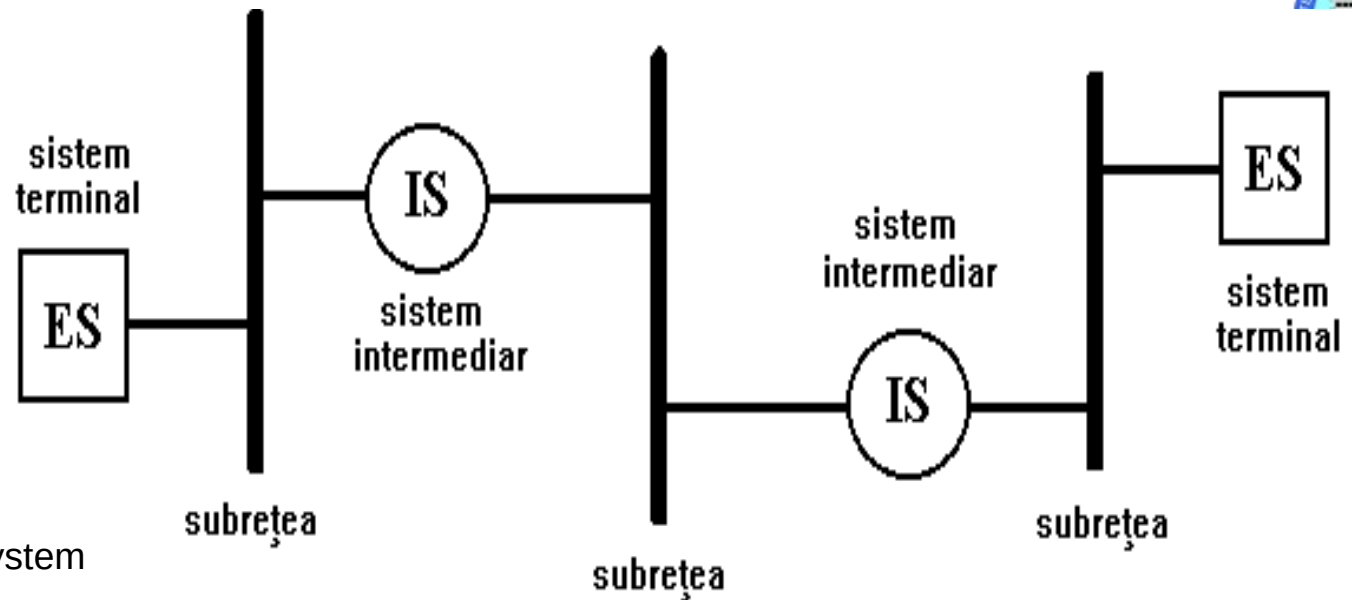
- **sloturi de timp** - STDM – Synchronous Time Division Multiplexing
- **subcanale** de frecvențe diferite – FDM – Frequency Division Multiplexing

- **multiplexare statistica** – STMD – sloturile sunt alocate **la cerere**; dacă o singură pereche de noduri are de transmis, nu așteaptă ptr. slot;
- pentru a evita **acapararea** legăturii, transmisia se face în **pachete cu dimensiune limitată**

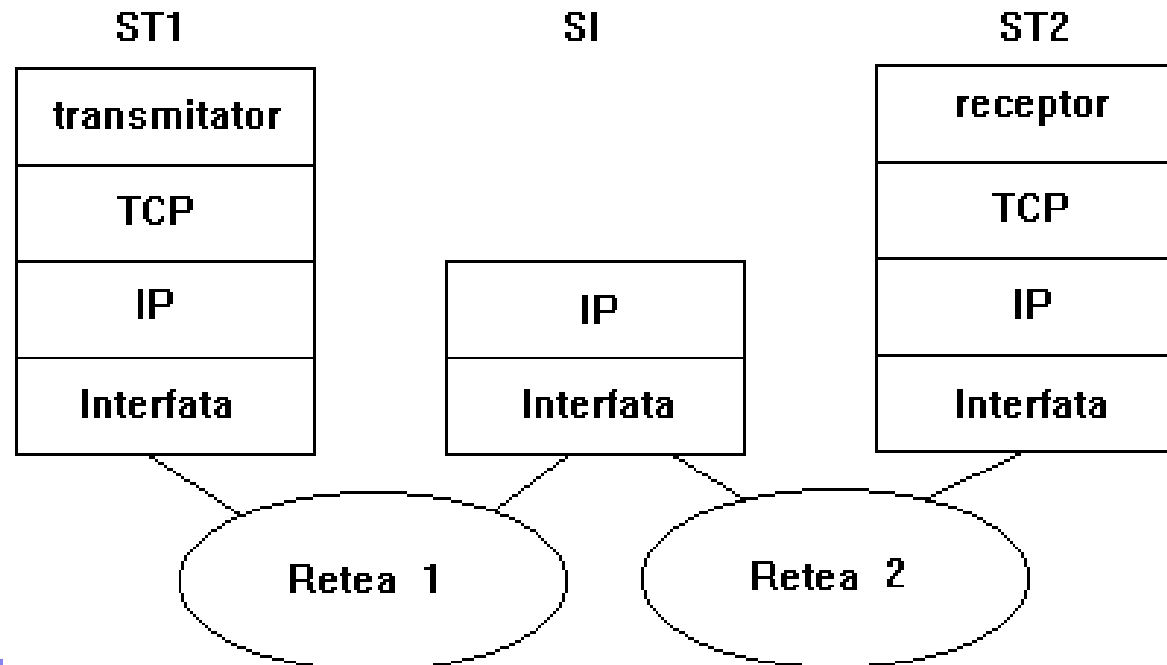


## Modelul Internetului

- ES – End system
- IS – Intermediate system



## Nivelele străbătute de pachete





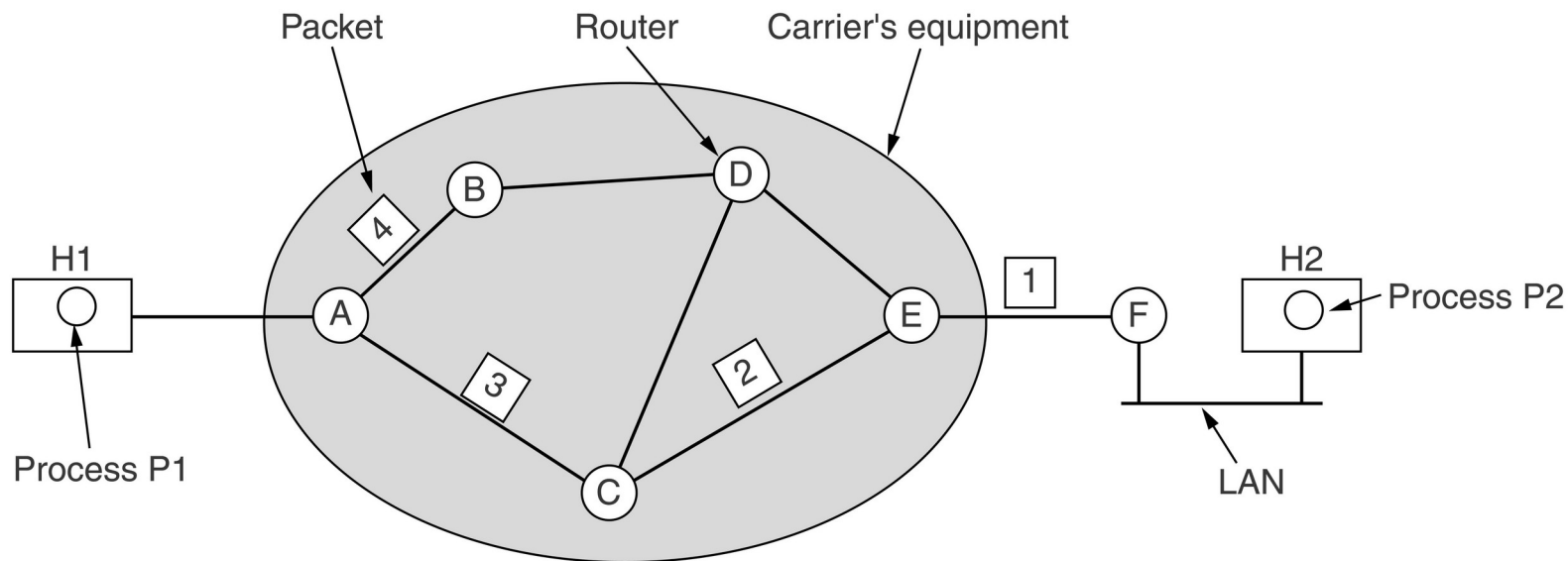
## Funcțiile nivelului rețea

- dirijarea pachetelor
- Adresarea

## Aspecte principale

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

# Organizarea internă - datagrame



A's table

initially

later

C's table

E's table

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

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

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

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

Dest. Line

Folosita de  
pachetele  
1, 2 si 3

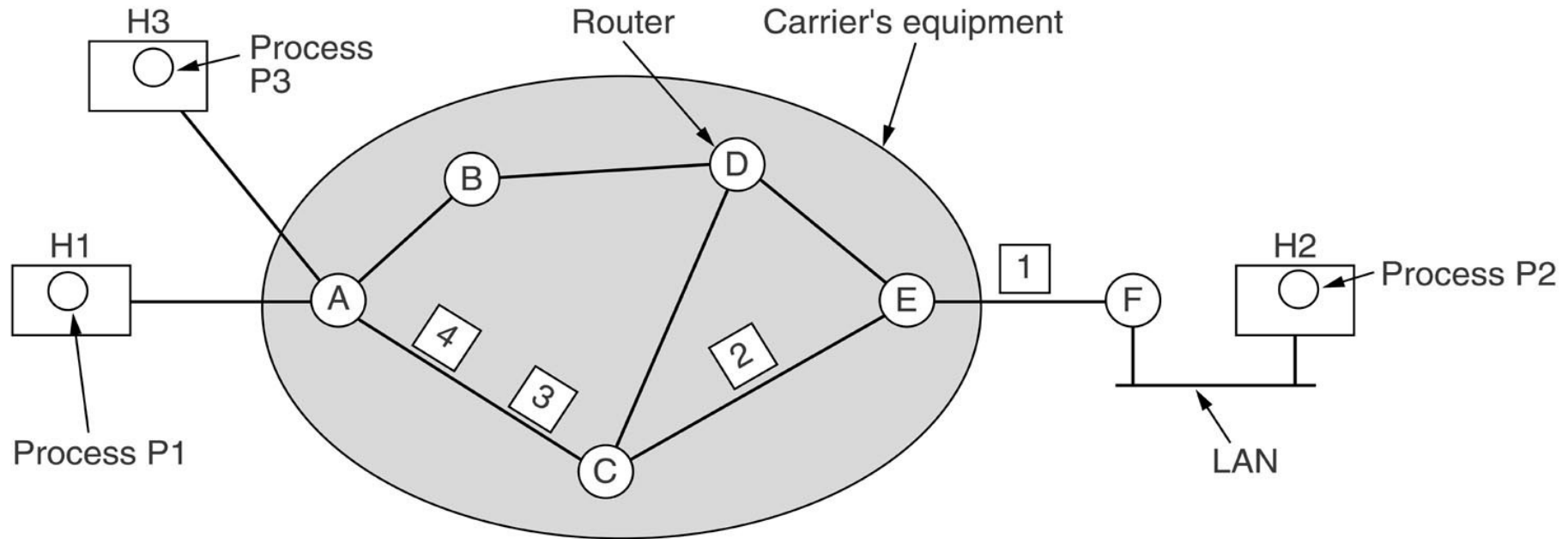
Folosita de  
pachetul 4



# Caracteristici datagrame

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

# Organizarea internă – circuit virtual



A's table

H1	1	C	1
H3	1	C	1

In      Out

C's table

A	1	E	1
A	1	E	2

E's table

C	1	F	1
C	2	F	2

Nodul A re-numeroteaza circuitul virtual





# Caracteristici circuit virtual

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



# IP



# Protocolul IP

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



# Protocol IPv4 – formatul pachetului

VERS = IPv4, IPv6....

