

## CURS 4

192.167.1.17 – format zecimal cu punct

Reprezentarea în memorie a adresei IP:

- **IPv4** – pe 4 octeți (cu valori între 0 și 255)
- **IPv6** – pe 16 octeți

**INET\_ADDR** – convertește adresa IP dată ca *string* (în format zecimal cu punct) în reprezentarea internă.

EX: `inet_addr("192.168.1.17")`

`{"192", "168", "1", "17"} => (atoi) {192, 168, 1, 17}`

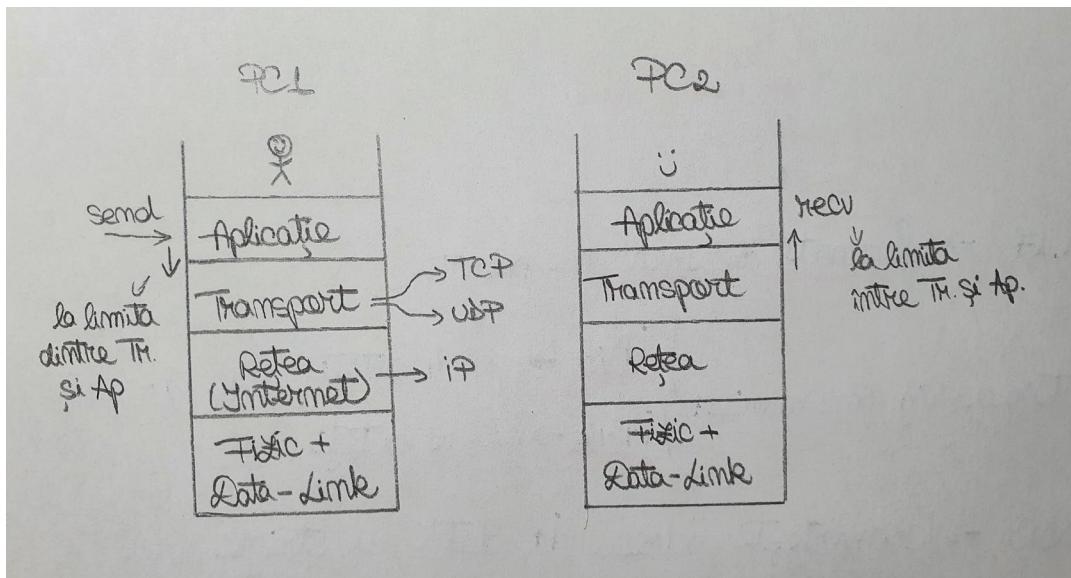
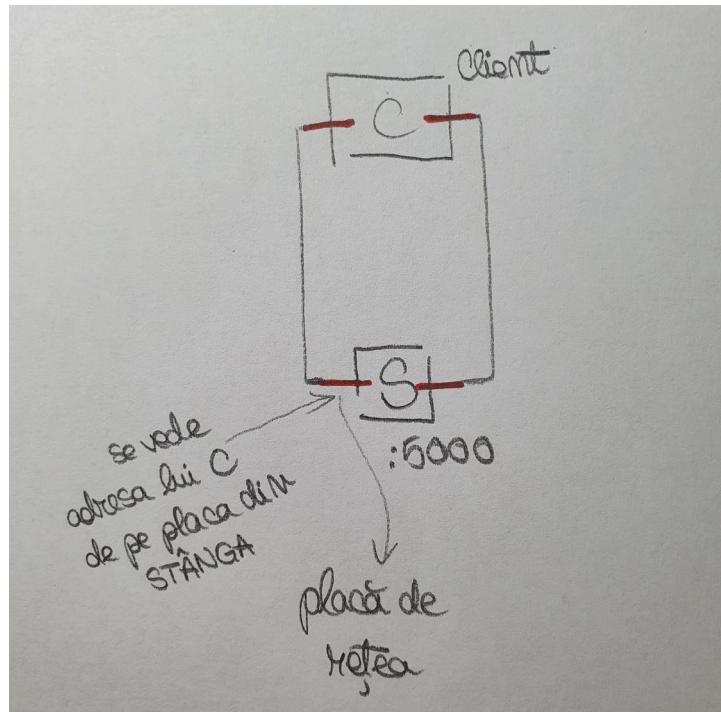
$\rightarrow 192 * 256^3 + 168 * 256^2 + 1 * 256 + 17$

