Nivelul Rețea
- Organizare internă, adresare și dirijare -

Cuprins

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

De ce este nevoie de pachete?

Legăturile unei rețele pot fi folosite simultan de transmisii paralele între mai multe perechi de noduri

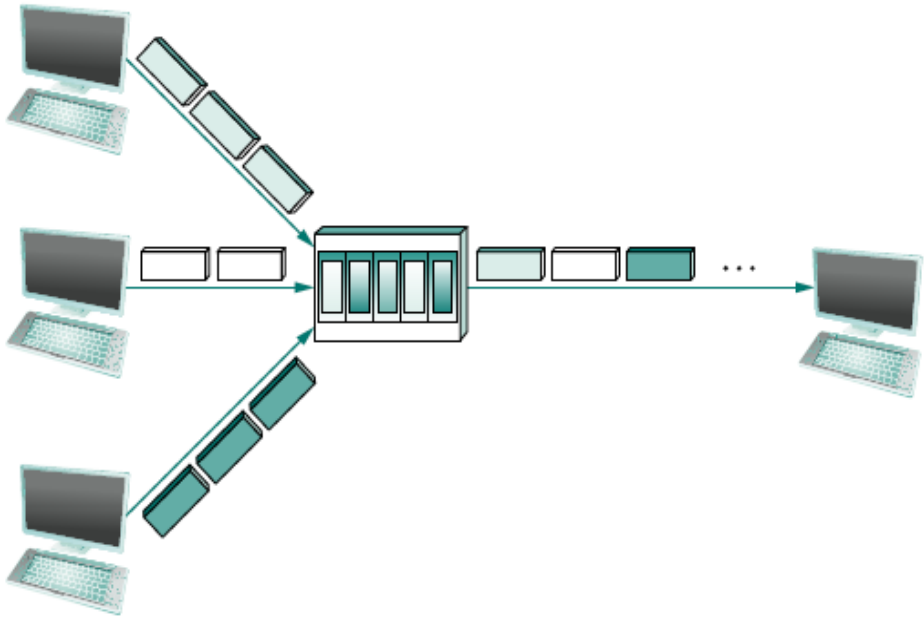
Multiplexare – se alocă transmisiilor

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

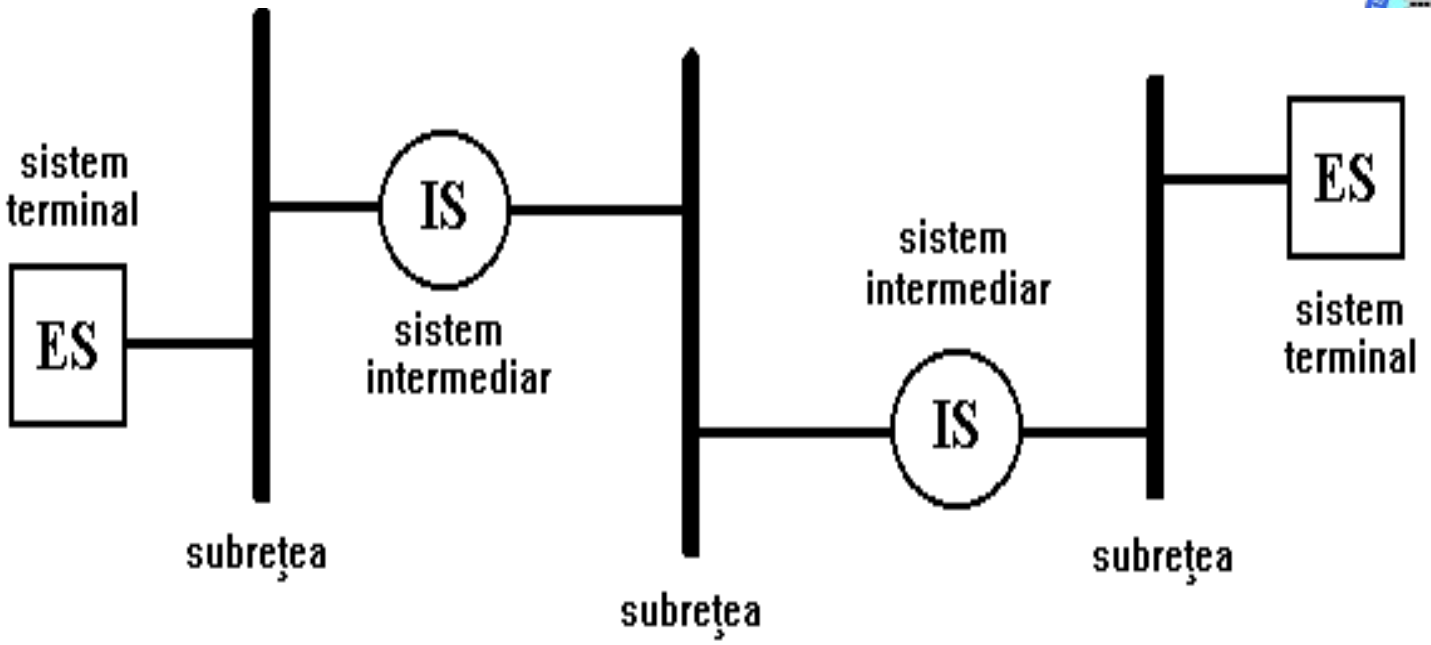
- multiplexare statistică – STMD – sloturile sunt alocate la cerere;

  dacă o singură pereche de noduri are de transmis, nu așteaptă slot;

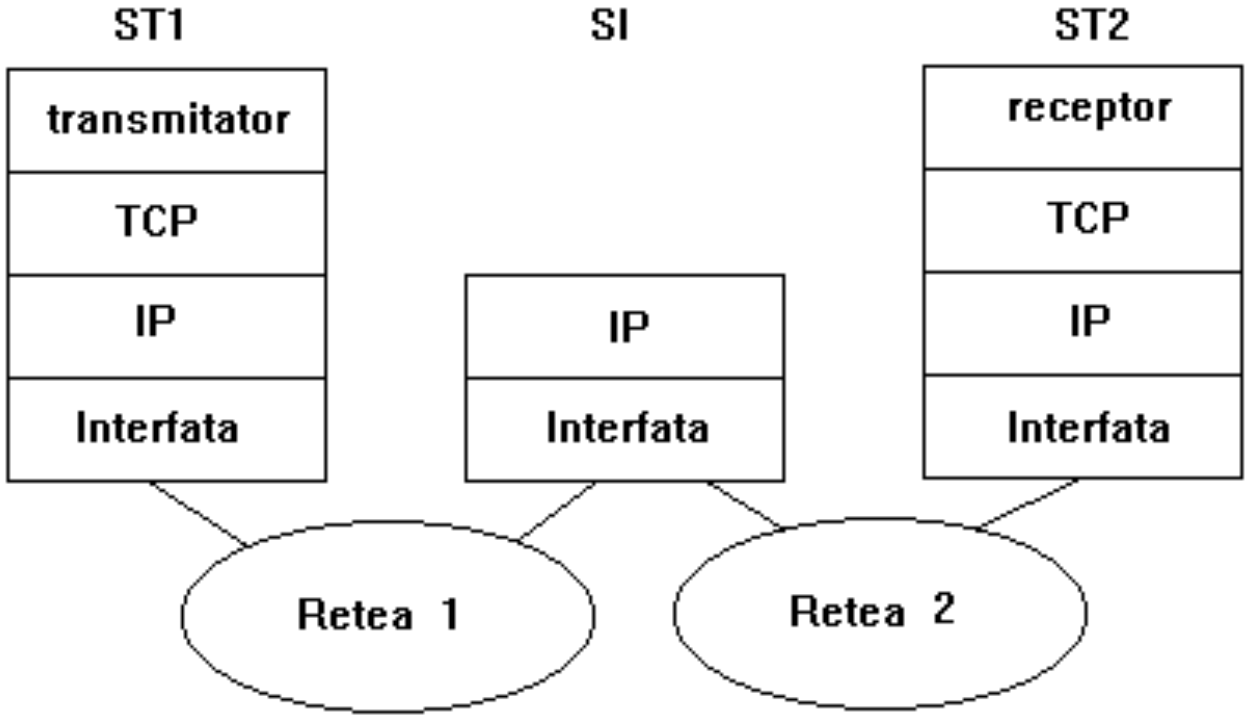
- pentru a evita acapararea legăturii, transmisia se face în pachete cu dimensiune limitată