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

PROTOCOL = (TCP, UDP, etc.)

OPTIONS:

Security

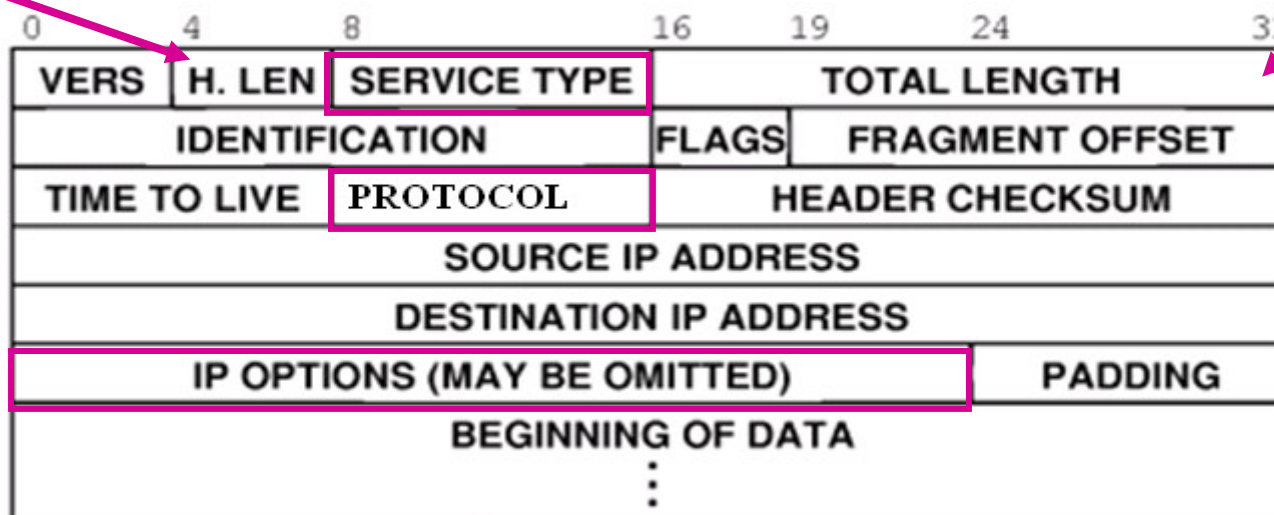
Strict source routing

Loose source routing

Record route

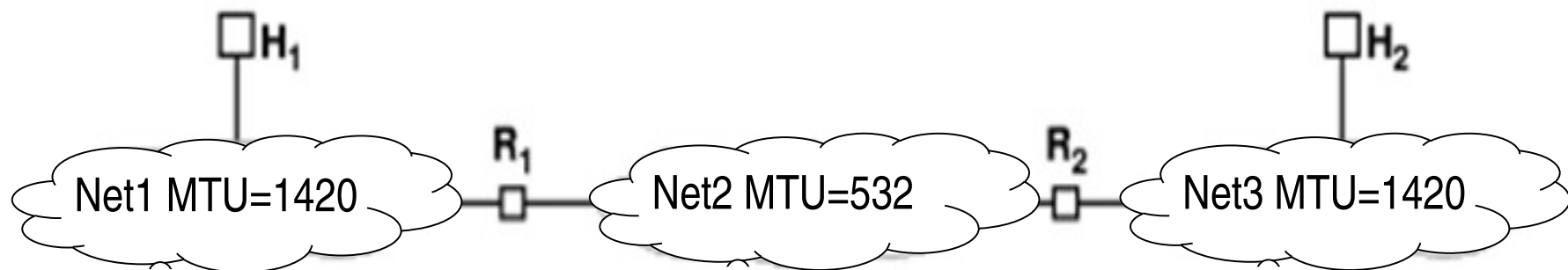
Timestamp

numar  
cuvinte  
de 32  
biti

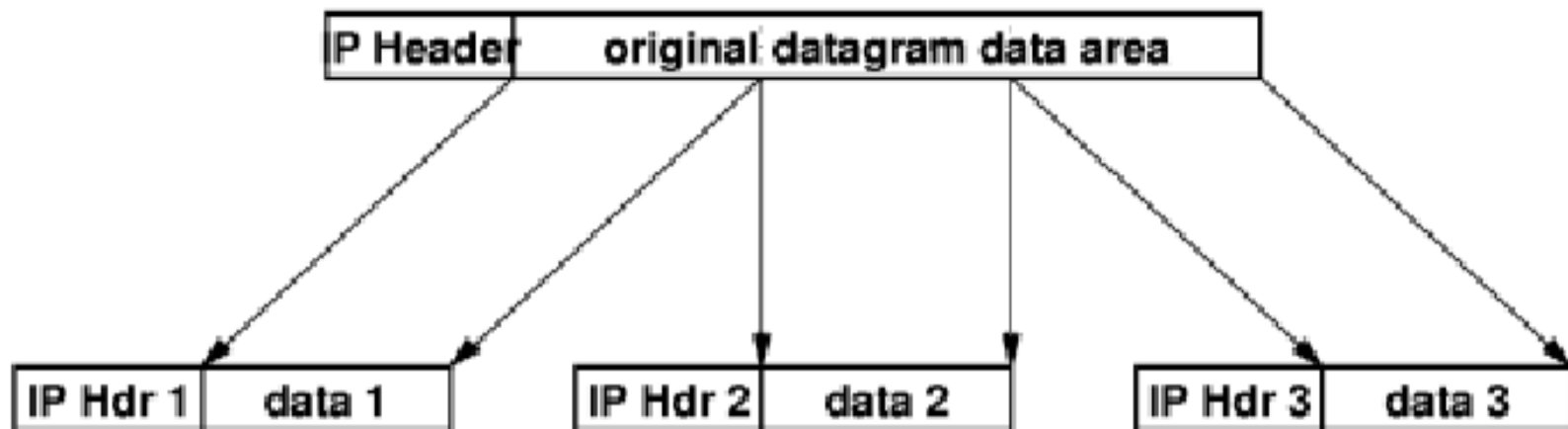


max  
65535  
octeti

# MTU – Maximum Transmission Unit



MTU – Maximum transmission unit  
**Fragmentarea se face la  $R_1$**



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



# Protocol IPv4 – formatul pachetului

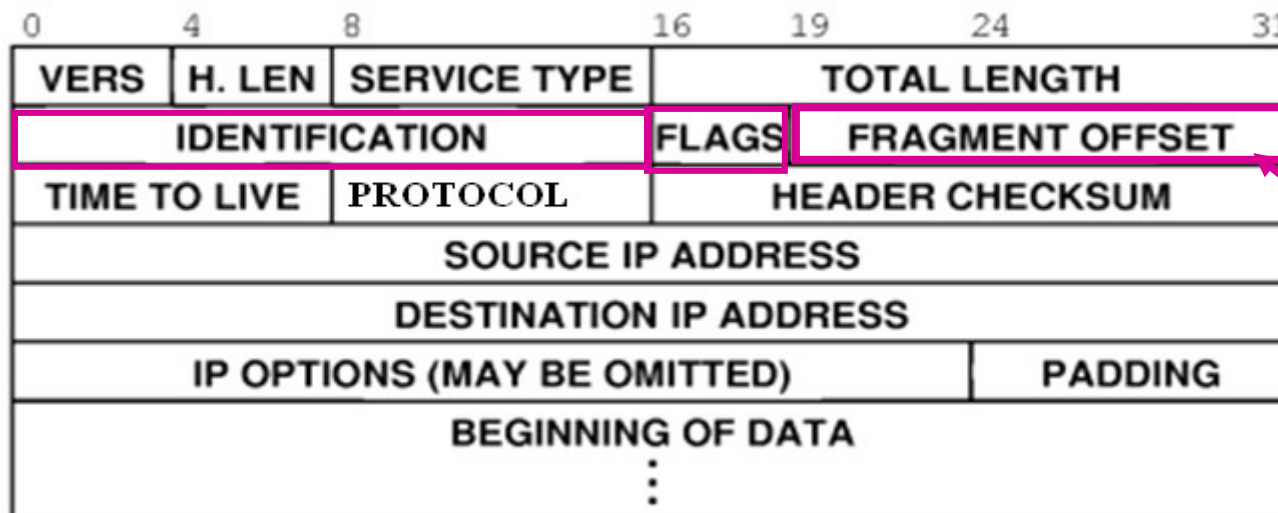
IDENTIFICATION Id datagrama de care aparține fragmentul