$\rightarrow 192<<24 + 168<<16 + 1<<8 + 17$  (shiftări pe biți)

**INET\_NTOA** – primește ca parametru adresa în *reprezentarea internă* și o convertește la string (Network TO Ascii)

*Comenzi terminal:*

- **nc (network cat)** – arată conținutul rețelei
  - `nc -l 5000` – face un server pe portul 5000 care stă și așteaptă ceva
- **netstat -lnp** – arată toate procesele server care așteaptă pe calculator
  - `-l` : arată toate socket-urile în starea *listen*
  - `-n` : arată *numărul portului* în loc de numele protocolului și *adresa IP* în locul numelui calculatorului
  - `-p` : arată *PID*-ul procesului
- **telnet localhost 5000** – face un connect la serverul de pe localhost cu portul 5000



Orice proces, orice client, server, este localizat la nivelul *Aplicație* a stivei TCP/IP.

Când se trimit pachete, acestea coboară de la nivel la nivel, fiecare “pasând” (folosind *memcpy*) informația utilă de la nivelul superior la nivelul inferior. Fiecare nivel inferior adaugă niște antete/câmpuri în fața informației utile (**headere**). La nivel *minimal*:

- Data-Link: adaugă (MAC sursă, MAC destinație)

- Rețea: adaugă (IP sursă, IP destinație)
- Transport: adaugă (port sursă, port destinație)

**!!!** Informația utilă nu se copiază de la un nivel la altul

**!!!** Fiecare nivel vede informația utilă ca fiind informația utilă originală + toate celelalte headere adăugate de nivelurile anterioare

### Ce se întâmplă când se face *send* de mai mulți octeți? (ex. 10k octeți)

Nivelul Transport are grija să “rupă” acel send în mai multe “pliculete” (pachete de informație). Dimensiunea unui pachet depinde de mai mulți factori:

- **MTU (Maximum Transfer Unit)**: pentru cablul UTP – 1500 octeți (dimensiunea informatiei utile + toate headerele)
- **Protocolul de la nivel Transport** (64 kilo)
  - nivelul transport rupe în plicuri de maxim 64 kilo
  - nivelul data-link ia plicul, îl rupe în mai multe pachete mai mici, și la fiecare pachet adaugă header-ul care se adaugă la nivel DL

## Diferența dintre TCP și UDP

### TCP

- protocolul orientat pe conexiune
- asigură livrarea sigură datelor de la sursă la destinație (*reliable*)
- la nivelul Transport, se împart datele în pachete și se “numerotează”
- se trimit datele la nivelul Rețea și tot mai jos, până ajung în cealaltă parte, la nivel Transport; datele trebuie pregătite, la nivel Transport, deoarece la receptor, tot la nivelul Transport, trebuie rearanjate exact în ordinea trimisă
- se asigură că informația este aranjată exact în ordinea în care a fost trimisă
- dacă lipsește un pachet, receptorul anunță ce pachet nu a primit, prin niște confirmări (implicite sau cumulate) → emițătorul retrimit pachetele neprimită (nu instant, se mai trimit alte pachete anticipat, ca să se verifice dacă nu cumva acel pachet pierdut nu ajunge cu întârziere)

### UDP

- datele se pot pierde (*nu este reliable*)
- nu se fac toate verificările ce se fac la nivel Transport în cazul TCP
- este mai rapid decât TCP, datorită faptului că nu se fac atâtea verificări

Protocole care folosesc UDP: cele care se bazează pe mecanism cerere-răspuns (ceea ce cere clientul serverului începe într-un singur plic, și răspunsul serverului începe și el într-un singur plic) – *NTP* (Network Time Protocol), *DNS* (Domain-Name Server) – ambele sunt de la nivel Aplicație

## CURS 5 + 6 + 7

### Notiunea de protocol

În cadrul stivei TCP/IP, la nivel Aplicație, este localizat orice proces, server. La acest nivel, aplicațiile comunică prin protocole: *HTTP* (browser – clientul, server web), *FTP* (File Transfer Protocol), *SMTP* (Simple Mail Transfer Protocol), *POP* (Post Office Protocol), *IMAP*, *DNS*, *NTP*.