Modelul Internetului



Nivelele străbătute de pachete



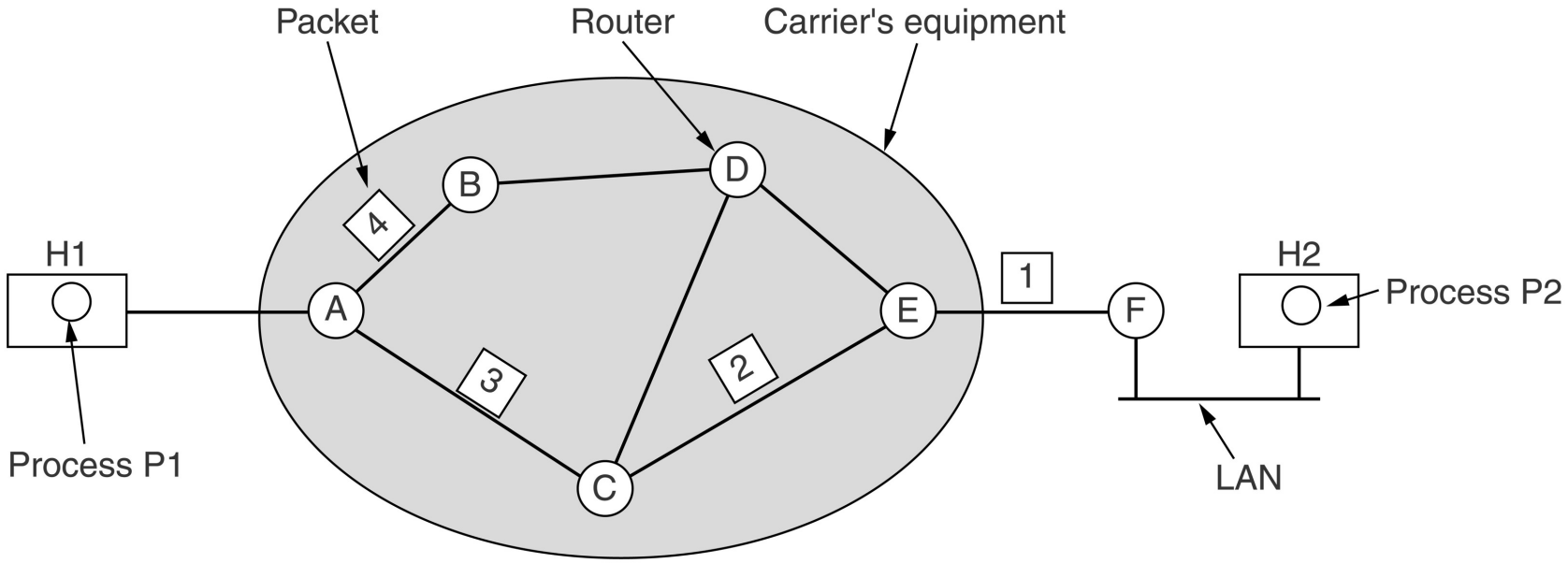
Funcţiile nivelului reţea

- dirijarea pachetelor
- adresarea

Aspecte principale

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

Organizarea internă - datagrame



A's table

Folosita de pachetele 1, 2 si 3

C's table

A	A
B	A
C	–
D	D
E	E
F	E

E's table

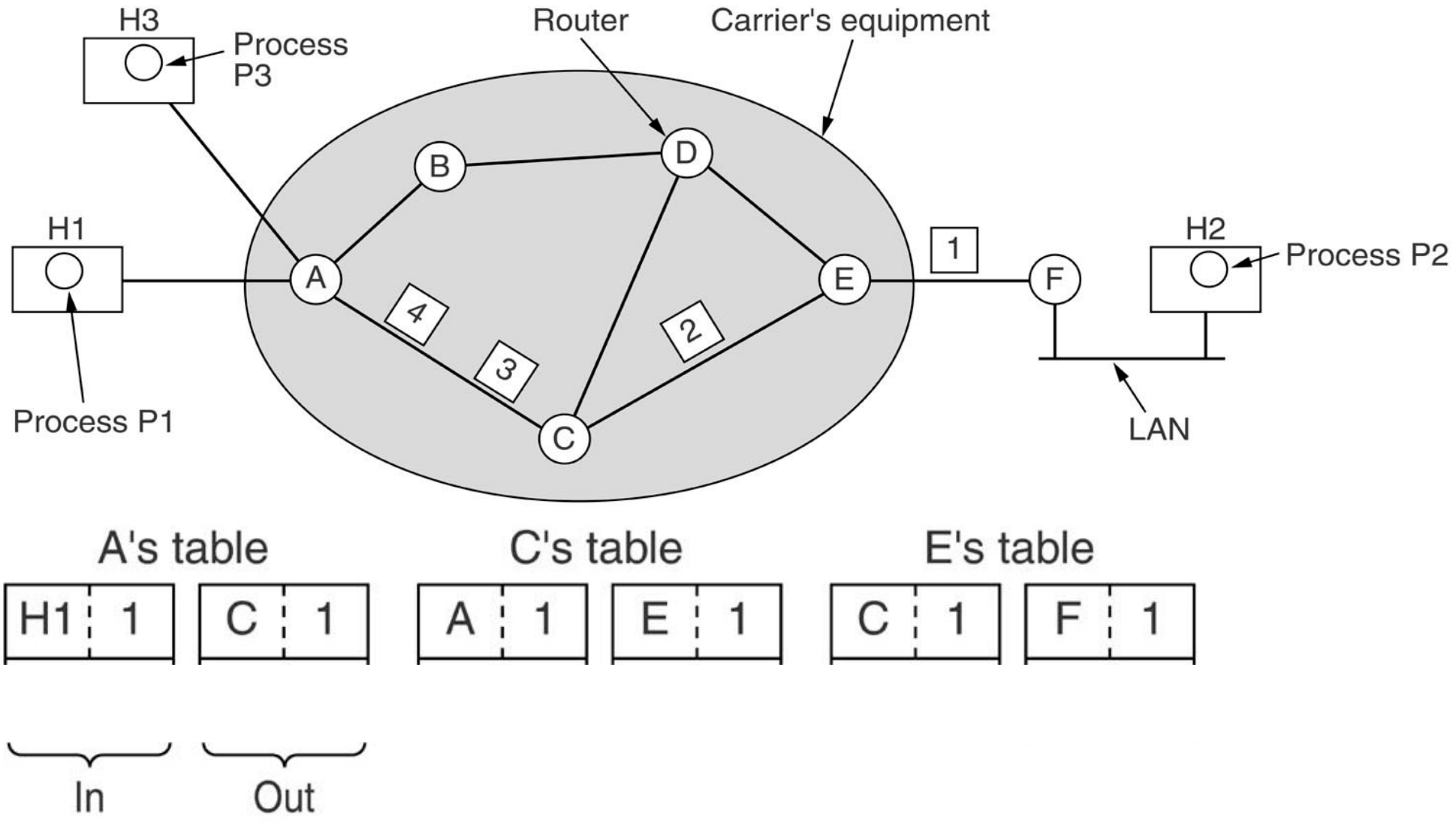
A	C
B	D
C	C
D	D
E	–
F	F

Folosita de pachetul 4

Caracteristici datagrame

- Pachetul conține toate informațiile necesare ruterelor pentru “pasarea” lui către destinație (inclusiv adresele sursă și destinație)
- Organizare fără conexiune
  - Un calculator gazdă (*host*) poate trimite pachetul oricând și oriunde în rețea
- Nu asigură corectitudinea
  - Transmițătorul nu poate ști dacă pachetul este livrat sau dacă destinatarul mai este conectat
- Nu păstrează ordinea pachetelor
  - Pachetele sunt dirijate independent unele de altele
- Robustă
  - La defectarea unei legături se găsesc rute alternative
- Utilizare largă în Internet

Organizarea internă – circuit virtual



Nodul A re-numerotează circuitul virtual

Caracteristici circuit virtual

- Organizare bazată pe conexiune
  - Transferul începe după stabilirea conexiunii
  - Pachetul conține id-ul conexiunii
  - Se pot aloca resurse la stabilirea conexiunii (memorie tampon pentru pachete)
- Transmițătorul știe
  - că există o conexiune
  - că receptorul este pregătit să primească pachete
- Se păstrează ordinea pachetelor
- Se poate controla fluxul

- Folosit în
  - MPLS (MultiProtocol Label Switching)
  - rețele virtuale private (VPN – Virtual Private Network)

Protocolul IP

- Are
  - o schemă de adresare care permite identificarea oricărui calculator din Internet
  - un model - datagramă - 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

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

PROTOCOL = (TCP, UDP, etc.)

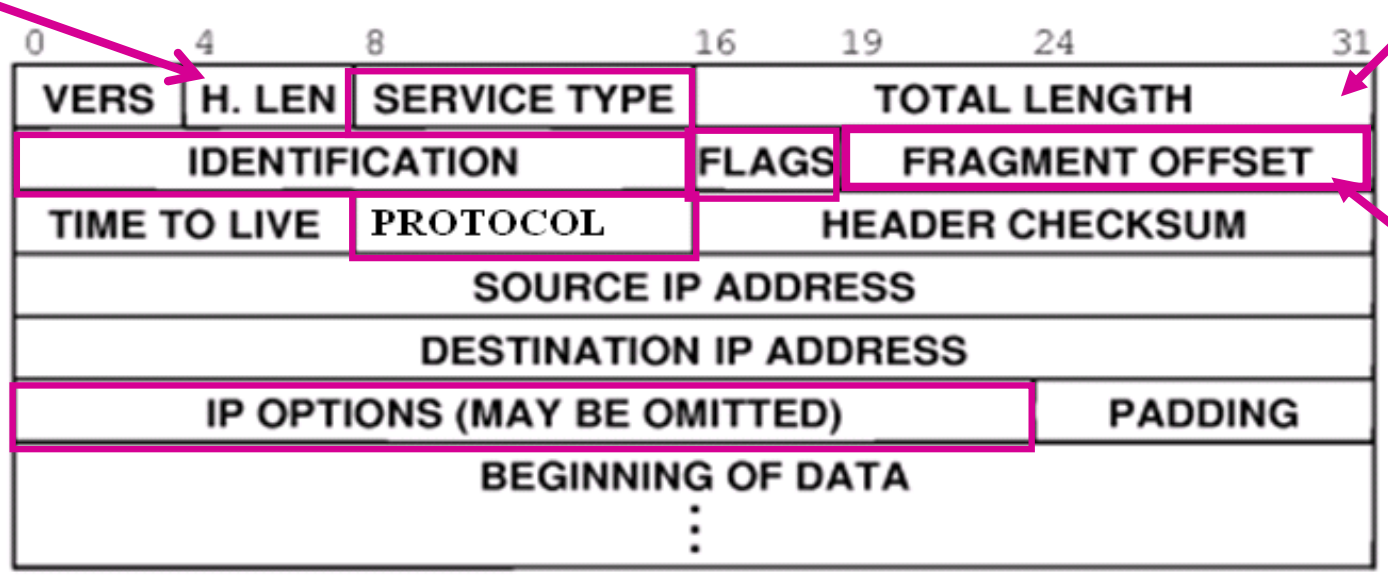
IDENTIFICATION Id datagrama de care aparţine fragmentul