FRAGMENT OFFSET pozitia fragmentului in pachet

FLAGS

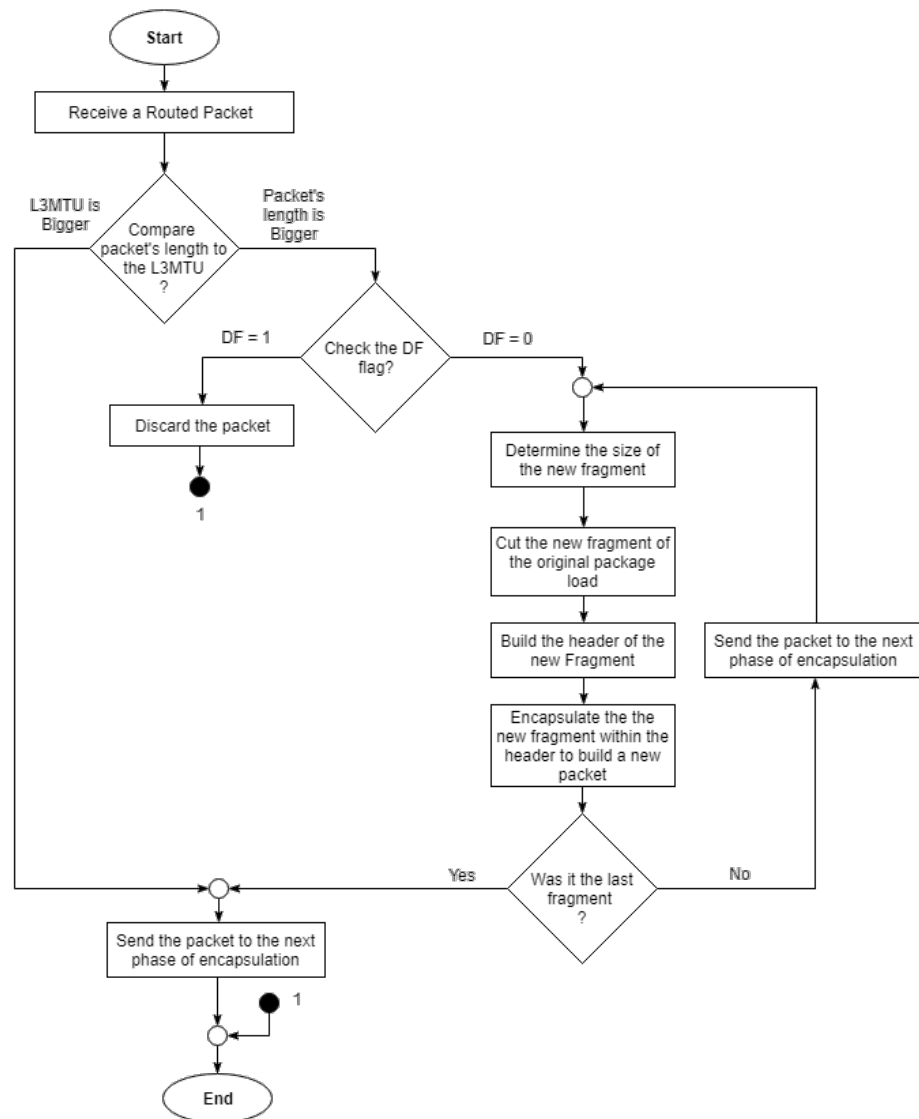
**DF** = Don't Fragment

**MF** = More Fragments



max 8192  
fragmente  
a 8 octeti

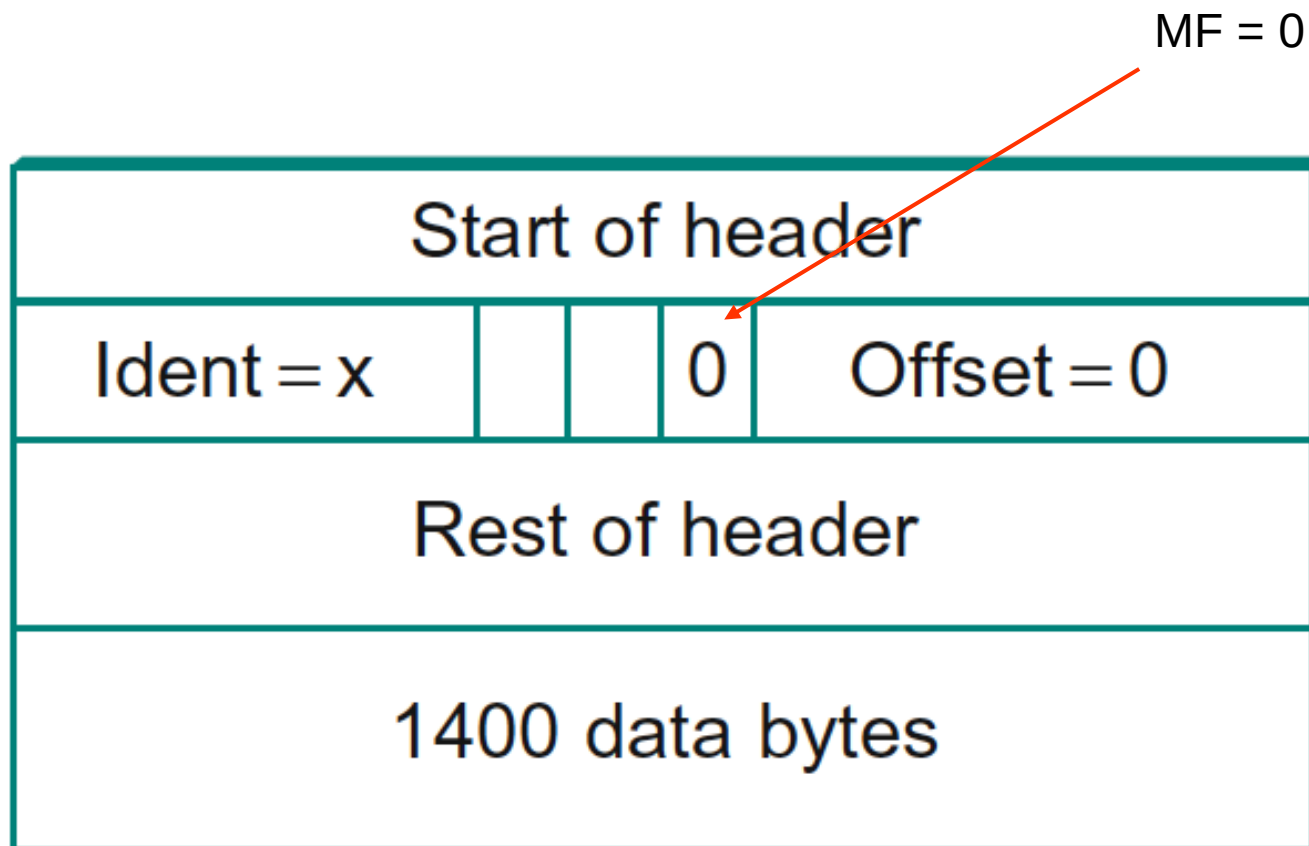
# Protocol IPv4 – fragmentarea





# Campurile de antet pentru fragmentare (1)

- a) pachet transmis ne-fragmentat prin Net1 cu MTU = 1420  
are: 1400 octeti de date  
20 octeti de antet IP





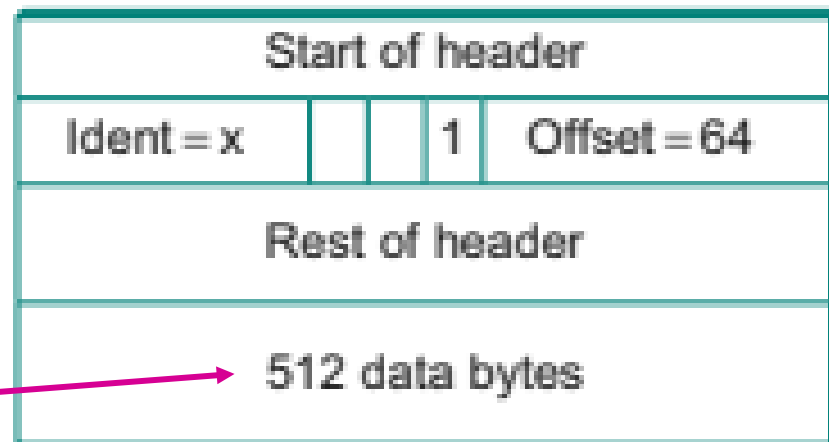
# Campurile de antet pentru fragmentare (2)

Net2 are  $MTU = 532$

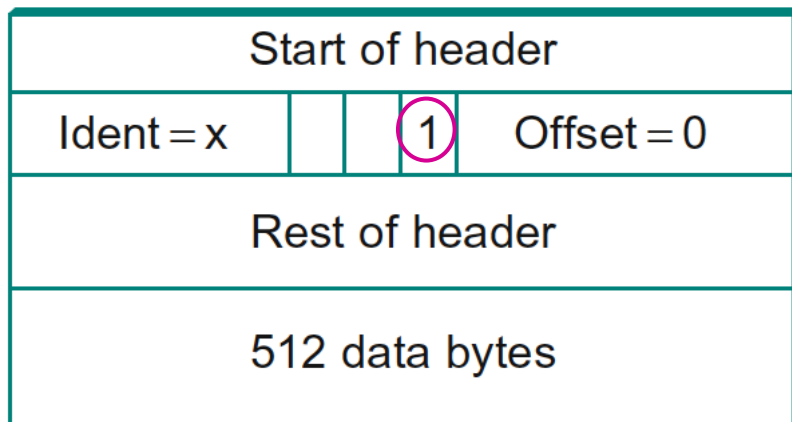
Ruterul R1 segmenteaza pachetul in 3 fragmente (fig b1-b3)

- lungimea fiecarui segment este multiplu de 8 octeti
- **offset**-ul numara grupuri de cate 8 octeti
- $Offset = 64 = 512 / 8$   
unde este restul pana la 532 octeti?

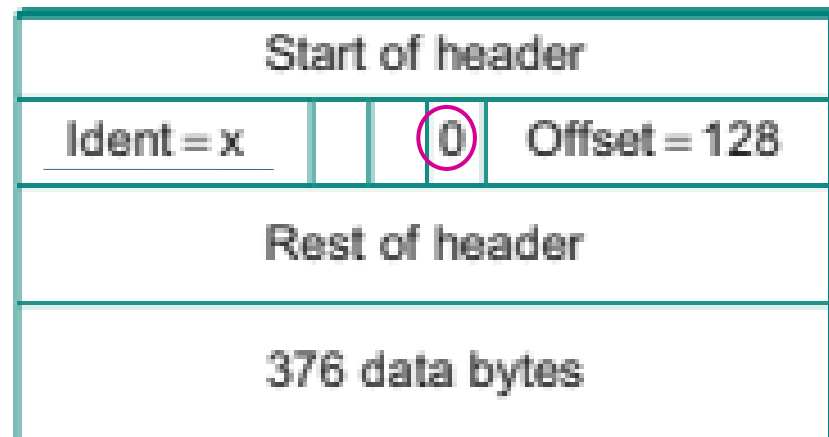
b2



b1



b3





# ARP



# Conversie adresa IP – adresa fizică

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

**Nu există** o asociere directă între adresa fizică și adresa IP

De ex.

- adresa IP are 32 biți
- adresa Ethernet are 48 biți

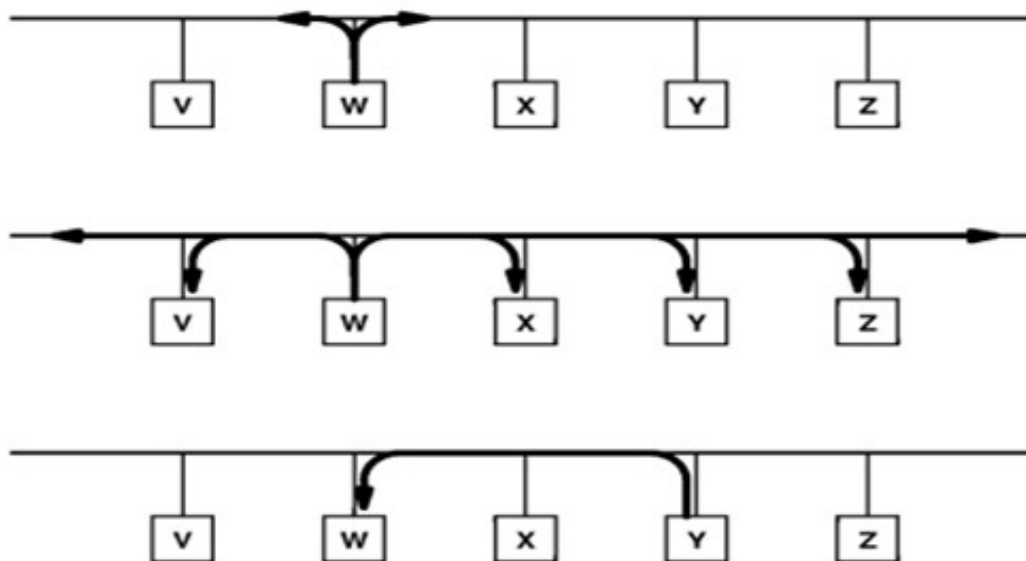
Pentru mapare se pot folosi

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

# ARP - Address Resolution Protocol

Secvența de mesaje pentru a afla adresa fizică

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



# Conversie adresa IP – adresa fizică

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

**Format mesaje ARP** – conceput pentru diverse categorii de adrese

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



# ICMP

# ICMP- Internet Control Message Protocol

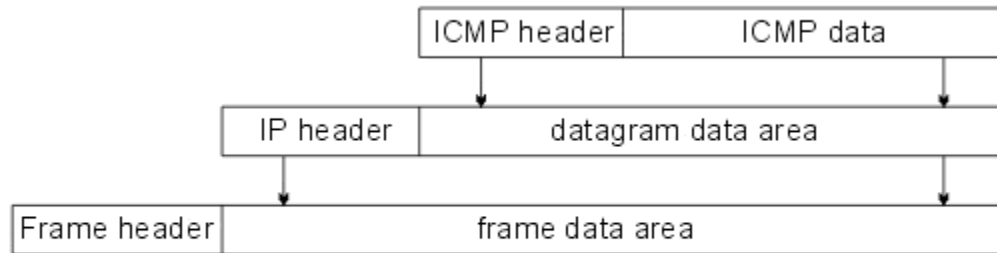
Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request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