**Protocol** = set de reguli pe care trebuie să le respecte doi parteneri care comunică.

Pentru toate protocolele din stiva TCP/IP există documente standard numite **RFC** (Request For Comments) care descriu foarte precis toate aspectele de comunicare de care trebuie să țină cont un implementator de client și unul de server, ca să le facă interoperabile, să funcționeze împreună.

### Protoale text vs. Protoale binare

```
unsigned short int i = 65123; // același lucru cu uint16_t
```

Cum scriem acest număr într-un fișier?

Varianta 1:

```
f = open("date.txt", ...)  
write(f, &i, sizeof(i)); // FE63 – în memorie 63h FEh – se scrie numărul  
// aşa cum e el în memorie
```

Varianta 2:

```
f = fopen("date.txt", ...)  
fprintf(f, "%uh", i); // se scriu 5 octeti: "65123", adică cifrele  
// numărului
```

**Cum e cel mai bine să reprezentăm numărul în fișier?**

Dacă vrem să scriem un număr într-un fișier text, e nevoie de un delimitator, în cazul în care vrem să scriem după el un alt număr. În protocol, trebuie definit acest delimitator.

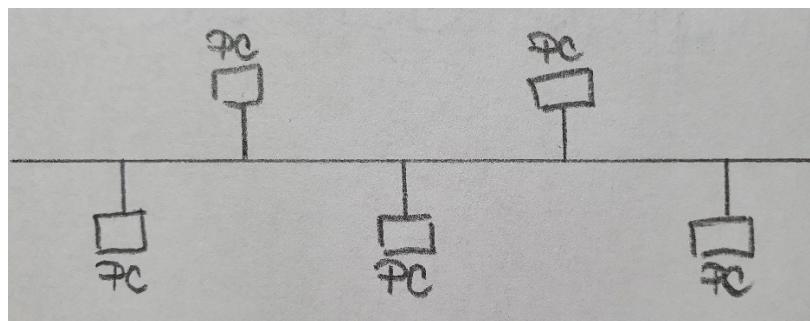
În format binar, trebuie să fim atenți la dimensiunea de reprezentare.

!!! În internet, se folosesc foarte mult protocolele text, pentru că e mai ușor de făcut debugging și e mai ușoară de interpretat informația care se trimite pe pipe.

## Topologii de rețele

### Rețele cu topologie liniară (topologie BUS)

- toate calculatoarele erau conectate la același fir de comunicații (există o magistrală unică care conectă toate calculatoarele)



- operaau la viteze de până la 10 Mbps, partajați între toate calculatoarele din cadrul rețelei (ex. dacă două calculatoare vor să comunice între ele, iar restul nu făceau trafic în internet, ele puteau comunica cu 10 Mbps; dacă voiau să comunice, în același timp, 4 calculatoare, 2 câte 2, viteza maximă cu care puteau să comunice 2 calculatoare se înjumătățea)
- această limitare a vitezei este datorată coliziunilor – fenomen strâns legat de rețelele cu difuzare

**Rețele cu difuzare** – dacă un calculator vrea să comunice cu altul, informația care se trimit nu se propagă numai pe segmentul de rețea dintre cele două calculatoare, ci se propagă pe toata lungimea firului → informația ajunge pe toate calculatoarele din rețea → este la latitudinea fiecărui calculator dacă vrea să primească informația → probleme de securitate.

!!! Singura metodă de contracarare a difuzării este **criptarea**

- dacă se rupe cablul de rețea, niciun calculator nu mai poate comunica, chiar dacă unele se află pe același segment întreg

Obs:

1 octet = 1 byte = 8 biți

1 Kilobyte = 1024 =  $2^{10}$  octeți

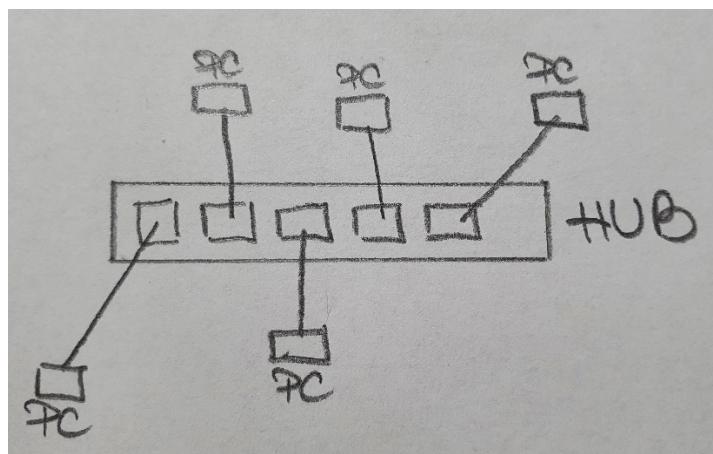
1 Megabyte = 1024 Kilobytes

1 Gigabyte = 1024 Megabytes =  $2^{30}$  octeți

→ unități care măsoară cantitatea de date

1 Kilobit = 1000 biti

### Rețele cu topologie stea

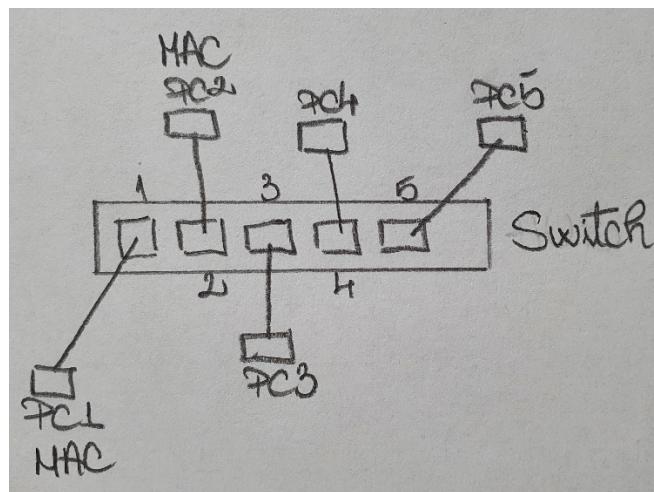


- primele rețele operaau cu 10 Mbps
- în centrul rețelei se află un HUB, care lua semnalul trimis de un calculator și trimitea tututor calculatoarelor din rețea, nu numai calculatorului căruia îi era destinată informația
- dacă pica linia de comunicație între hub și un calculator, celelalte calculatoare din rețea continuau să funcționeze
- se păstrează problemele de coliziune și de înjumătățire a vitezei
- un HUB poate fi văzut ca un amplificator de semnal

## HUB vs. Switch

Spre deosebire de un HUB, un switch e un echipament “smart”, el are procesor și memorie, poate să ruleze și să implementeze logici funcționale. Un atu mare este că switch-ul își numerotează porturile și învață, în timp, ce calculator se află în spatele fiecărui port. Astfel, se evită coliziunile și viteza nu scade.

Switch-ul operează la nivelul Data-Link. Fiecare calculator este identificat printr-o adresă MAC (adresă Ethernet, adresă Hardware).

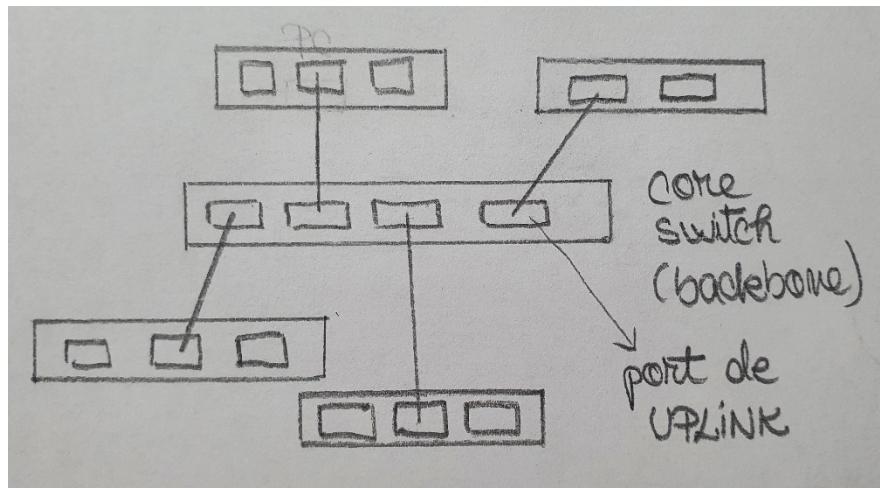


La nivel Data-Link, structura de date care se trimite, conținută din informația utilă + header-ele de la nivelul DL, poartă numele de *frame* (*cadru*). La nivel mai sus în stiva TCP/IP, acest frame se numește *pachet*.