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

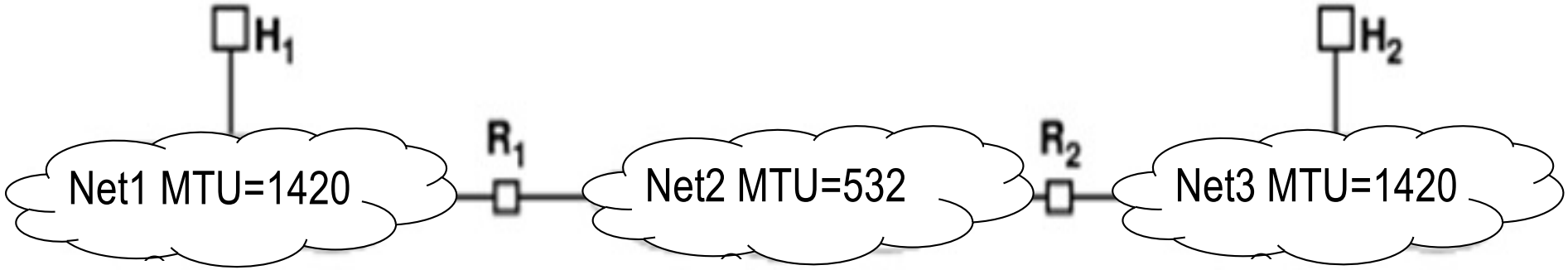
numar cuvinte de 32 biti



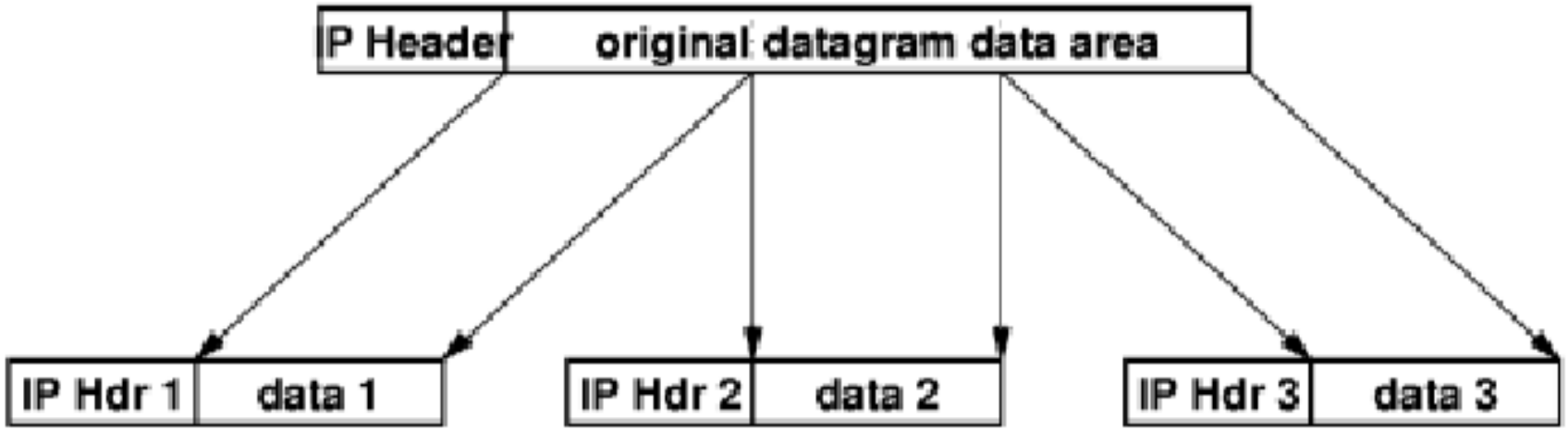
max 65535 octeti

max 8192 fragmente a 8 octeti

MTU – Maximum Transmission Unit



Fragmentarea se face la R_1



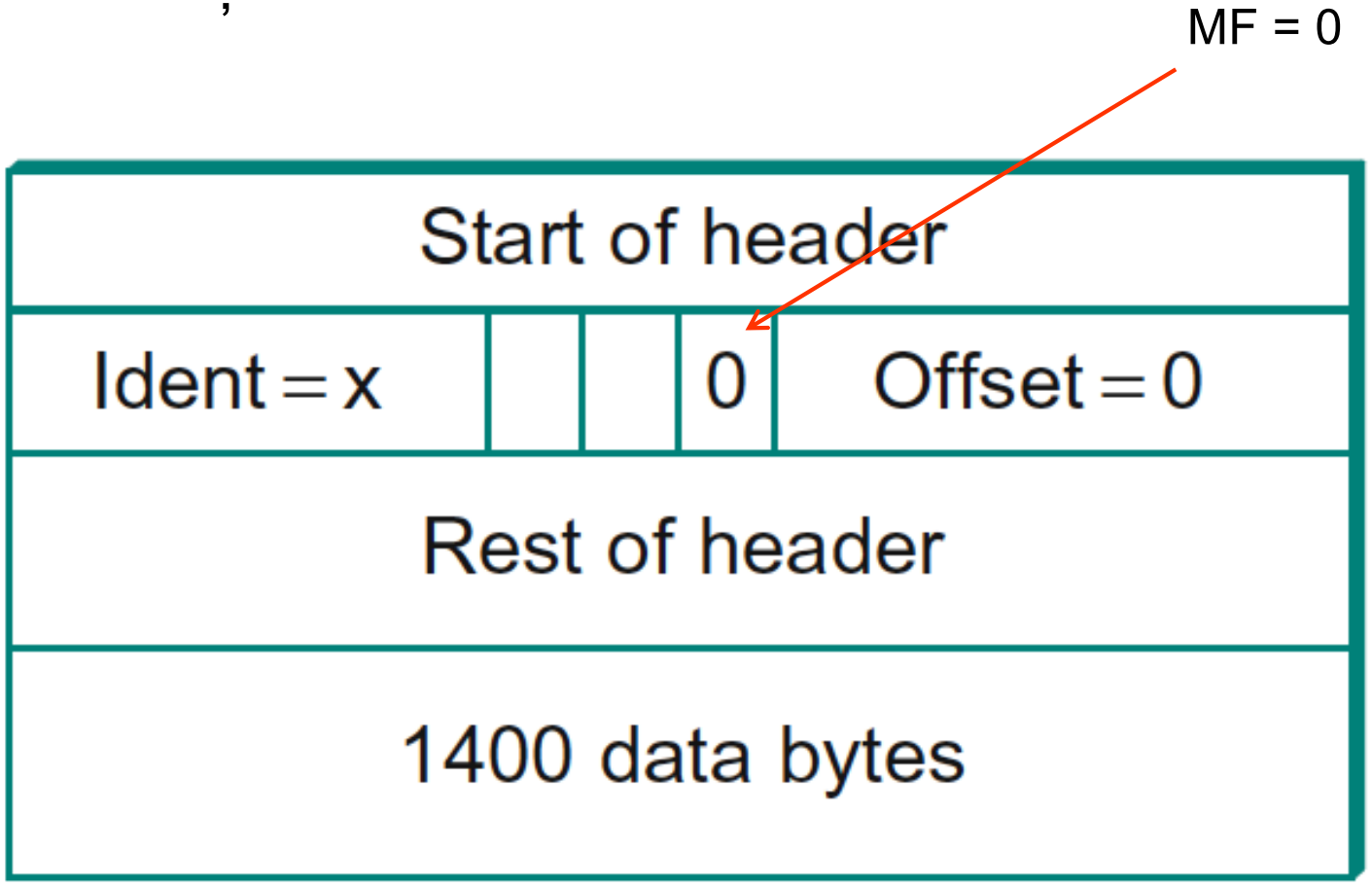
Reasamblarea se face la H_2 – de ce?

Câmpurile de antet pentru fragmentare (1)

(a) pachet transmis ne-fragmentat prin Net1 cu MTU = 1420

are: 1400 octeți de date

20 octeți de antet IP



Câmpurile de antet pentru fragmentare (2)

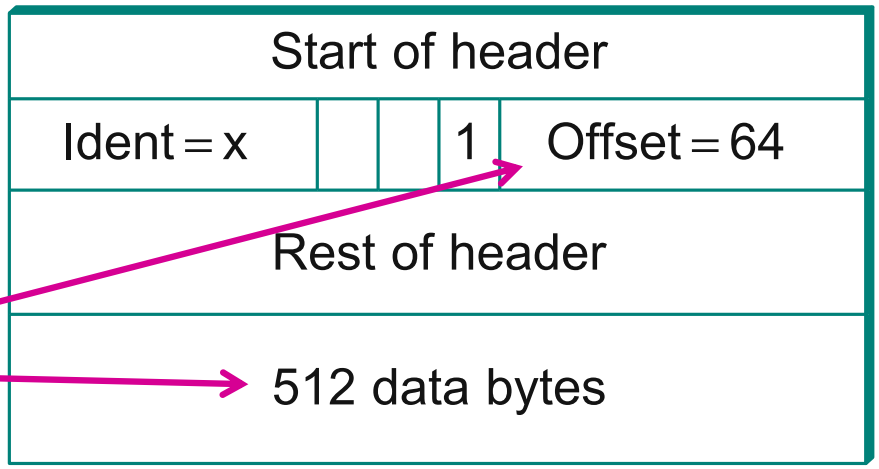
Net2 are MTU = 532

Ruterul R1 segmentează pachetul în 3 fragmente (fig: b1-b3)

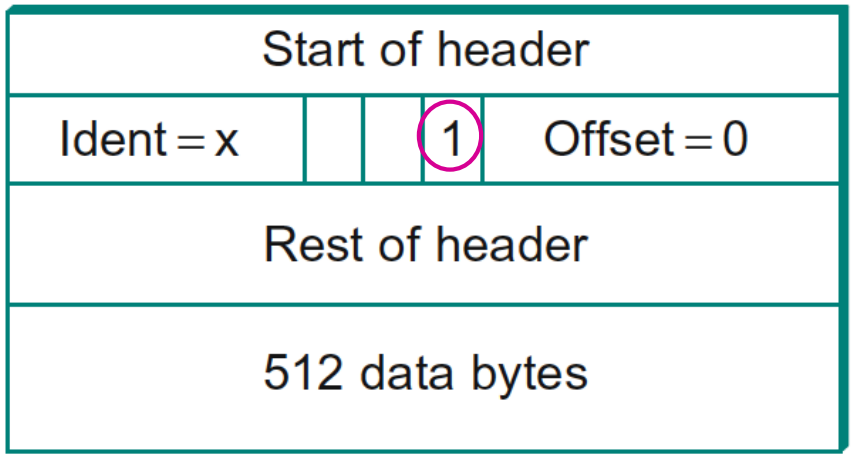
- lungimea fiecărui segment este multiplu de 8 octeți
- offset-ul numără grupuri de câte 8 octeți
- Offset = 64 = 512 / 8

unde e restul până la 532 octeți?

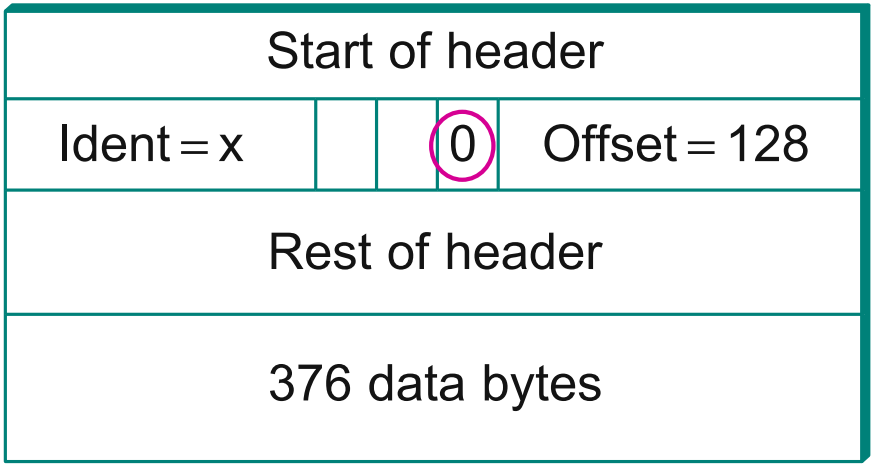
b2



b1

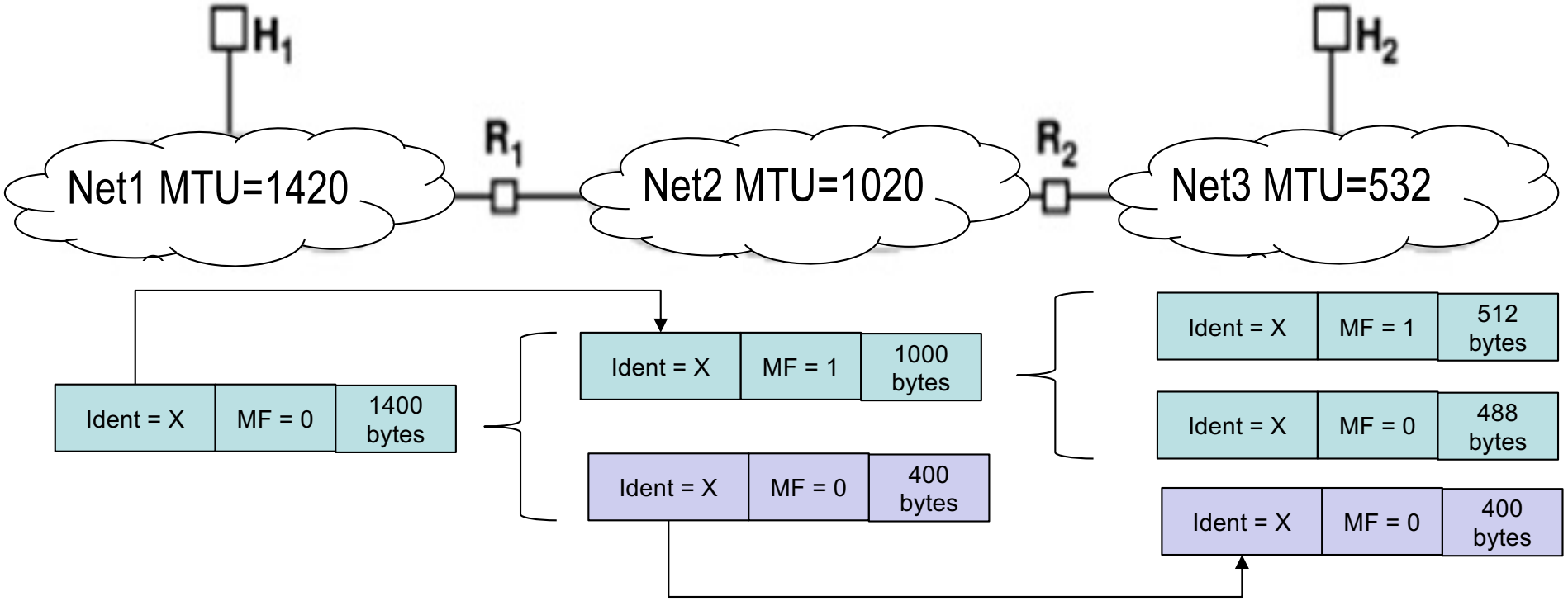


b3



Putem avea refragmentare?

- Răspunsul teoretic: DA / însă în practică, se folosește PathMTU și singura fragmentare se face la sursă



- Reasamblarea se face la destinație, unde se copiază datele în memorie pentru toate pachetele având Ident = X
  - se copiază tot ce primim pt. pachete până când primim un pachet având Ident != X
  - dacă un fragment întârzie, datele nu au sens și sunt discarded
    - fie la nivel IP dacă nu avem toate sub-fragmentele din ultima fragmentare
    - fie la nivel TCP dă eroare –se face din nou o verificare CRC

Clase de adrese IP

Fiecare nod (gazdă/ruter) are o adresă IP asociată cu interfața lui de rețea

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

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

Câteva adrese speciale

Prefix (rețea)	Suffix (gazdă)	Semnificație	Scop
toţi 0	toţi 0	acest calculator	Folosită când nodul încă nu are o adresă
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 adresă 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 aceeasi rețea fizica 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 <adresa_retea, adresa_gazda> destinație din datagramă

Caută o intrare cu adresa_retea în tabela de dirijare

if adresa_retea apare în tabela de dirijare
    if adresa_retea indică o rețea direct conectată **then**
        transmite datagrama direct la adresa_gazda
    else transmite datagrama următorului ruter (Next Hop)
else transmite datagrama unui ruter implicit

Adresarea ierarhica <adresa_retea, adresa_gazda> reduce numărul de intrări în tabela de dirijare (o intrare pentru o adresa_retea)