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

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

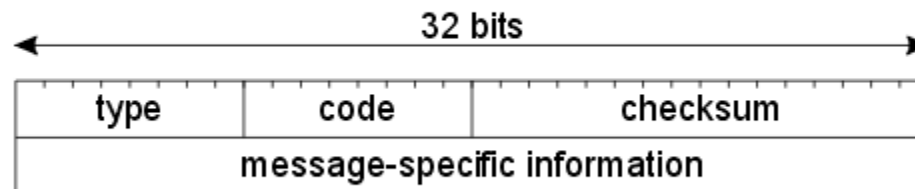
- **Trasare ruta** (**traceroute** trimite serie de datagrame cu valori TIME TO LIVE crescătoare și primește mesaje ICMP **Time exceeded** din care extrage adresa ruterului)

# ICMP header



ICMP encapsulation

## ICMP packet format





# ICMP- Internet Control Message Protocol

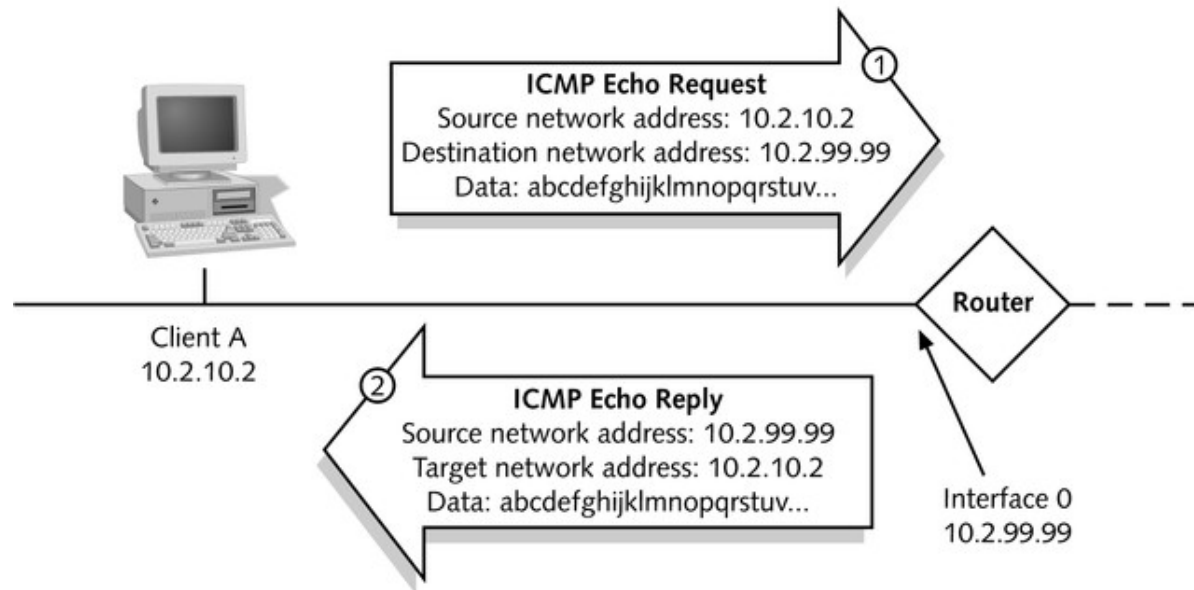


Figure 4-1 PING utility uses ICMP echo requests and replies

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

Type 8: Echo request

Type 0: Echo reply

Type: 8 or 0	Code: 0	Checksum
Identifier		Sequence number
Optional data		
Sent by the request message; repeated by the reply message		



# Folosire ICMP pentru aflare **path MTU**

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

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



# Adrese IP

# Clase de adrese IPv4

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

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

0	7	8	31	
0	netid		hostid	clasa A
0	15	16	31	
10	netid		hostid	clasa B
0	23	24	31	
110	netid		hostid	clasa C
0			31	
1110	adresa multicast			clasa D
0			31	
11110	rezervat pentru utilizare viitoare			clasa E

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

## Câteva adrese speciale

Prefix (Network)	Suffix (Host)	Semnificație	Scop
toți 0	toți 0	acest calculator	Folosită cand nodul inca nu are o adresa
network	toți 0	network	Identifică rețeaua
network	toți 1	broadcast	broadcast în rețeaua specificată
toți 1	toți 1	broadcast	broadcast în rețeaua locală
17	orice	loopback	testare

### Notății pentru adrese

binară 11000010 00011000 00010001 00000100

zecimală 194.24.17.4



# Tabele de dirijare

Orice pachet conține o **adresa IP** a destinatarului, cu două părți  
<adresa\_retea, adresa\_nod>

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

**Rol ruter** – primește un pachet și

- îl livrează **gazdei** de destinație (dacă este în aceeași rețea)
  - Toate nodurile care au aceeași **adresa\_retea** sunt situate în **aceeași rețea fizică** și pot comunica direct prin **legătura de date** (transmit **cadre**)
- altfel, îl 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  $\langle \text{adresa\_retei}, \text{adresa\_gazda} \rangle$  destinație din datagrama  
Caută o intrare cu  $\text{adresa\_retei}$  în tabela de dirijare

*if*  $\text{adresa\_retei}$  apare în tabela de dirijare  
    *if*  $\text{adresa\_retei}$  indică o rețea direct conectată **then**  
        transmite datagrama direct la  $\text{adresa\_gazda}$   
    **else** transmite datagrama următorului ruter (*Next Hop*)  
**else** transmite datagrama unui *ruter implicit*

Adresarea *ierarhică*  $\langle \text{adresa\_retei}, \text{adresa\_gazda} \rangle$  reduce  
numarul de intrări în tabela de dirijare (o intrare pentru o  
 $\text{adresa\_retei}$ )

În practică, tabelele de rutare sunt separate pe clase de adrese  
- Căutare prin: *indexare* (A și B) sau *hashing* (C)

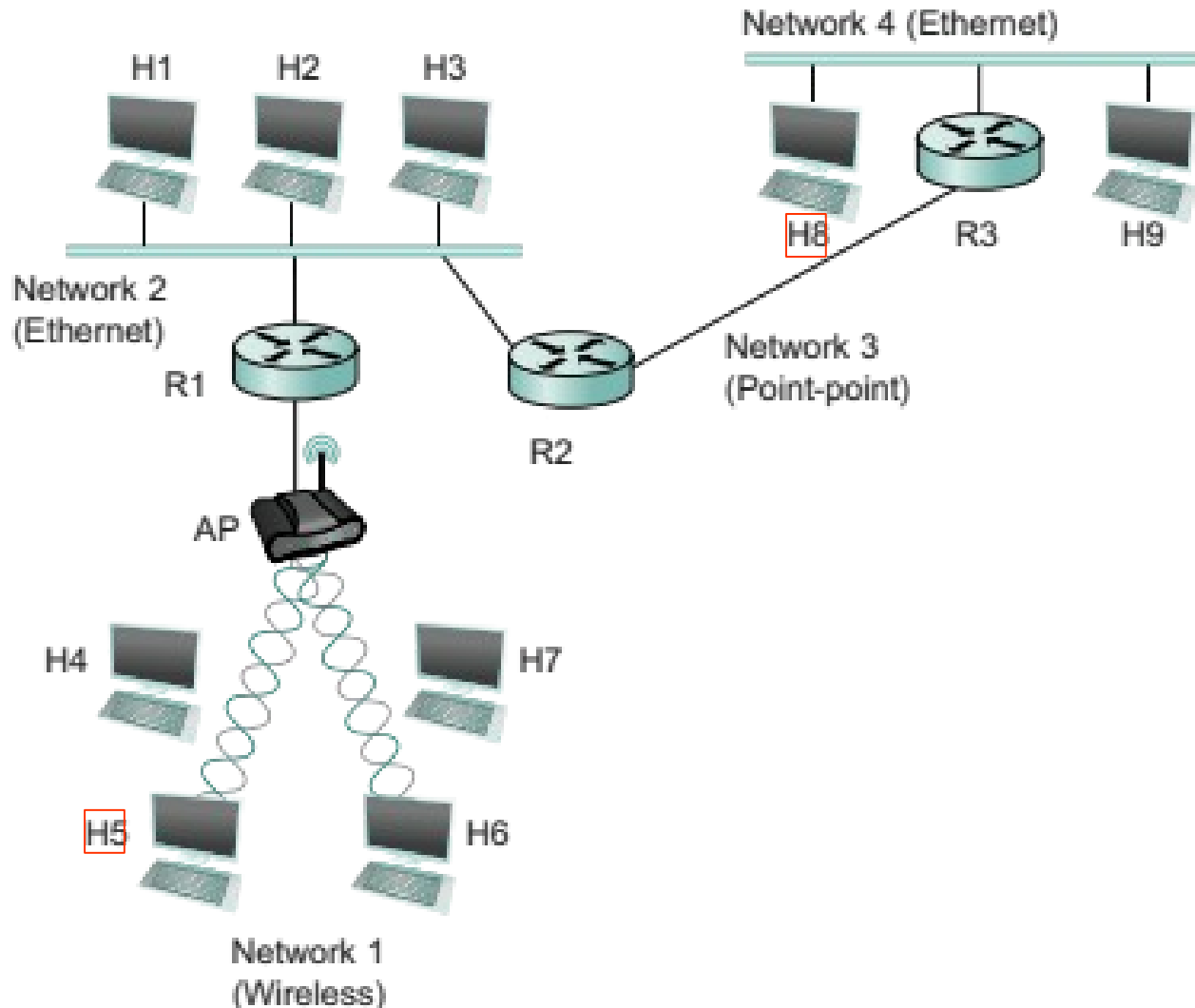
# Exemplu

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

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

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

**R3** este în aceeași rețea cu **H8** și îi livrează direct datagrama







# Subrețele

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



# CIDR – Classless InterDomain Routing

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

**Notăția specială** pentru **adresa de rețea CIDR**

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

**adresa\_retea** ocupă **21 biți**

**adresa\_gazda** ocupă **11 biți**

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



# CIDR – Exemplu

Pentru a afla [adresa\\_retea](#) se folosesc [măști](#)

**Cambridge:** Adresă **11000010 00011000 00000000 00000000**

Mască **11111111 11111111 11111000 00000000**

**Edinburgh:** Adresă **11000010 00011000 00001000 00000000**

Mască **11111111 11111111 11111100 00000000**

**Oxford:** Adresă **11000010 00011000 00010000 00000000**

Mască **11111111 11111111 11110000 00000000**

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

# CIDR – reguli de alocare a adreselor

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

Reguli pentru a avea o mască pentru un bloc de adrese

⇒ **lungimea blocului** trebuie să fie o **putere a lui 2**

⇒ toate adresele din bloc au **aceeași adresă\_retea** ⇒ adresa de început a blocului de adrese trebuie să fie **multiplu de dimensiunea acestuia**

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

**0**

**2048**

**3072**

**4096**

**Cambridge**

**Edin.**

**Oxford**

# Algoritm *forwarding*

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

Algoritmul alege intrarea pentru care

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

Ex. Sosește pachet cu adresa\_IP = 194.24.17.4

compara cu Cambridge /21 – adresa\_retea = 194.24.0.0

adresa\_retea: **11000010 00011000 00000000 00000000**

Masca: **11111111 11111111 11111000 00000000**

adresa\_IP: **11000010 00011000 00010001 00000100**

Adresa\_IP AND Masca:

**11000010 00011000 00010000 00000000**

= 194.24.16.0  $\neq$  194.24.0.0  $\rightarrow$  nepotrivre



## Algoritm *forwarding* (2)

Ex. Sosește pachet cu adresa\_IP = 194.24.17.4

Cambridge /21 – adresa\_rețea = 194.24.0.0

(Adresa\_IP AND Masca) = 194.24.16.0 ⇒ nepotrivire

Edinburgh /22 - adresa\_rețea = 194.24.8.0

(Adresa\_IP AND Masca) = 194.24.16.0 ⇒ nepotrivire

Oxford /20 - adresa\_rețea = 194.24.16.0

(Adresa\_IP AND Masca) = 194.24.16.0 ⇒ potrivire

Dacă nu sunt alte potriviri ⇒ folosește intrarea pentru Oxford



# Potriviri multiple

Prefixe de lungimi diferite

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

Ex.