Pe măsură ce calculatorul PC1 vrea să trimită un frame la PC2, switch-ul e capabil să se uite la fiecare frame care circulă pe porturile sale și să vadă de unde vine. Apoi, switch-ul ține minte că în spatele portului 2 se găsește calculatorul cu MAC-ul lui PC2.

Inițial, switch-ul nu știe unde să trimită frame-ul, aşa că îl trimită pe toate porturile sale (comportament de HUB). În momentul în care PC2 îi răspunde lui PC1, switch-ul vede că pe portul 1 îi vine frame-uri care are în antet adresa MAC al lui PC2. Astfel, a doua oară când se trimit informații de la PC1 la PC2, se va trimit doar lui, nu tuturor calculatoarelor.

## Rețele cu topologie stea extinsă



- în porturile din switch-ul central pot fi conectate și alte device-uri, nu numai alte switch-uri
- toată rețeaua (și cea cu topologie stea, și cea cu topologie BUS) reprezintă o rețea locală – LAN (Local Area Network)
- în spatele unui port din switch-ul central se găsesc echipamente cu mai multe MAC-uri

## Switch cu management

- are un terminal, o consolă
- își poate da o adresă IP
- se pot configura porturile, ca ele să meargă la anumite viteze, sau să fie conectate calculatoare cu o anumită adresă MAC
- au o interfață web, pentru administrare

**LAN** (Local Area Network) – de obicei acoperă o organizație, o locație, de dimensiune “rezonabilă”

**MAN** (Metropolitan Area Network) – construită conectând mai multe rețele locale prin intermediul unor echipamente care conectează undeva mai sus de nivelul Data-Link

**WAN** (Wide Area Network) – toată rețeaua unui provider la nivel național

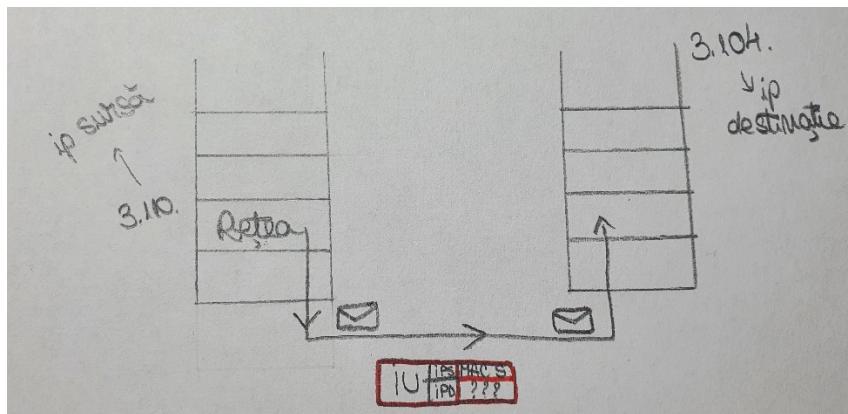
## FF:FF:FF:FF:FF:FF

- MAC de broadcast
- util când vrem să comunicăm cu toate echipamentele de la nivelul unui LAN
- în situații în care expeditorul vrea să trimită frame-uri intenționat la toate echipamentele din rețea → se setează ca *MAC destinație* acest MAC de broadcast

## Protocolele ARP și RARP

Două entități care vor să comunice se identifică prin nume, la nivel Aplicație. La nivel mai jos, ele se identifică prin IP (Rețea) și port (Transport). La nivel Data-Link, ele se identifică prin adrese MAC.

ping 192.168.3.104 – operează la nivel Rețea (merge printr-un protocol special, numit ICMP, undeva între Transport și Rețea)



**ARP (Address Resolution Protocol)** – ajută la determinarea adresei MAC pe baza adresei IP a destinației

Ca să determine adresa MAC a destinației, calculatorul trimite un frame special (**ARP request**) în care pune MAC-ul sursă, și la MAC destinație pune MAC-ul de broadcast. Acest request ajunge la toate calculatoarele și roagă calculatorul cu adresa IP la care vrem să ajungem să “spună” MAC-ul. Calculatorul va răspunde cu un **ARP reply**.