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

Exemplu

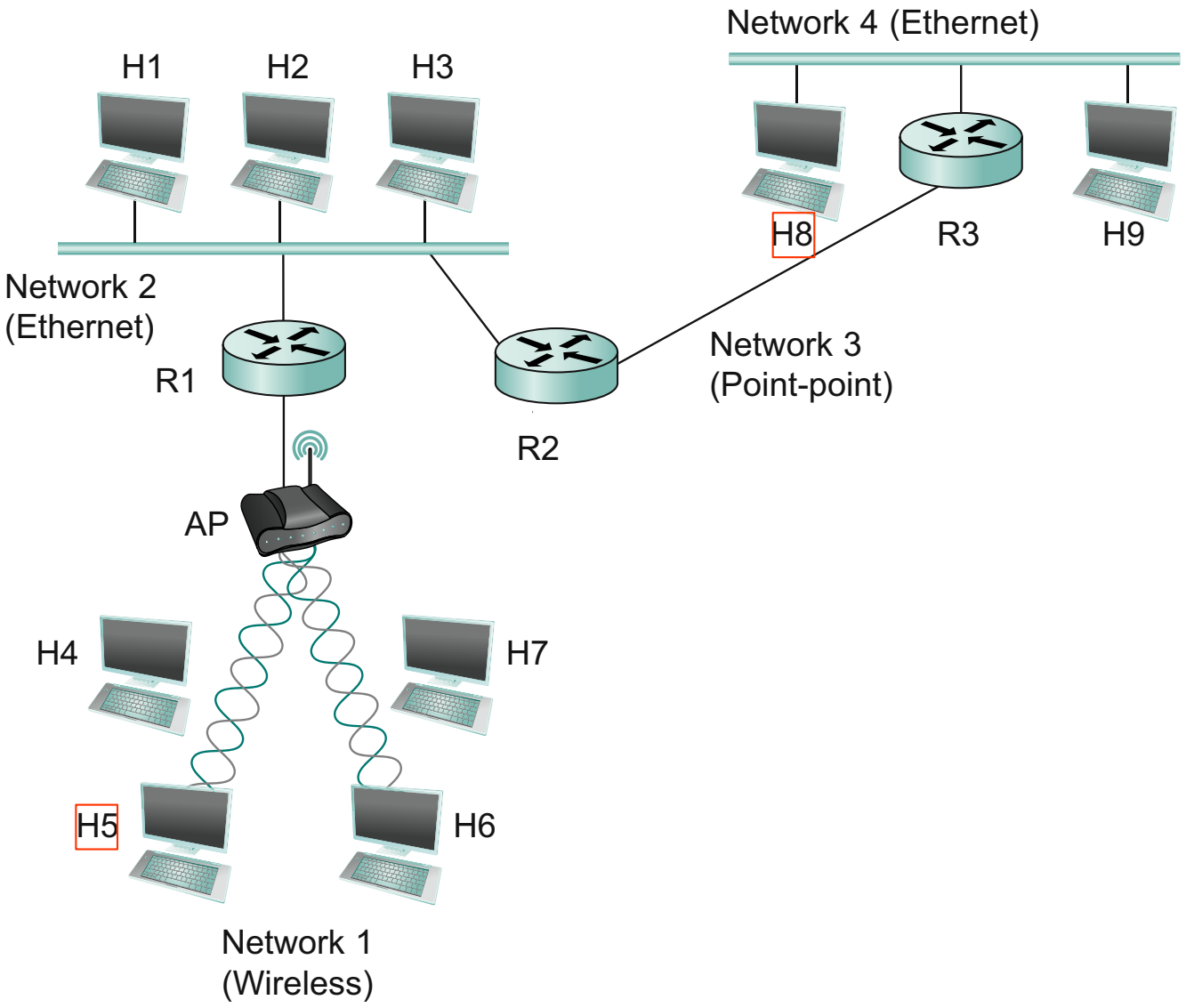
Transferul unei datagrame între H5 și H8

H5 are legătura (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



Subreţele

- Regula “o adresă pentru fiecare rețea fizică separată”
  - folosește ineficient spațiul de adrese
    - o rețea de clasă C cu 2 noduri consumă doar 2 din totalul de 255 de adrese de nod
    - o rețea de clasă 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 clasă B este împărțită în mai multe subrețele apropiate geografic
- soluția 2 - adrese fără clase

CIDR – Classless InterDomain Routing

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

Notația specială pentru adresa de rețea CIDR

194.24.0.0/21 ➔ din cei 32 de biți ai adresei IP

adresa_retea ocupă 21 biţi

adresa_gazda ocupă 11 biti

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

CIDR – Exemplu

Pentru a afla adresa_retea se folosesc măști

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

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

CIDR – reguli de alocare a adreselor

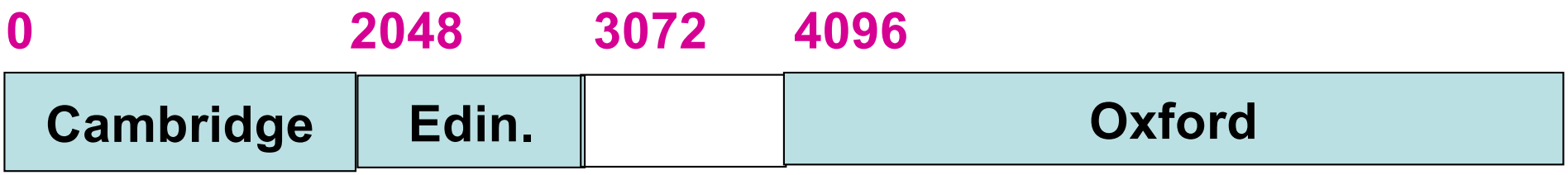
University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7. **255**	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Reguli pentru a avea o mască pentru un bloc de adrese

→ lungimea blocului trebuie sa fie o putere a lui 2

→ toate adresele din bloc au aceeași adresa_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



Algoritm *forwarding*

Intrare în 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

compară cu Cambridge /21 – adresa_rețea = 194.24.0.0

adresa_rețea: 11000010 00011000 00000000 00000000

Masca: 11111111 11111111 11111000 00000000

adresa_IP: 11000010 00011000 00010001 00000100

Adresa_IP AND Masca:

11000010 00011000 00010000 00000000

➔ = 194.24.16.0 ≠ 194.24.0.0 ➔ nepotrivire

Algoritm *forwarding (2)*

Ex. Sosește pachet cu adresa_IP = 194.24.17.4

Cambridge /21 – adresa_rețea = 194.24.0.0

(Adresa_IP AND Masca) = 194.24.16.0 ➔ nepotrivire

Edinburgh /22 - adresa_rețea = 194.24.8.0

(Adresa_IP AND Masca) = 194.24.16.0 ➔ nepotrivire

Oxford /20 - adresa_rețea = 194.24.16.0

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

Dacă nu sunt alte potriviri -> folosește intrarea pentru Oxford

Potriviri multiple

Prefixe de lungimi diferite

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

Ex.

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

Regula: se alege potrivirea “mai lungă”

Reducere dimensiune tabelă rutare

Soluție - agregarea unor adrese

Considerăm 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

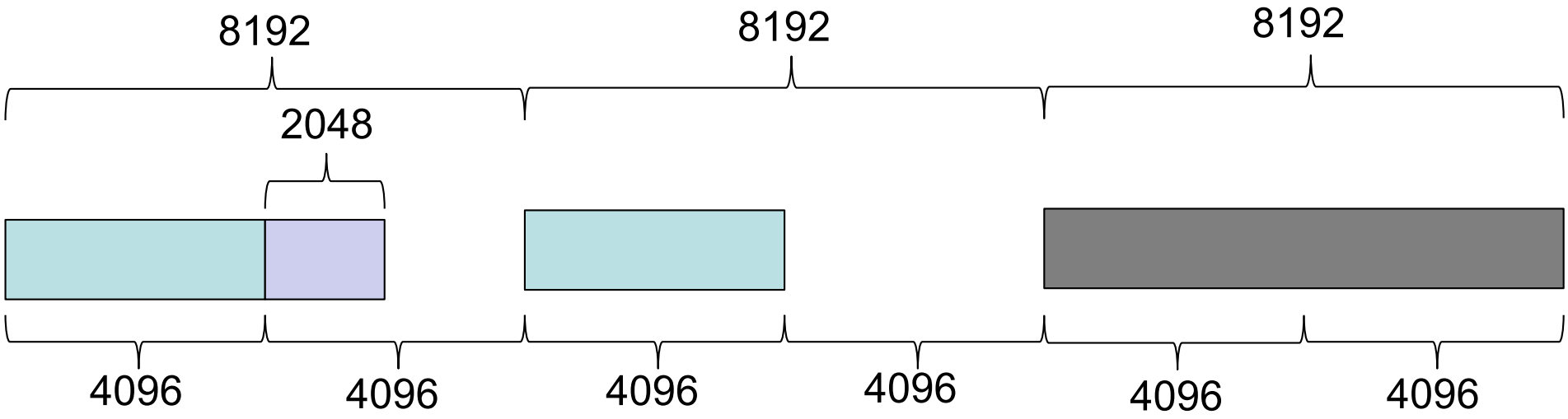
Tabela de dirijare poate include o singură intrare comună pentru

adresa_retea 11000010 00011000 00000000 00000000

adică 194.24.0.0/19

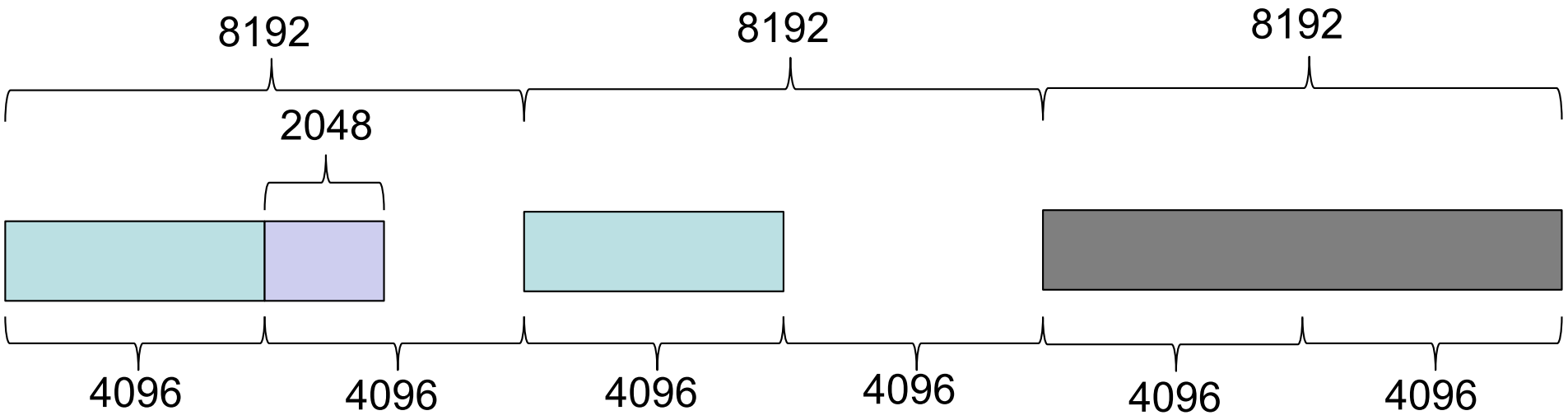
Exercițiu

- Un număr mare de adrese IP consecutive sunt disponibile începând cu 198.16.0.0. Să presupunem că patru organizații, A, B, C, D, cer câte 4000, 2000, 4000 şi 8000 adrese, în această ordine. Precizați, pentru fiecare dintre ele, prima și ultima adresă IP atribuită, precum și masca în notația **w.x.y.z/s**.
- 198.16.0.0 = 11000110.00010000.00000000.00000000
- Primul numar ^2 > 4000 = 4.096 (adrese de alocat pt. A, C)
- Primul numar ^2 > 2000 = 2.048 (adrese de alocat pt. B)
- Primul numar ^2 > 8000 = 8.192 (adrese de alocat pt. D)



Exercitiu (2)

- 198.16.0.0 = 11000110.00010000.00000000.00000000



- A: 4096 = 2^12 => hostID are 12 biti si **netID are 20 biti**
- 11000110.00010000.00000000.00000000 / 20 = 198.16.0.0 / 20
- B: 2048 = 2^11 => hostID are 11 biti si **netID are 21 biti**
- 11000110.00010000.00010000.00000000 / 21 = 198.16.16.0 / 21
- C: 4096 = 2^12 => hostID are 12 biti si **netID are 20 biti**
- 11000110.00010000.00100000.00000000 / 20 = 198.16.32.0 / 20
- D: 8192 = 2^13 => hostID are 13 biti si netID are 19 biti
- 11000110.00010000.01000000.00000000 / 19 = 198.16.64.0 / 19

Kahoot!

Nivelul Rețea
- ICMP / IPv6 -

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 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)

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 primeşte eroarea ICMP
    Destination Unreachable
    cu Fragmentation Needed and Don't Fragment was Set
  - Sursa trimite probe mai scurte

Conversie adresa IP – adresa fizică

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

Nu există o legătură biunivocă între adresa fizică și adresa IP

De ex.