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

**Regula:** se alege potrivirea “mai lungă”  
(“Longest prefix match”)



# Reducere dimensiune tabelă rutare

Soluție - agregarea unor adrese

Consideram adresele de rețea:

C - Cambridge: 194.24.0.0/21

E - Edingurgh: 194.24.8.0/22

O - Oxford: 194.24.16.0/20

C: adresa\_retea **11000010 00011000 00000000 00000000**

E: adresa\_retea **11000010 00011000 00001000 00000000**

O: adresa\_retea **11000010 00011000 00010000 00000000**

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

Tabela de dirijare poate include o singură intrare comună pentru

adresa\_retea **11000010 00011000 00000000 00000000**

adica **194.24.0.0/19**





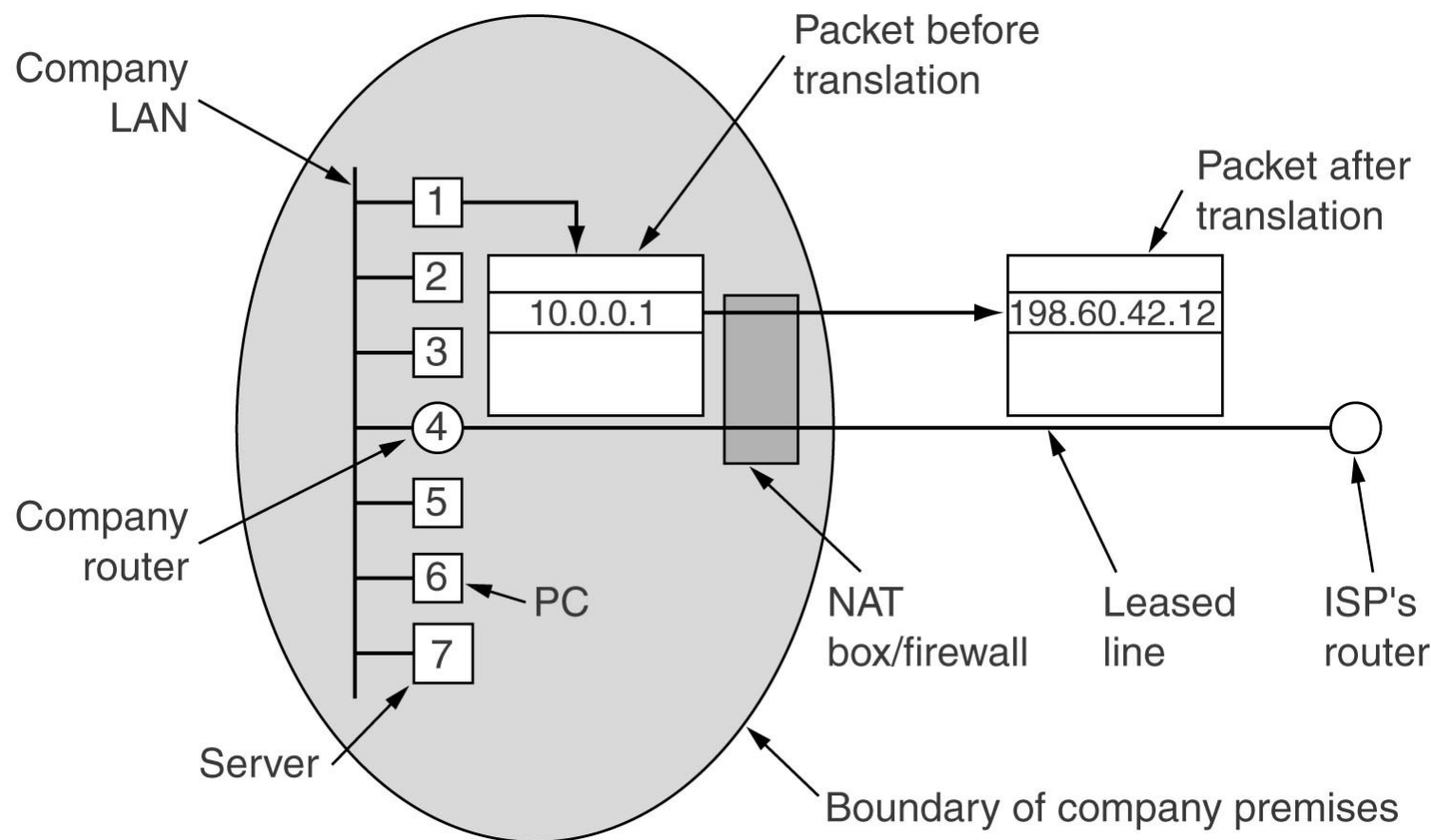
# NAT

# NAT – Network Address Translation

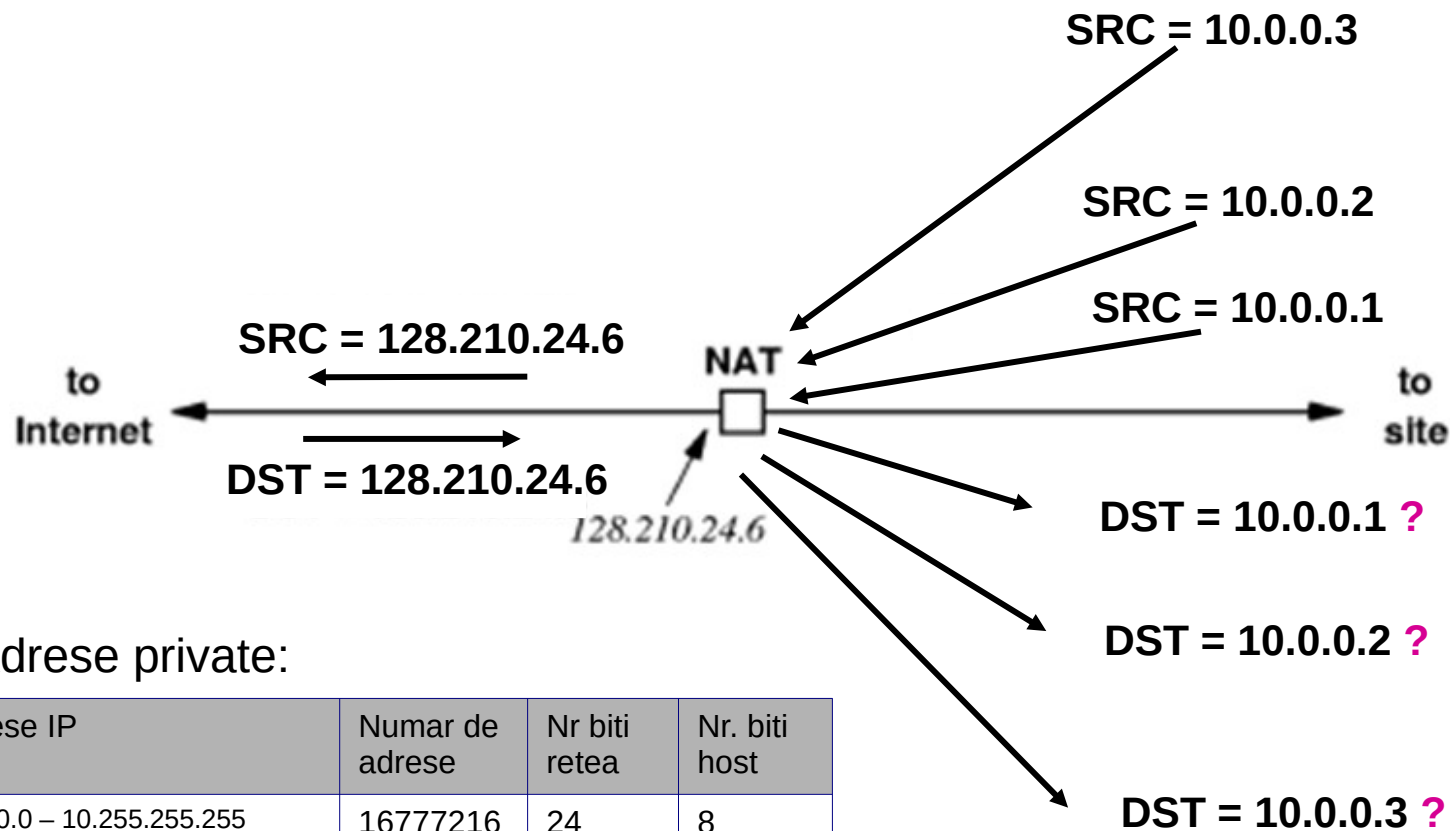
O adresă este asignată pentru mai multe calculatoare

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

NAT translatează între adresa privată și o adresă globală



# Translatarea adresa globală $\Rightarrow$ adresa privată



Adrese private:

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

# Principiul NAT

Folosește

adresa IP + număr port transmitator  
tabela de traducere

Transmisie

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



antet  
TCP

antet  
IP

tabela de traducere

IP-local	IP-glob	port
10.0.0.1	128.210.24.6	1200

20





## Receptie

obține număr port din pachet (= index în tabela de traducere)

extrage adresa IP locală și număr port

înlocuiește adresa IP și număr port din pachet

re-calculează sumele de control IP și TCP



antet  
TCP

antet  
IP

IP-loc	IP-glob	port
10.0.0.1	128.210.24.6	1200

20





# IPv6



# IPv6 - Motivații

## Spațiul de adrese

IPv4 - 32 biți = peste un milion de rețele

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

## Tip servicii

Aplicații diferite au cerințe diferite de livrare, siguranță și viteză

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

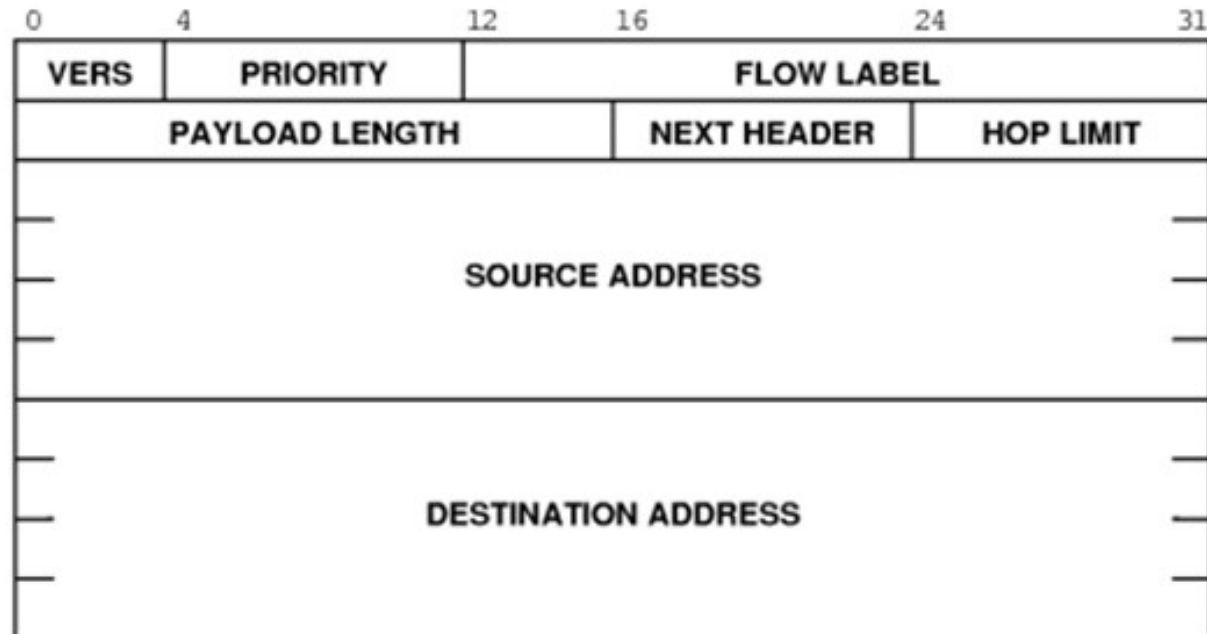
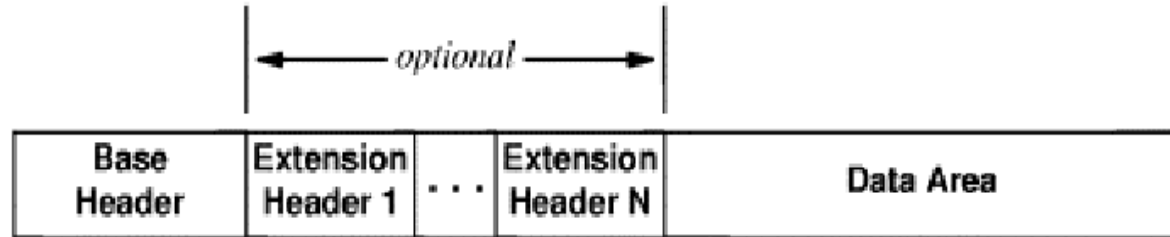
# IPv6 - format Base header

lungime fixă = 40 octeti

Priority - clasa de trafic

**FLOW LABEL** - asociază datagramele unui flux

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





## Conține mai puține info decât antet IPv4

Restul de info în extensii

NEXT HEADER definește tipul datelor (ex. TCP)

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



(a)



(b)



## IPv6 – antete extensie

**Hop-by-hop header** – info pentru rutere – deocamdata:  
suport datagrame excedând 64K (jumbograme)  
specifica lungimea;  
campul de lungime din antetul de baza este 0

**Destination header** – info aditionale pentru destinație  
nefolosit

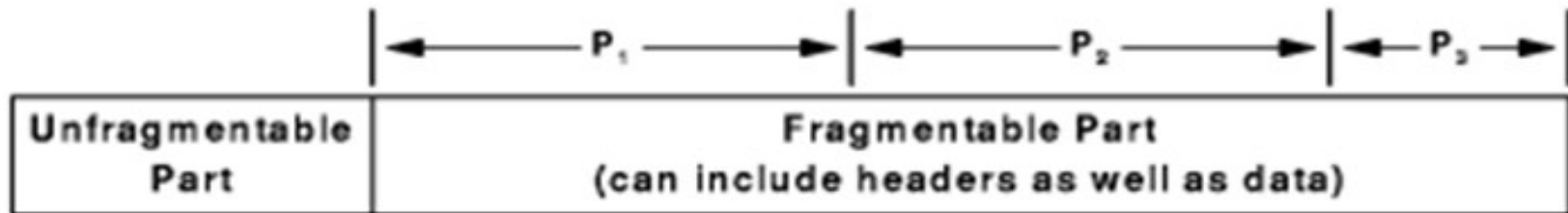
**Routing** – lista rutere de vizitat

**Fragmentation** – identificare fragmente

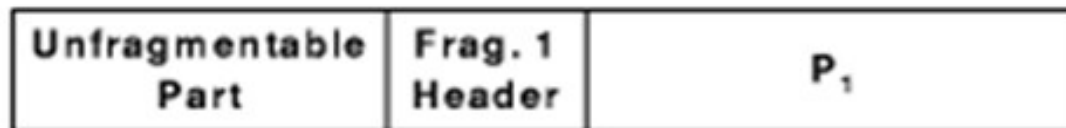
**Authentication** – verificare identitate transmițător

**Encrypted security payload** – info despre conținut criptat

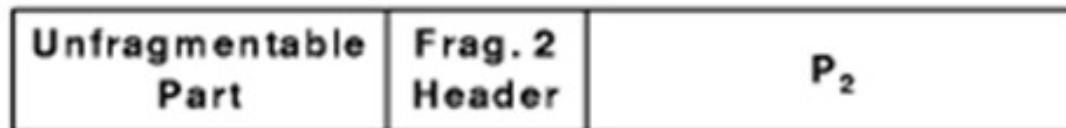
# Fragmentarea



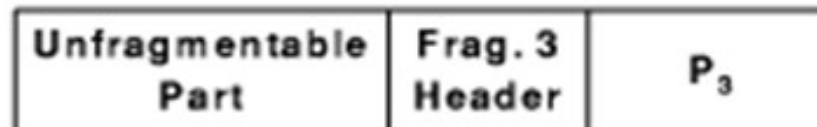
(a)



(b)



(c)



(d)



## 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 de 128 biti

Includ prefix rețea și suffix gazdă

Fără clase de adresă – limita prefix/suffix oriunde

### Tipuri speciale de adrese:

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



## Notăția adresei

- 16 numere  
105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255
- Notăție hexazecimală
- 69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF
- Compresie zerouri
- FF0C:0:0:0:0:0:0:B1
- FF0C::B1
- Adrese IPv6 cu 96 zerouri prefix sunt interpretate ca adrese IPv4

# IPv6 adoption rate

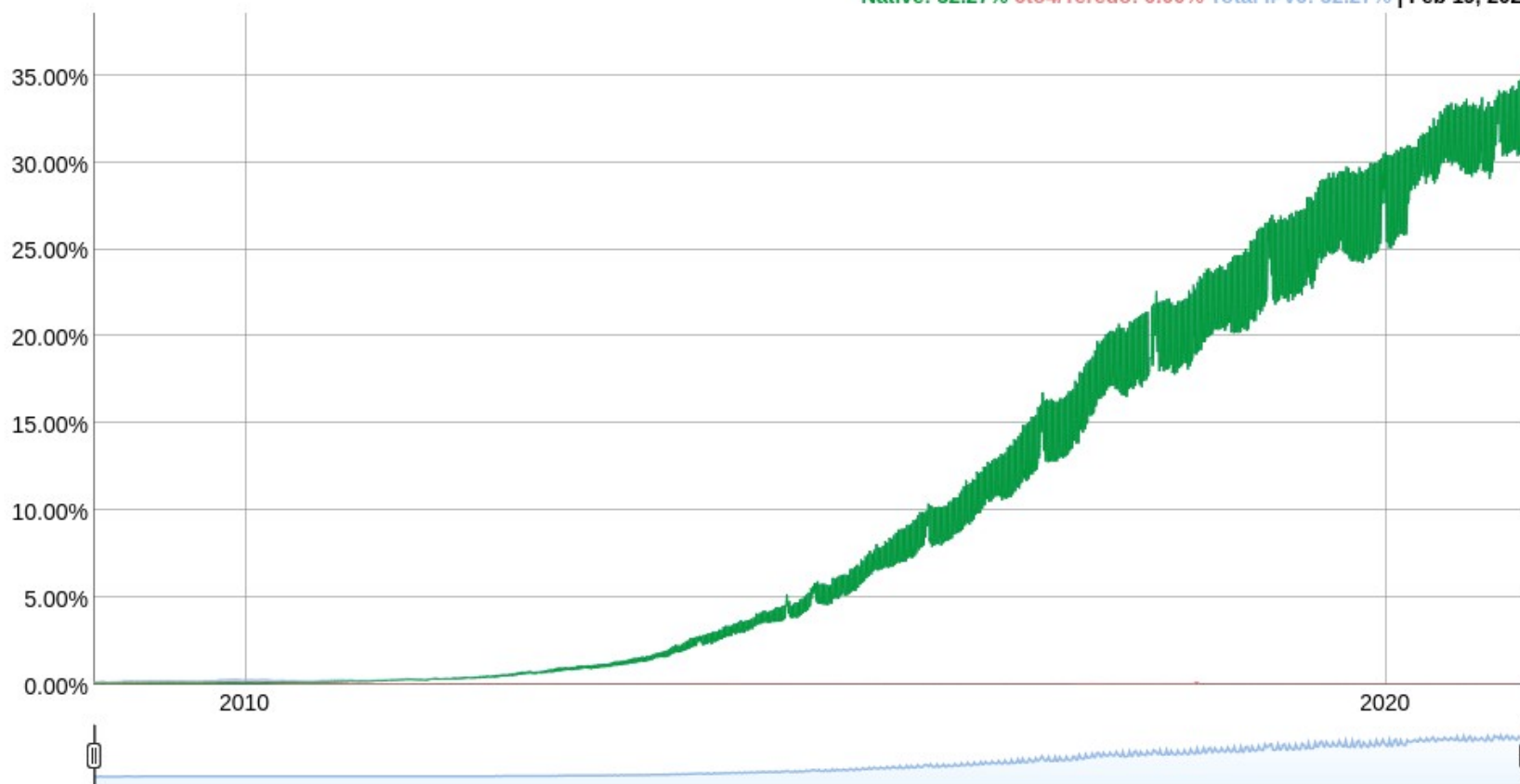
## IPv6 Adoption

### Per-Country IPv6 adoption

## IPv6 Adoption

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

Native: 32.27% 6to4/Teredo: 0.00% Total IPv6: 32.27% | Feb 19, 2021

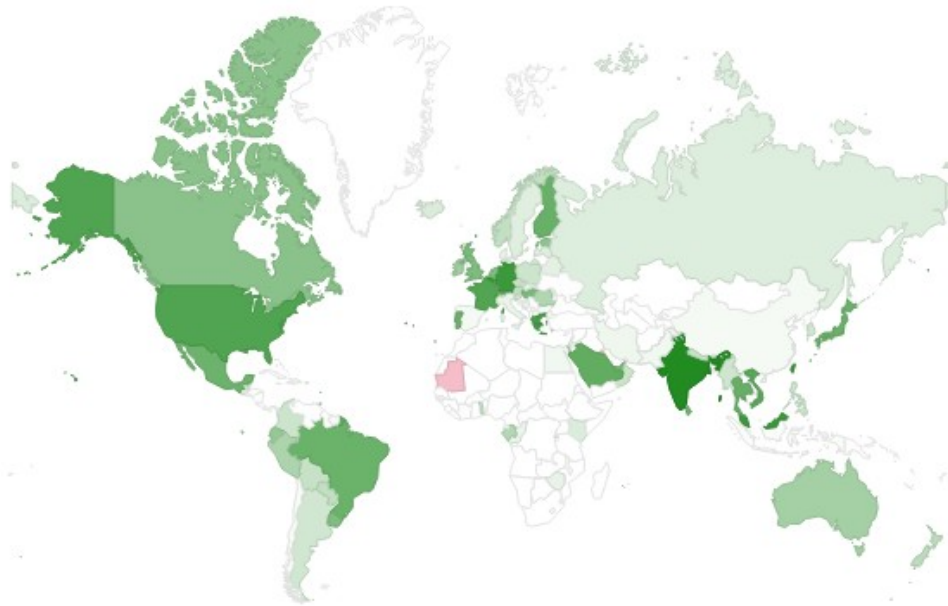


# IPv6 adoption rate




IPv6 Adoption

Per-Country IPv6 adoption

Per-Country IPv6 adoption

[World](#) | [Africa](#) | [Asia](#) | [Europe](#) | [Oceania](#) | [North America](#) | [Central America](#) | [Caribbean](#) | [South America](#)