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

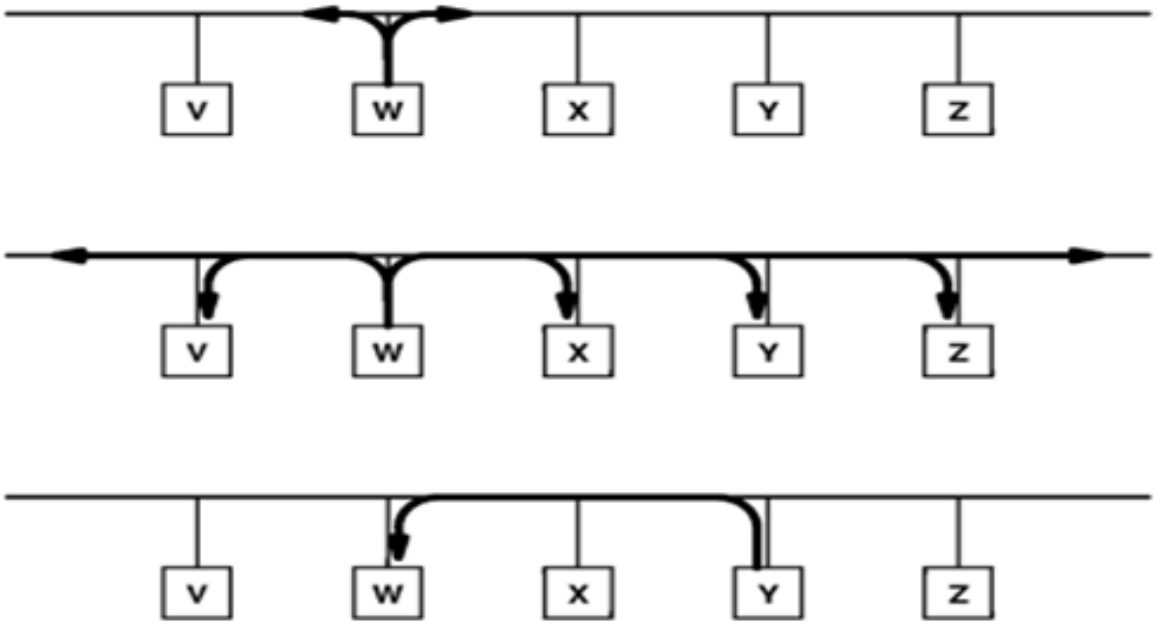
Pentru mapare se pot folosi

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

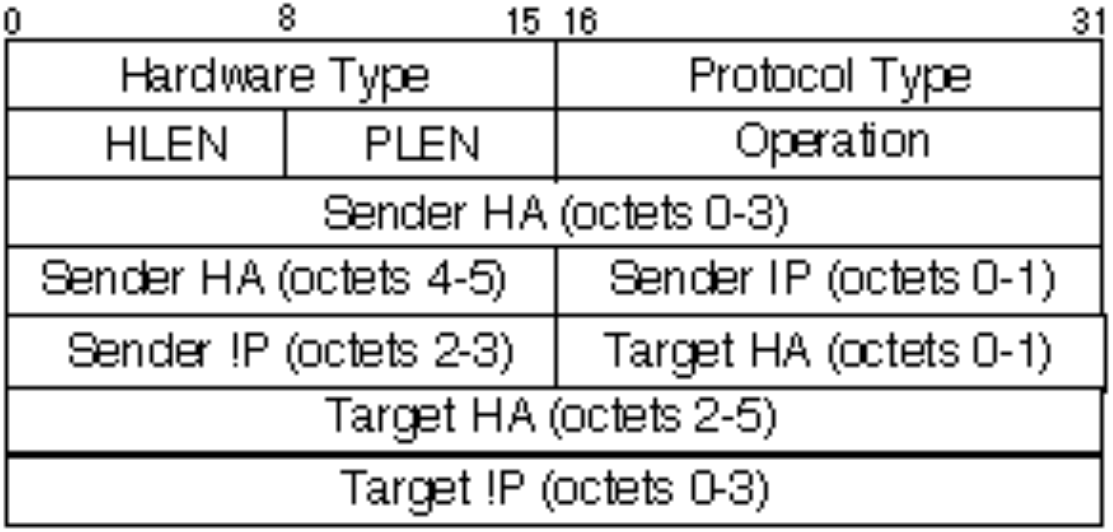
ARP - Address Resolution Protocol

Secvența de mesaje pentru a afla adresa fizică

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



Conversie adresa IP – adresa fizică



Format mesaje ARP – conceput pentru diverse categorii de adrese

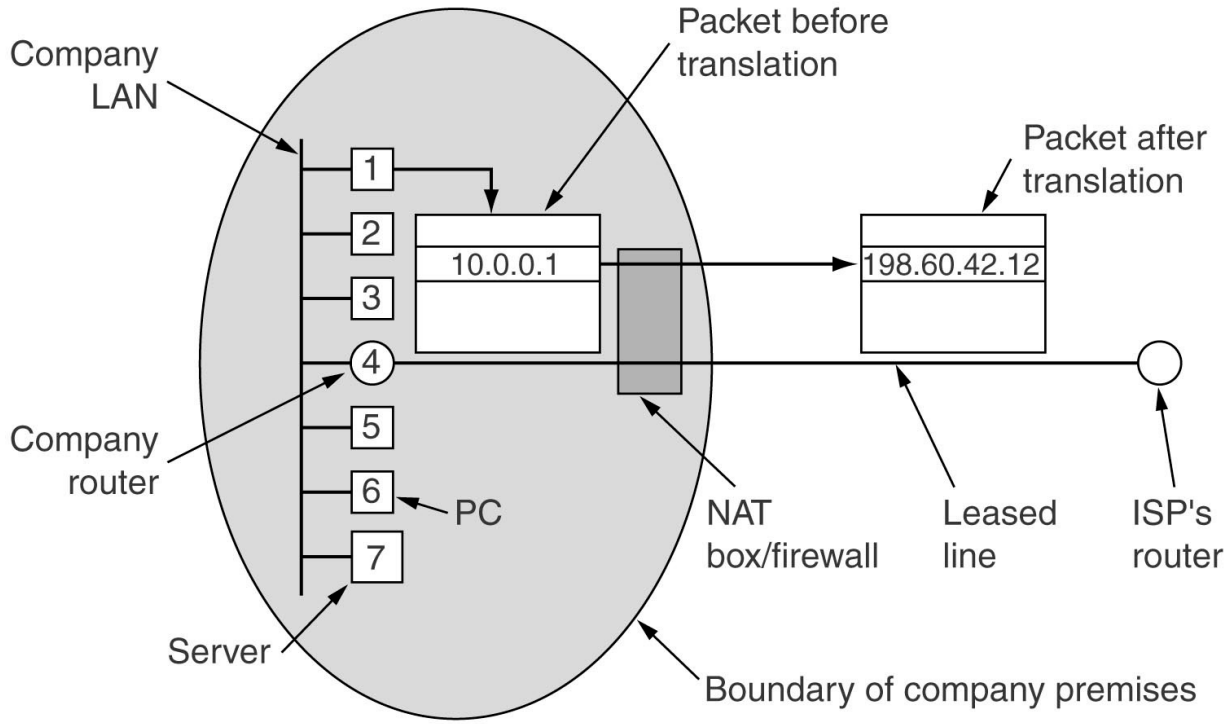
- Hardware type: 1 pentru Ethernet
- Protocol type: 0x0800 pentru IP (0000.1000.0000.0000)
- HLEN - Hardware len: 6 octeți pentru Ethernet
- PLEN - Protocol len: 4 octeți pentru IP
- Operation: 1=cerere, 2=răspuns

NAT – Network Address Translation

O adresă pentru mai multe calculatoare

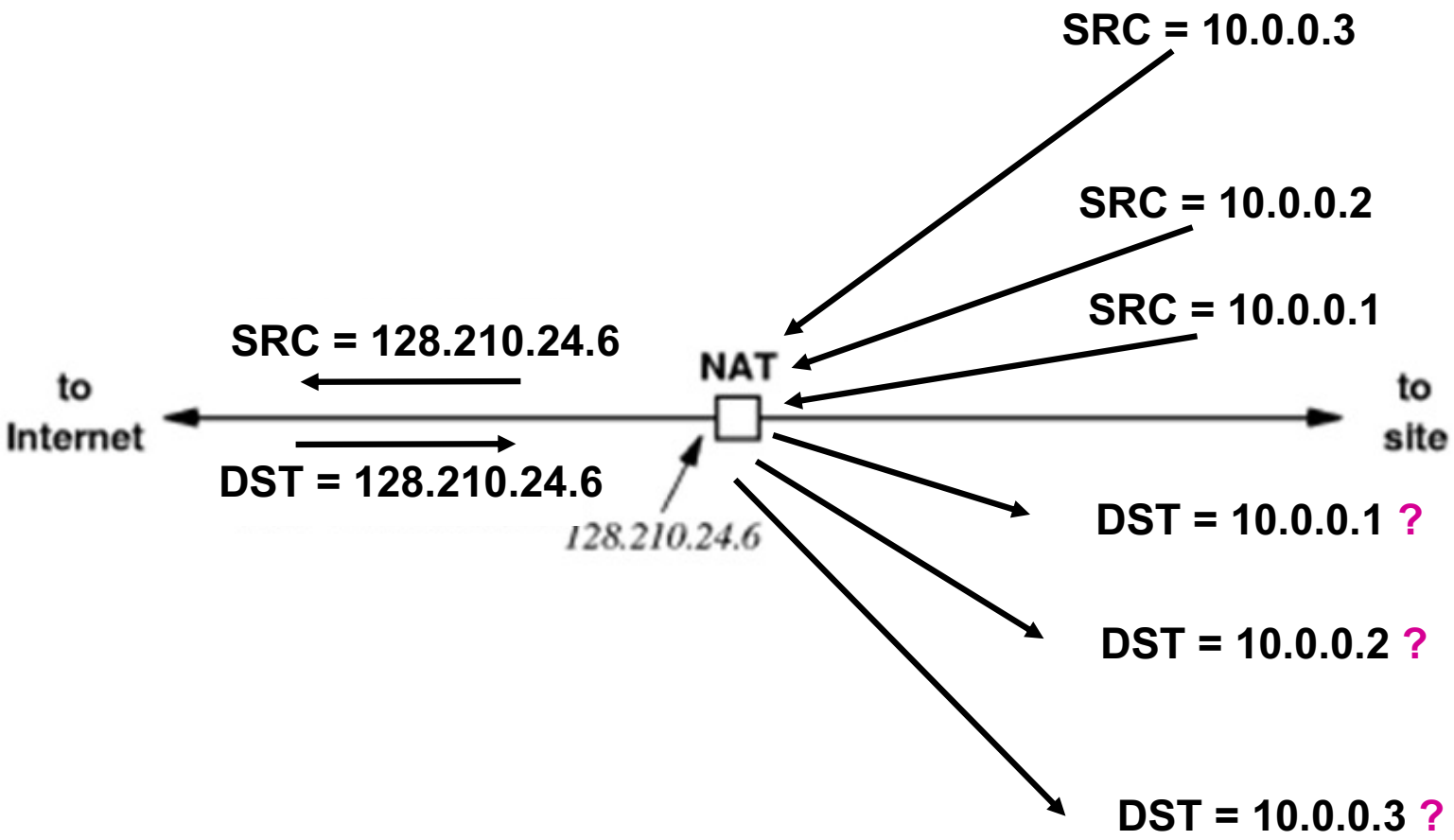
Foloseşte adrese locale (private sau non-rutabile)

NAT translatează între adresa privată şi o adresă globală



- **192.168.0.0 – 192.168.255.255** (65.536 adrese IP)
- **172.16.0.0 – 172.31.255.255** (1.048.576 adrese IP)
- **10.0.0.0 – 10.255.255.255** (16.777.216 adrese IP)

Translatarea adresa globală → adresa privată



Principiul NAT

Foloseşte

adresa IP + număr port transmitator

tabela de translatare

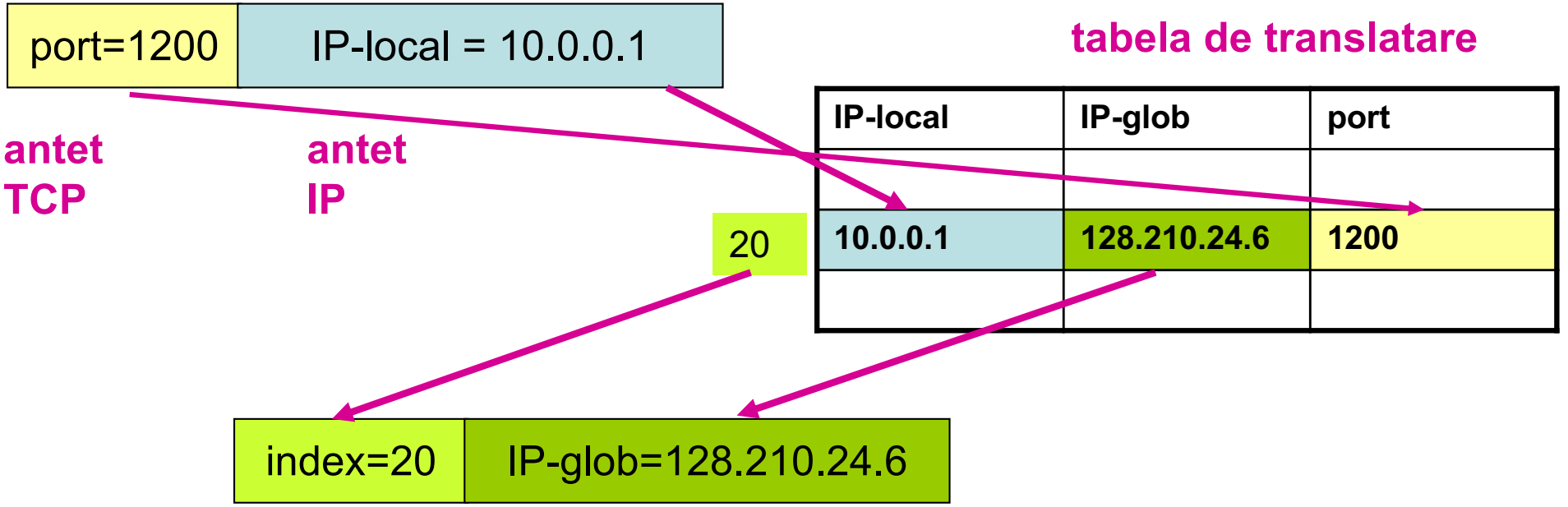
Transmisie

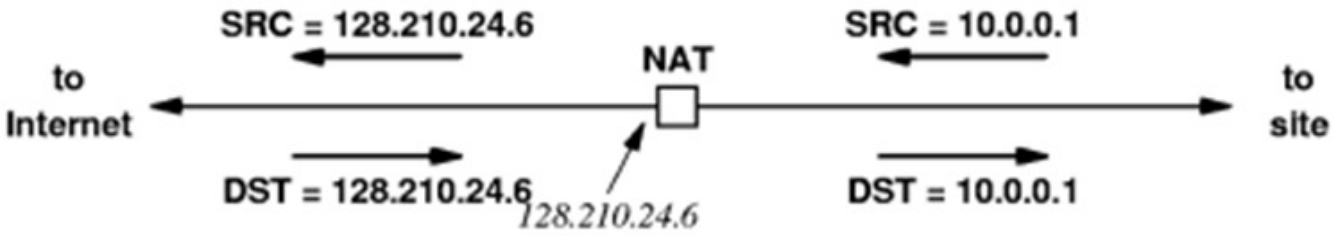
înlocuieşte adresa IP locală cu o adresă IP globală

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





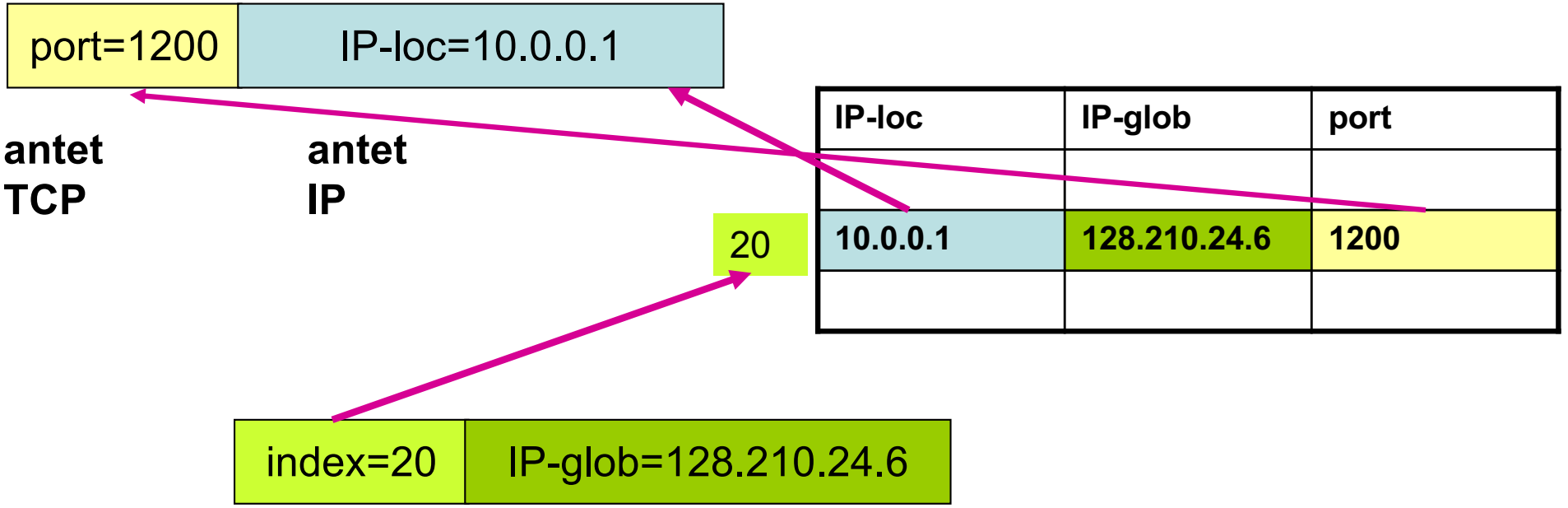
Receptie

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



IPv6 - Motivaţii

Spaţiul de adrese

32 biţi = peste un milion de reţele

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

Tip servicii

Aplicaţii diferite au cerinţe diferite de livrare, siguranţă şi viteză

IPv4 are tip de serviciu dar adesea nu este implementat

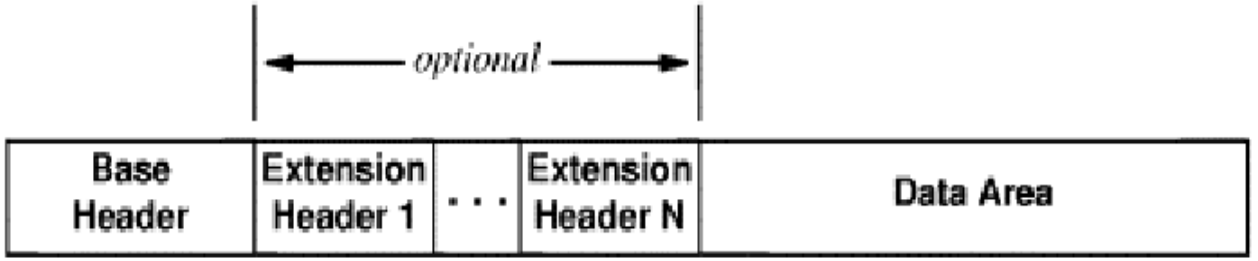
IPv6 - format Base header

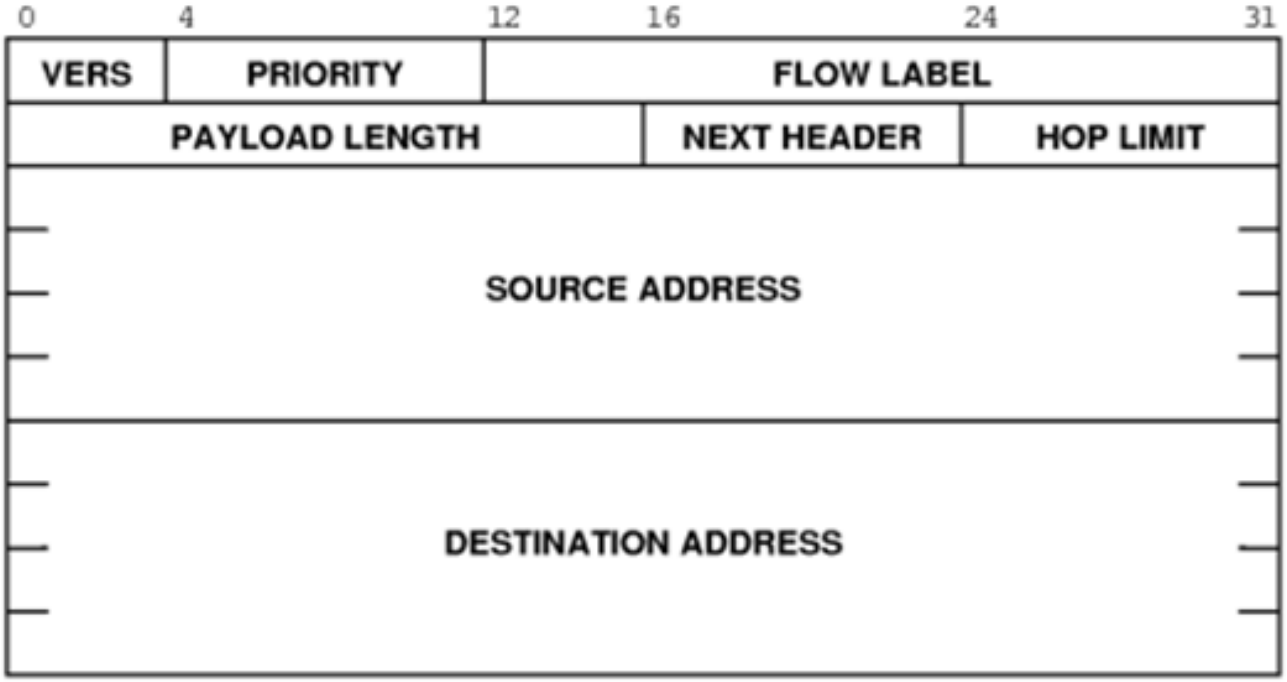
lungime fixă = 40 octeti

Priority - clasa de trafic

FLOW LABEL - asociază datagramele unui flux

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





Conţine mai puţine info decât antet IPv4

Restul de info în extensii

NEXT HEADER defineşte tipul datelor (ex. TCP)

NEXT HEADER defineşte tipul antetului de extensie urmator (ex. Route Header)

(a)

(b)

IPv6 – antete extensie

Hop-by-hop header – info pentru rutere – deocamdata:

suport datagrame excedând 64K (jumbograme)

specifica lungimea;

campul de lungime din antetul de baza este 0

Destination header – info aditionale pentru destinaţie

nefolosit

Routing – lista rutere de vizitat

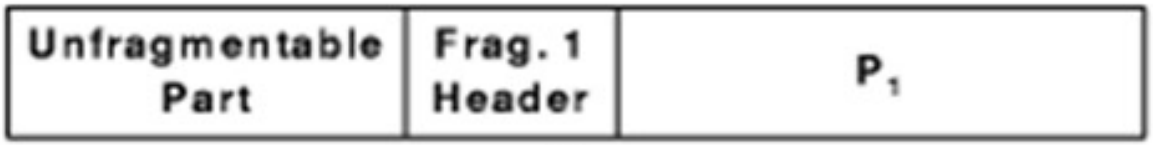
Fragmentation – identificare fragmente

Authentication – verificare identitate transmiţător

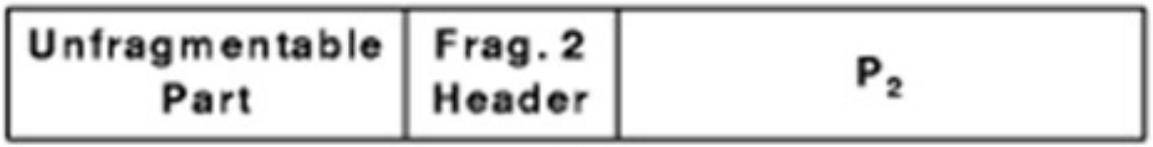
Encrypted security payload – info despre conţinut criptat