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

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





# Dirijarea



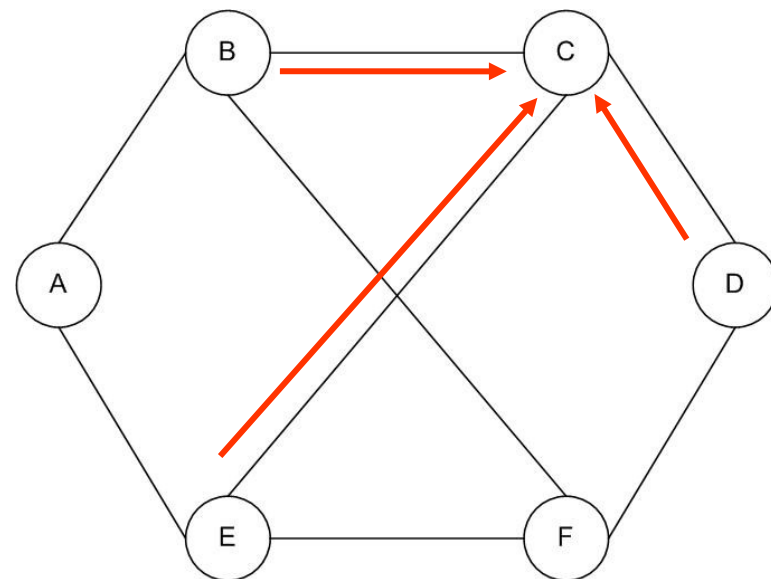
# Dirijarea - clasificare

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

# Vectorii distanțelor (Distance vector)

## Algoritm distribuit !

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



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

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

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

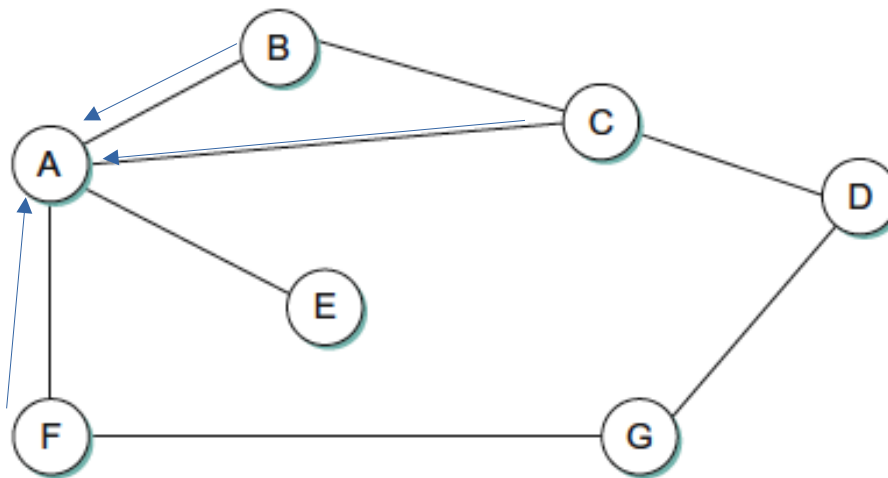
De la E: (7, 6, 3, 9, 0, 4).

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

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

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

# Problema numărării la infinit



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

La E:      1   2   2   3   0   2   3      distante initiale la E

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

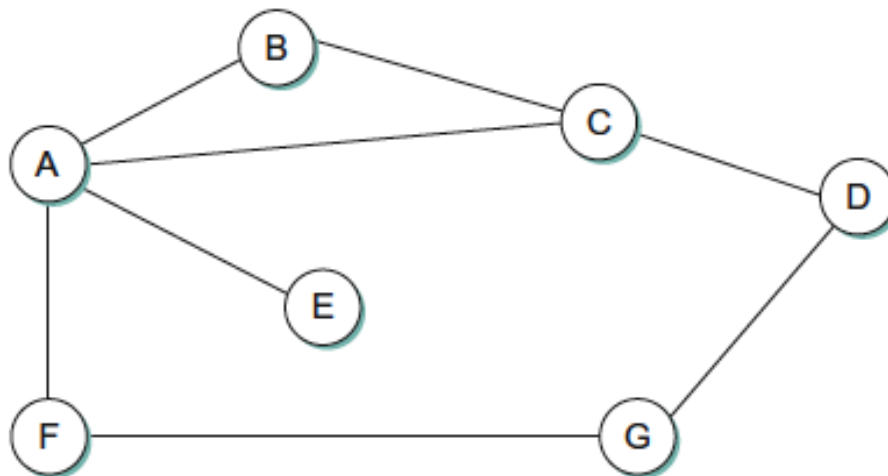
distantele catre E anuntate de: **A =  $\infty$ , B = 2, C = 2**

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

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

B anunta dist. 3 lui A

## Problema numărării la infinit (2)



De la A B C D E F G

4 3 2 3 0 2 3

A alege ruta prin B dist=4

A anunța dist. 4 lui C

4 3 5 3 0 2 3

C recalculează dist=5

distanțele cresc teoretic la infinit

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

## Problema numărării la infinit - solutii (3)

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

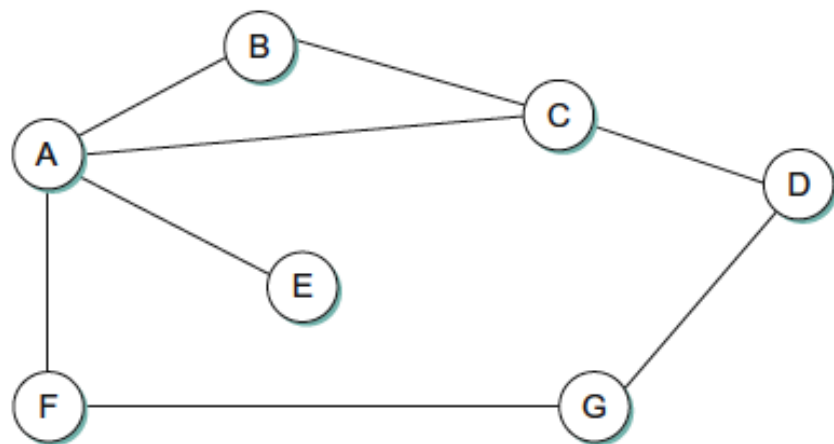
- B are ruta de distanta 2 catre E prin A
- B nu include noua distanta catre E in actualizarea trimisa lui A

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

- B trimite distanta  $\infty$  catre A
- A **nu va mai alege** o cale prin B

**Neajuns**: solutiile nu functioneaza in toate cazurile

- de ex. pentru bucle cu mai multe noduri

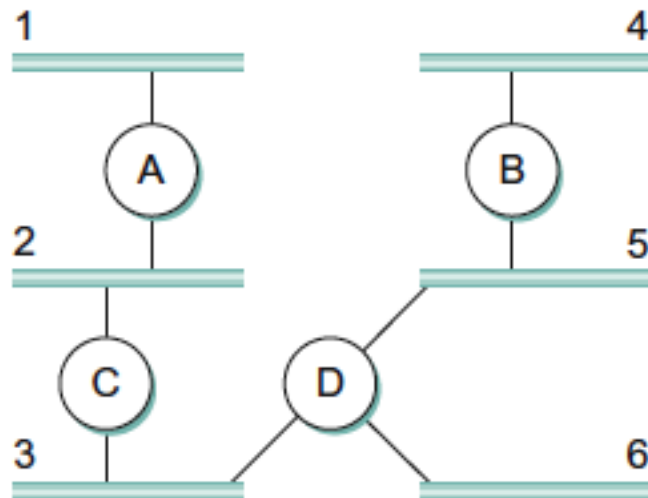


# RIP - Routing Information Protocol

Specificatia RIP v2 in RFC 2453

<https://datatracker.ietf.org/doc/html/rfc2453>

- Fiecare legatura are cost 1
- Foloseste **distanțe** la rețele (nu la noduri)
  - ruterul C are distanta 0 la rețeaua 2 si 2 la rețeaua 4
- Transmit vectorii distantelor la fiecare 30 secunde
- Distanțe maxime de 15 hop-uri (16 inseamna infinit)
  - rețele de mici dimensiuni





# RIP - Routing Information Protocol

- Command = pachetul este cerere sau raspuns

```

      0              1              2              3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|  command (1)  |  version (1)  |          must be zero (2)          |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|
|          Route Table Entry 1 (20)
|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|
|          ...
|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|
|          Route Table Entry N (20)
|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

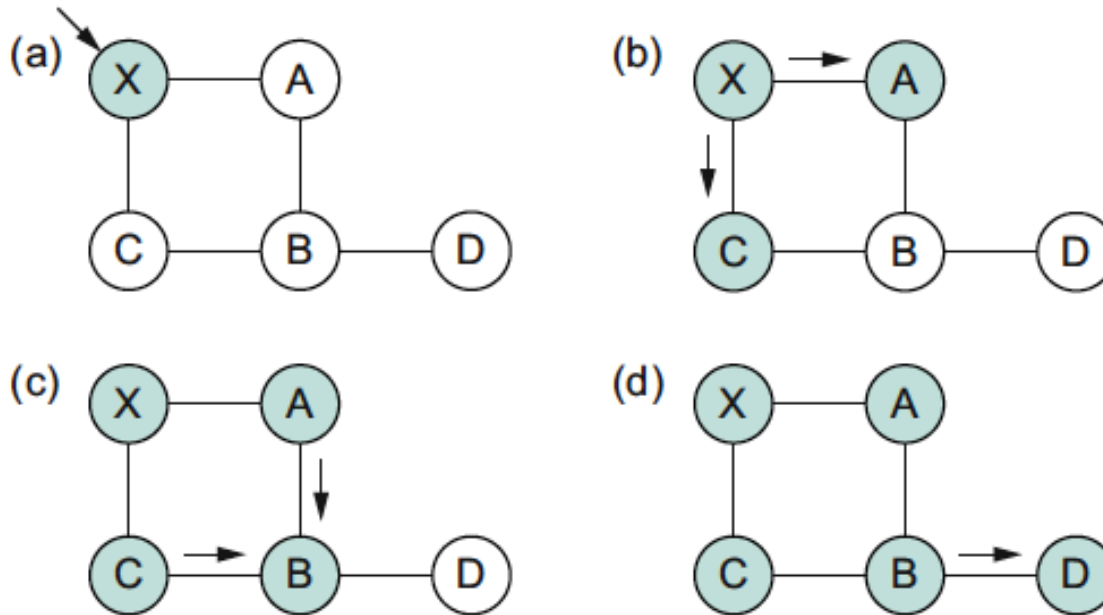
```