Fragmentarea

(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

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 adresei

16 numere

105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255

Notaţie hexazecimală

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

Compresie zerouri

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

FF0C::B1

adrese IPv6 cu 96 zerouri prefix sunt interpretate ca adrese IPv4

Nivelul Rețea
- Algoritmi de rutare -

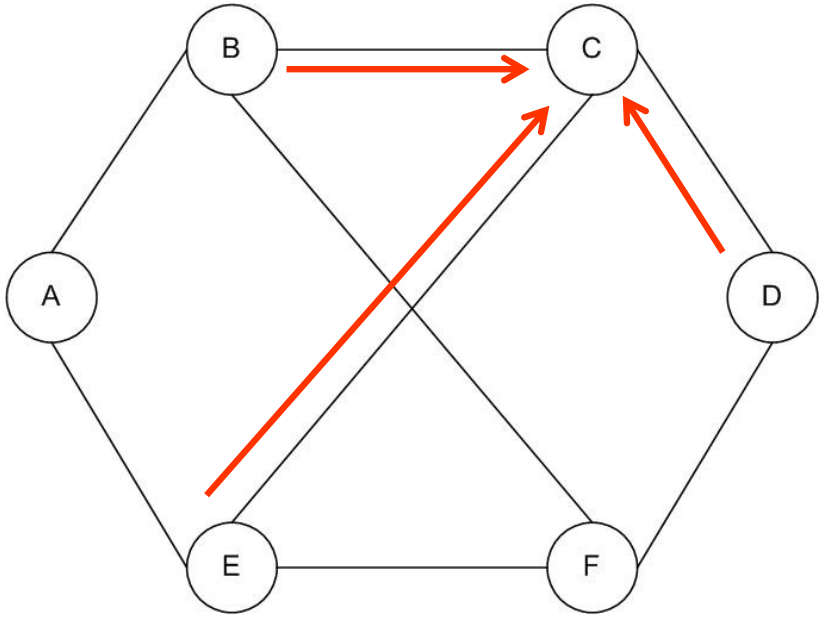
Dirijarea - clasificare

- 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 distanţelor

Algoritm distribuit !

Fiecare nod trimite periodic vecinilor săi o listă cu distanțele 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):

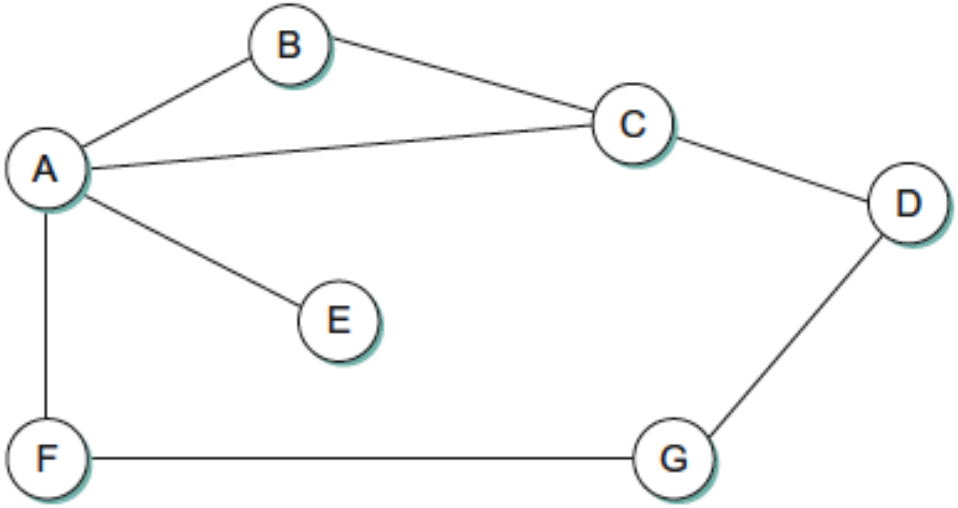
De la B: (5, 0, 8, 12, 6, 2);
De la D: (16, 12, 6, 0, 9, 10);
De la E: (7, 6, 3, 9, 0, 4).

De la / 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

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

Costurile de la C / La ↓ — Prin →	B	D	E		Cost Min	Pas următor
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 legatură

1 2 2 3 0 2 3 ← distanțe inițiale la E

∞ **2** **2** 3 0 2 3 legătura A – E cade

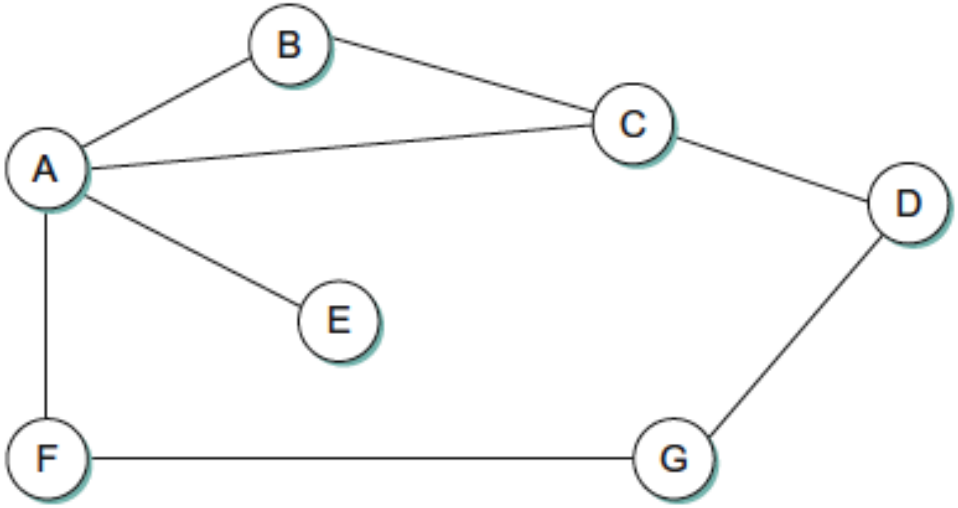
distantele către E anunțate de: **A = ∞, B = 2, C = 2**

În funcție de ordinea evenimentelor, se pot face modificările:

∞ **3** **2** 3 0 2 3 B alege ruta prin C, dist=3

B anunță 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 – soluții (3)

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

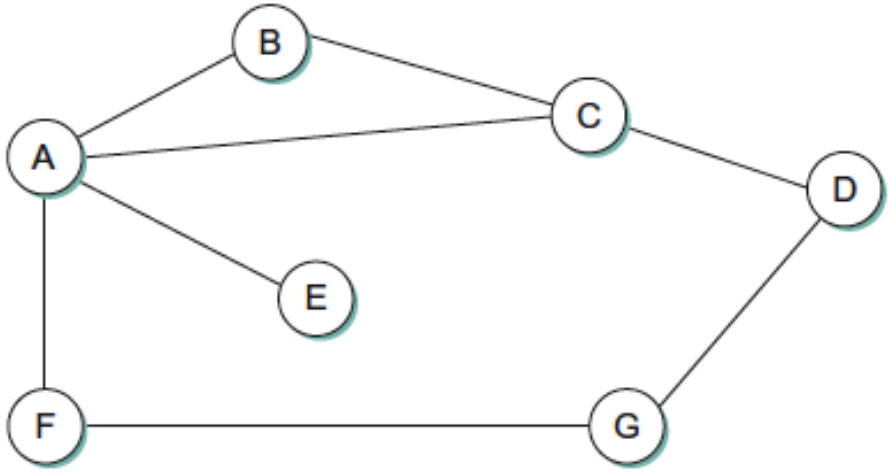
- B are ruta de distanță 2 către E prin A
- B nu include noua distanță către E în actualizarea trimisă lui A

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

- B trimite distanța ∞ către A
- A nu va mai alege o cale prin B

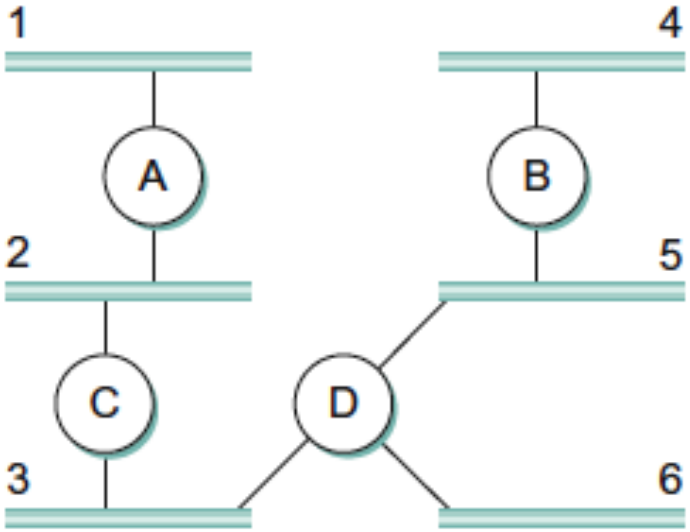
Neajuns: soluțiile nu funcționează în toate cazurile

- de ex. pentru bucle cu mai multe noduri



RIP - Routing Information Protocol

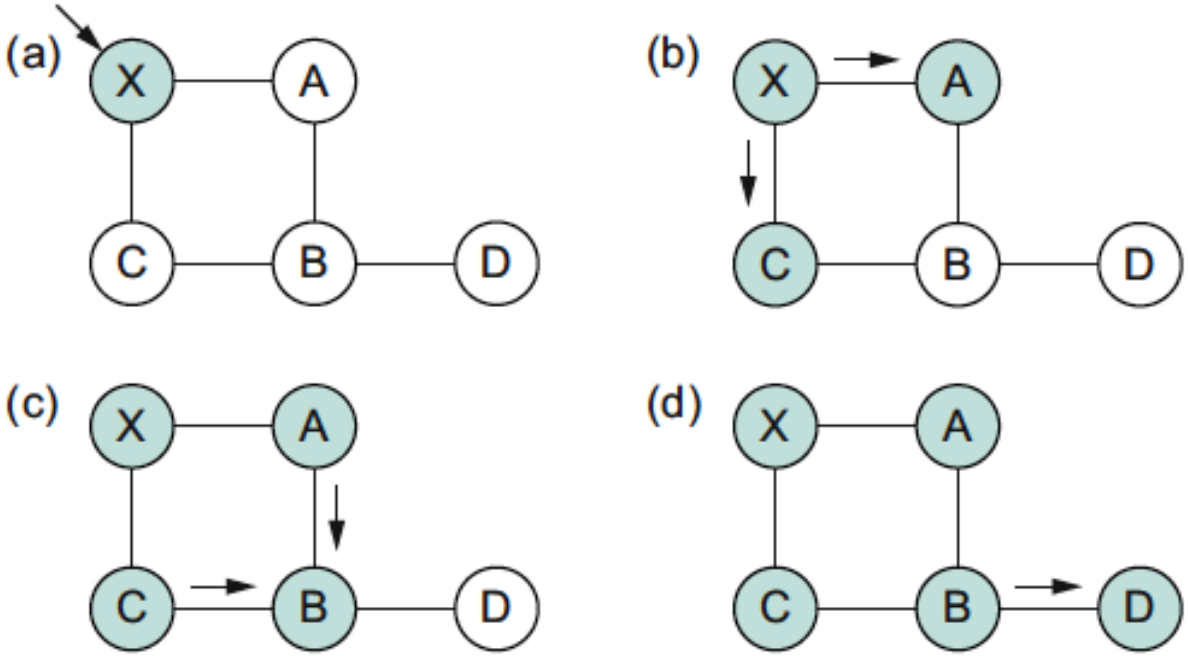
- Fiecare legătură are cost 1
- Folosește distanțe la rețele (nu la noduri)
  - ruterul C are distanța 0 la rețeaua 2 și 2 la rețeaua 4
- Transmit vectorii distanțelor la fiecare 30 secunde
- Distanțe maxime de 15 hop-uri (16 înseamnă infinit)
  - rețele de mici dimensiuni



Starea legăturii

- Presupune că fiecare nod poate găsi legăturile cu vecinii și costul fiecărei legături
- Informațiile sunt diseminate prin inundare tuturor celorlalte noduri
  - LSP – Link State Packet transmis prin inundare;
  - Pachetul conține
    - Id-ul nodului care creează pachetul
    - lista nodurilor conectate cu costul fiecărei legături
    - un număr de secvență
    - durata de viață a pachetului (număr)
- Cu informațiile primite, fiecare nod va calcula rutele cele mai scurte către 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:
  - număr decrementat la fiecare nod (hop) – asigură eliminare pachete vechi

Algoritmul căii celei mai scurte

Algoritmul lui Dijkstra

nnod	numărul nodurilor reţelei;
sursa	nodul sursă;
l[i][j]	costul legăturii (i,j), având valorile 0 dacă i = j; lungmax dacă i şi j nu sunt adiacente; o valoare între 0 şi lungmax în celelalte cazuri;
D[i]	costul minim al legăturii de la sursă la i;
S	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];             // distanțele minime de la sursă
            if (D[i] < lungmax)
                    V[i] = i;               // inițializează vecinii
            else
                    V[i] = 0;
     }
    S[sursa] = 1;                           // selectează 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++)               // recalculează distanțele
                    if ((S[j] == 0) && (D[k] + l[k][j] < D[j]))
                            { D[j] = D[k] + l[k][j];
                              V[j] = V[k];          // modifică tabela de dirijare
                            }
      }
}
```

Structura ierarhică a Internet-ului

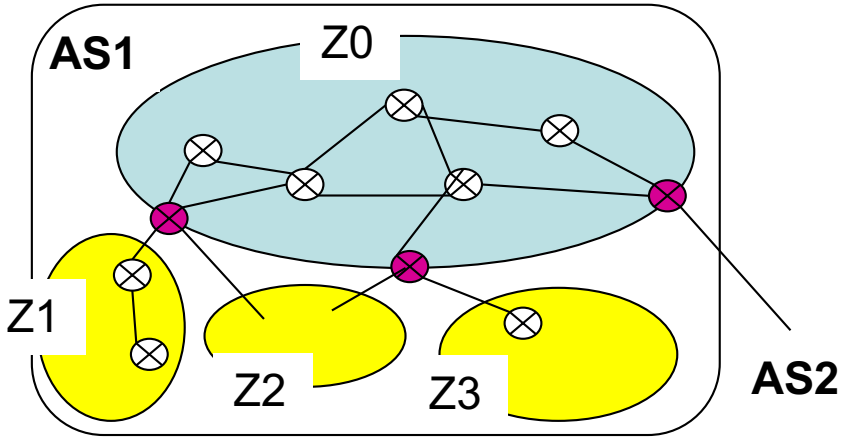
- Internet-ul este partajat în mai multe domenii numite sisteme autonome AS – Autonomous Systems
  - ASs sunt rețele independente operate de organizații diferite
- Într-un AS se folosește același algoritm de rutare “intra-domeniu”
  - denumit interior gateway protocol
  - ex. OSPF – Open Shortest Path First
- Rutarea între AS-uri (între domenii) folosește, de asemenea, un protocol comun
  - denumit exterior gateway protocol
  - ex. BGP – Border Gateway Protocol (bazat pe RIP)

Structura ierarhică AS

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

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

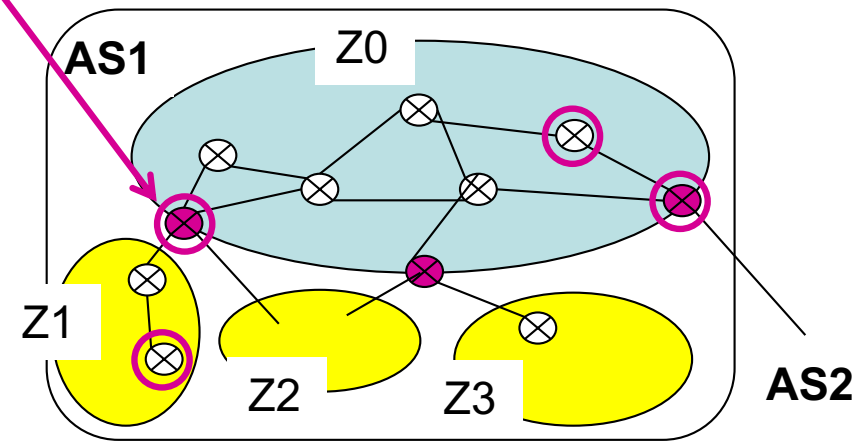
- Ierarhizarea crește scalabilitatea
  - un ruter dintr-un AS nu trebuie să știe cum se ajunge la fiecare rețea din AS, fiind suficient să știe cum se ajunge la zona în care se află rețeaua respectivă → reduce volumul tabelelor de dirijare



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**



Mesaje OSPF

Mesajele OSPF permit schimbul de informații între noduri

Sunt transmise în pachete IP cu 89 ca număr de protocol

Hello – stabilește și păstrează legături cu vecinii

descoperă 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 legături, ca răspuns la o cerere

Link state ack - Confirmare stare legătură

confirmă primirea unui mesaj de actualizare

Database description – trimite LSBD – LInk State Data Base

mesajele conțin info despre AS sau zonă

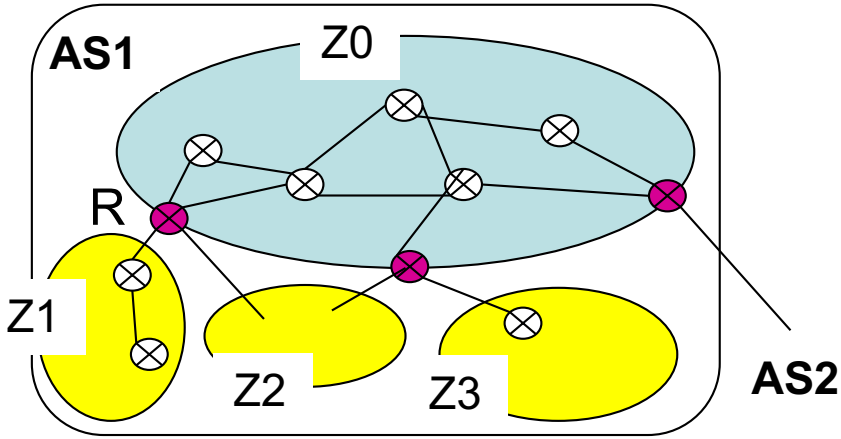
Calcul rute - Nivel 1 (zonă)

Folosind inundarea, fiecare ruter informează celelalte rutere din **zonă** despre legăturile sale și costurile acestora

- ex. informațiile de starea legăturilor schimbate între noduri din Z1 nu se transmit în afara acestei zone!

Fiecare ruter (inclusiv cele de graniță zonală) din **zonă** calculează separat căile cele mai scurte către ruterele din aceeași zonă

- în final, ruterul de graniță zonală R va cunoaște căile cele mai scurte către oricare rețea din Z1 și Z2



Calcul rute - Nivel 2 (AS)

Ideea: calea unui pachet între două zone diferite are trei părți:

- de a nodul sursa la zona backbone
- traversează backbone
- de la backbone la rețeaua de destinație

Ruterele de coloană vertebrală (backbone)

primesc informații de la ruterele de granița zonale și

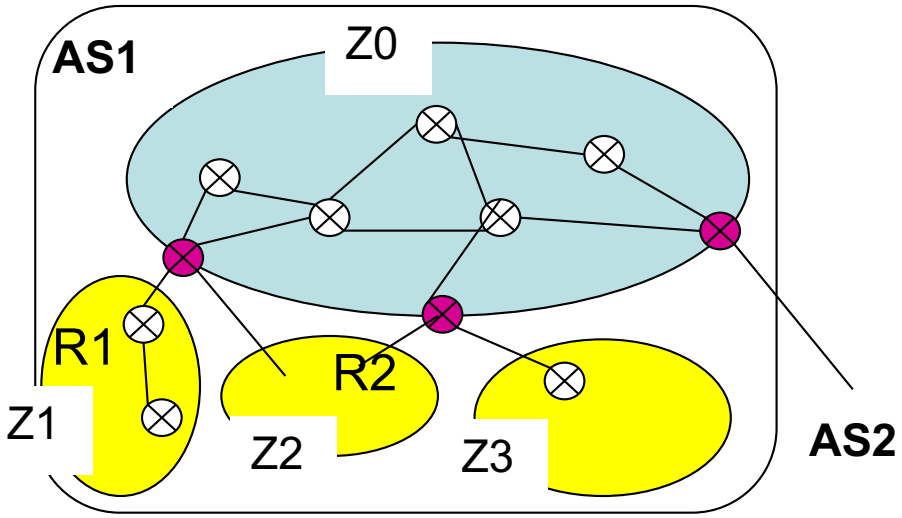
calculează cele mai bune rute la rețele din orice zonă

ex. R1 oferă căile cele mai scurte pentru Z1 și Z2

ruterele din Z0 calculează căile către orice rețea din Z1,Z2

Rezultatele sunt difuzate de la Z0 înapoi la zonele stub, care actualizează căile cele mai scurte la rețele din alte zone

Ex. ruterele din Z2 vor ști să aleagă între R1 și R2 pentru rutare spre retele din alte zone



BGP – Border Gateway Protocol

Algoritmi orientați pe aspectele politice, de securitate, economice

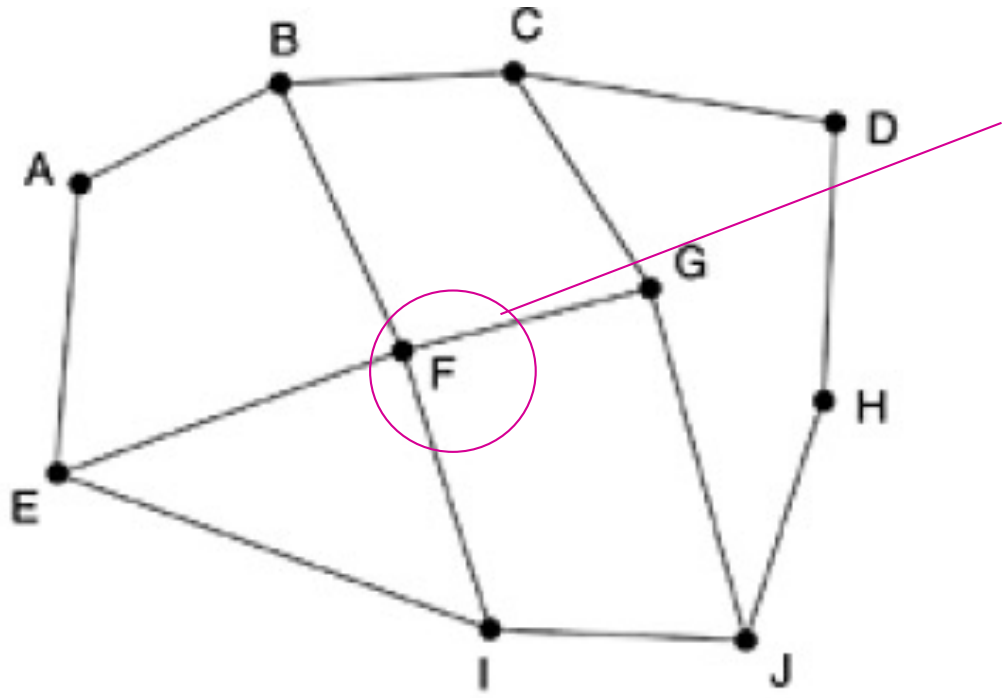
Reţea = ASes şi conexiunile

Protocol = vectorul distanțelor

Tabelele de dirijare conțin și rutele spre destinație

Comunică vecinilor căile utilizate efectiv (ocolește numărătoarea la infinit)

Ex. pp. F folosește calea FGCD la D



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

De la B: “Eu folosesc BCD”
De la I: “Eu folosesc IFGCD”
De la E: “Eu folosesc EFGCD”

F elimină căile care conțin F
și alege calea FBCD

Studiu individual

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

5.1 CERINŢELE DE PROIECTARE ALE NIVELULUI REŢEA

5.2.2 Dirijarea pe calea cea mai scurtă

5.2.3 Inundarea

5.2.4 Dirijare cu vectori distanţă

5.2.10 Dirijarea în reţele AD HOC

5.6.1 Protocolul IP

5.6.2 Adrese IP

5.6.4 Protocoale de control în Internet

5.5.5 Protocolul de dirijare folosit de porţile interioare: OSPF

5.6.5 Protocolul de dirijare pentru porţi externe: BGP

5.6.8 IPv6

Studiu individual

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

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