# Starea legaturii (Link State)

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

# Transmiterea prin inundare



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



# Algoritmul căii celei mai scurte

## Algoritmul lui Dijkstra

nnod	numărul nodurilor rețelei;
sursa	nodul sursă;
$l[i][j]$	costul legăturii (i,j), având valorile 0 dacă $i = j$ ; <b>lungmax</b> dacă i și j nu sunt adiacente; o valoare între 0 și <b>lungmax</b> în celelalte cazuri;
$D[i]$	costul minim al legăturii de la sursă la i;
S	mulțimea nodurilor deja selectate;
<b><math>V[i]</math></b>	<b>tabela de dirijare;</b> <b><math>V[i]</math> = vecinul prin care se transmit date de la nodul curent la nodul i.</b>



```

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

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

```



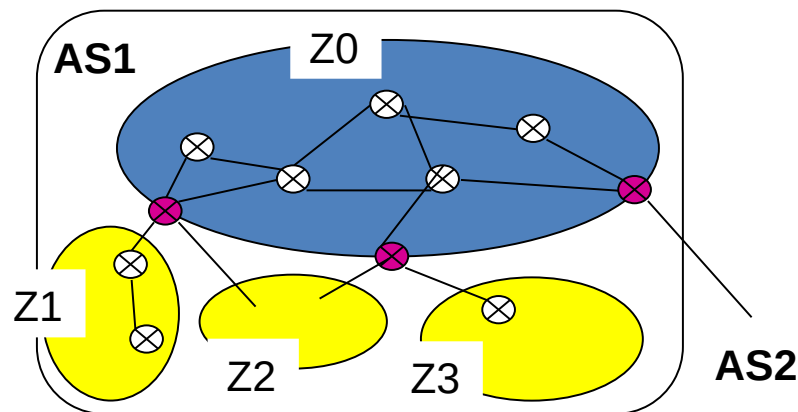
# Structura ierarhica a Internet-ului

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

# Structura ierarhica AS

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

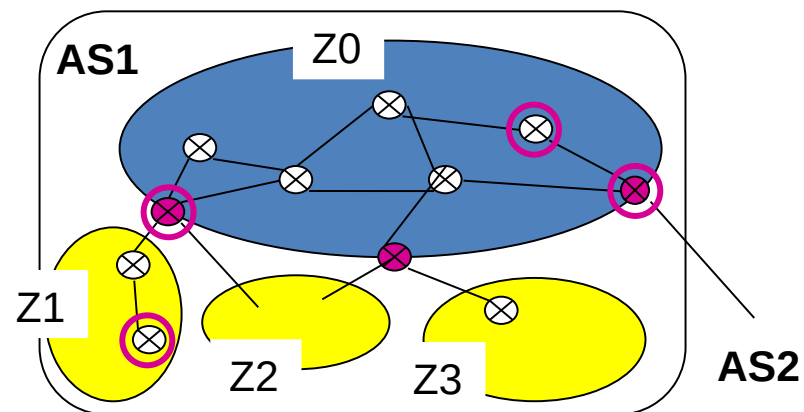
- zona **Z0** – coloana vertebrală (**backbone**)
  - zone “**stub**” Z1, Z2 ... – legate la **backbone**
  - **rutele** intre noduri din **zone stub diferite** trec prin zona Z0
- Ierarhizarea crește **scalabilitatea**
    - un ruter dintr-un AS nu trebuie sa stie cum se ajunge la fiecare **retea** din AS, fiind suficient sa stie cum se ajunge la **zona** in care se afla retea respectiva - > **reduce volumul tabelului de dirijare**



## Structura ierarhica AS (2)

Tipuri de rutere

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







# OSPF – Open Shortest Path First

Mesajele OSPF permit schimbul de informatii intre noduri  
Sunt transmise in pachete IP cu 89 ca numar de protocol

**Hello** – stabileste si pastreaza legaturi cu vecinii  
descopera nodurile (rutere) cu care este conectat direct

**Link state request** - Cerere stare legătură  
cere info despre anumite legături de la un alt ruter

**Link state update** - Actualizare stare legătură  
trimite info despre legaturi, ca raspuns la o cerere

**Link state ack** - Confirmare stare legătură  
confirmă primirea unui mesaj de actualizare

**Database description** – trimite LSDB – Link State Data Base  
mesajele contin info despre AS sau zonă

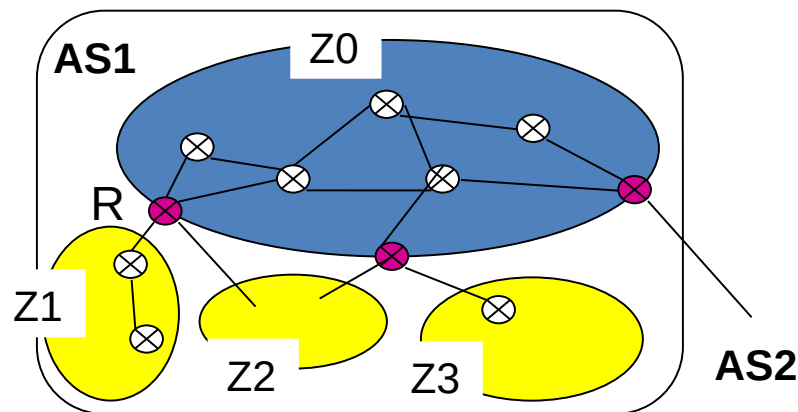
## Calcul rute - Nivel 1 (zona stub)

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

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

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

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



## Calcul rute - Nivel 2 (AS)

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

- de la nodul sursa la zona backbone
- traverseaza backbone
- de la backbone la retea de destinatie

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

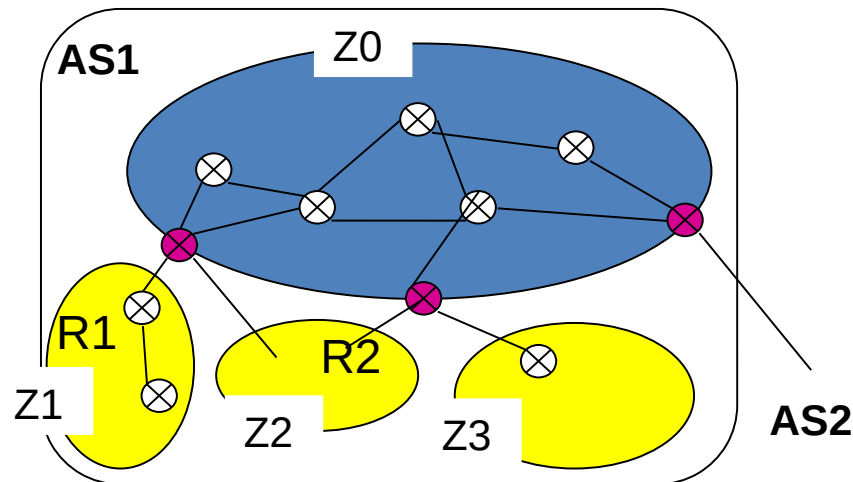
ex. **R1** ofera caile cele mai scurte pentru **Z1** si **Z2**

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

Rezultatele sunt **difuzate de la Z0**

inapoi la **zonele stub**, care actualizeaza caile cele mai scurte la retele din alte zone.

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



# BGP – Border Gateway Protocol

Algoritmi orientați pe aspectele politice, de securitate, economice  
Descris în RFC 4271

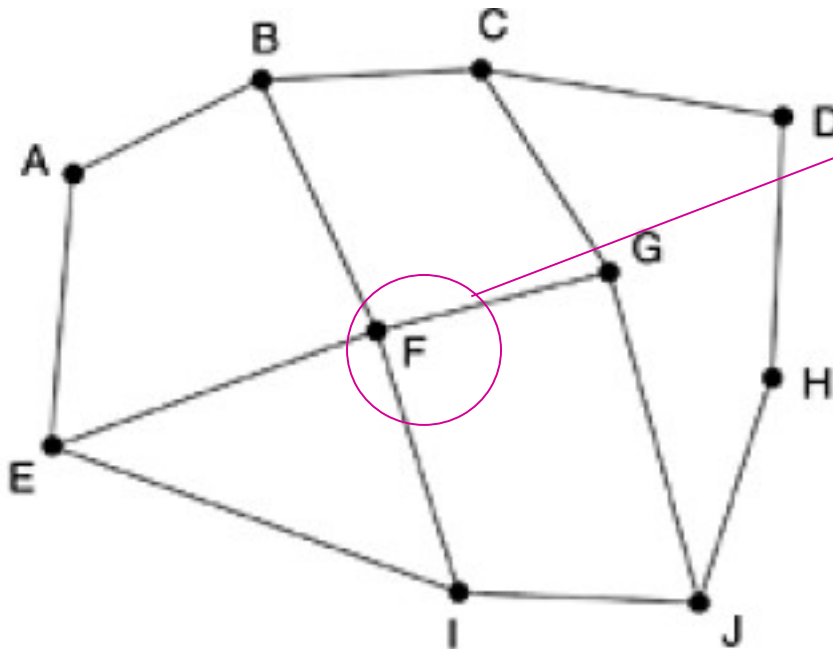
Rețea = nodurile reprezintă AS-uri, rutarea se face doar inter-domain

Protocol = vectorul distanțelor

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

Comunică vecinilor căile utilizate efectiv (evita problema numărării la infinit)

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



G se defectează  
Informațiile primite de F  
de la vecinii rămași, despre D:

De la B: “Eu folosesc BCD”

De la I: “Eu folosesc IFGCD”

De la E: “Eu folosesc EFGCD”

**F elimină caile care conțin F  
și alege calea FBCD**



# BGP – Border Gateway Protocol

## Tipuri de mesaje folosite de BGP :

- **OPEN** : mesaj care initializeaza sesiunea BGP si negociaza optiunile posibile.
- **NOTIFICATION** : mesaj folosit pentru a incheia o sesiune BGP, de obicei datorita unei erori. Router-ul se va inchide imediat dupa trimiterea sau primirea unui mesaj de tip NOTIFICATION.
- **UPDATE**: mesaj pentru trimiterea de rute noi sau modificari ale rutelor existente.
- **KEEPALIVE** : Mesaj pentru a confirma daca vecinii unui router sunt functionali. Daca un router trimite un mesaj UPDATE timp de 30 de secunde, atunci va trimite un mesaj KEEPALIVE. Daca nu se primeste de la un router nici un fel de mesaj timp de 90 de secunde, router-ul respectiv se va considera defect/inchis si toate rutele prin el vor fi retrase.



# Curs - Test 1

- Data: 13.04.2021, ora 10:00(321CD & 322CD),
  - ora 10:30(323CD & 324CD)
- Durata: 20 min
- 10 intrebari tip grila (un singur raspuns corect)
- Materie necesara – pana la nivelul retea (inclusiv)
- Total 0.5p

## Exemplu subiect:

*1. Ce se foloseste pentru a marca un pachet IPv4 care nu se poate fragmenta?*

- a) flag-ul DF*
- b) flag-ul MF*
- c) campul offset*
- d) toate pachetele IPv4 se pot fragmenta*