*arp -a* – arată perechile (ip, adresă mac) pe care le cunoaște calculatorul

**Default Gateway** = adresa IP a unei plăci de rețea din router; practic este adresa IP a router-ului căruia trebuie să îi trimit datele ca acestea să părăsească rețeaua locală și să meargă mai departe în Internet.

### RARP – Reverse Address Resolution Protocol

- se cunoaște adresa MAC, dar nu cea IP
- află adresa IP pe baza adresei MAC
- nu se mai folosește, a fost înlocuit de DHCP (**Dynamic Host Configuration Protocol**) – poate oferi mai multe setări de configurare

RARP este perfect pentru un device care tocmai a fost pornit, nu are configurat o adresă IP, și întreabă pe toată lumea din rețea printr-un **RARP request**, folosind adresa MAC de broadcast, ce adresă IP poate folosi.

### Ce setări *minimale* trebuie făcute unui device pentru a putea comunica în Internet?

- IP
- Netmask (subnet mask)
- Default Gateway (GW)
- DNS (adresa IP a cel puțin unui server DNS)

### RARP vs. DHCP

RARP poate da doar adresă IP. DHCP poate configura toate cele patru setări.

!!! Serverul DHCP este un proces care rulează pe router (cel mai natural, pentru că router-ul e mereu *up and running*).

!!! Serverul DHCP nu își poate da adresă IP lui însuși.

## CURS 8

### Placă de rețea vs. interfață

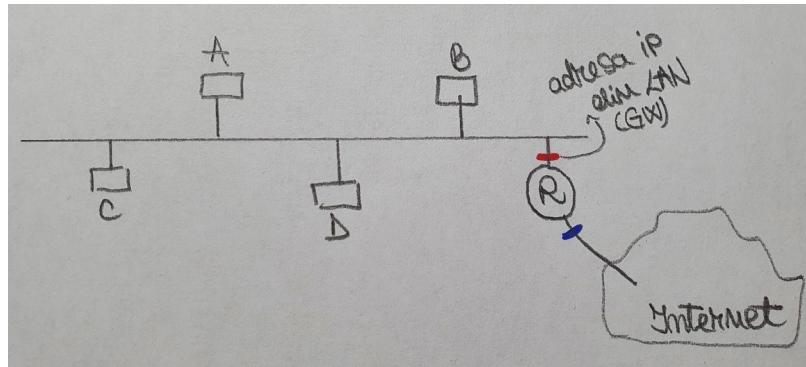
Placa de rețea este un device fizic prin care calculatorul se conectează cu exteriorul. Pe o placă fizică de rețea se pot pune mai multe adrese IP. Aceste adrese IP se numesc interfețe.

*ipconfig /release* – îi spune serverului DHCP că device-ul nu mai are nevoie de adresa IP asignată

*ipconfig /renew* – cere o adresă IP nouă

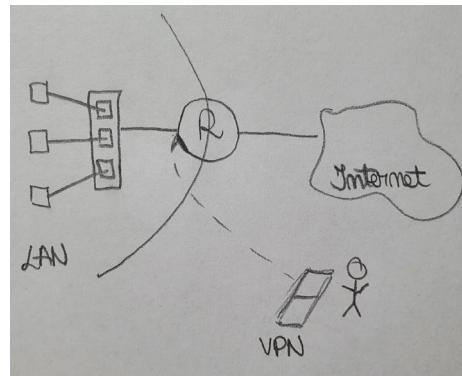
### ARP în LAN

De obicei, toate cererile ARP, RARP, DHCP se propagă numai la nivelul rețelei locale (nu trec dincolo de router).



Dacă A vrea să determine adresa MAC a lui B, va trimite niște cereri ARP. Acele cereri nu vor trece de router. Cererile RARP, la fel, se propagă numai la nivelul rețelei locale. (*proxy ARP* – anumite cereri ARP, RARP, DHCP pot trece dintr-o rețea în alta)

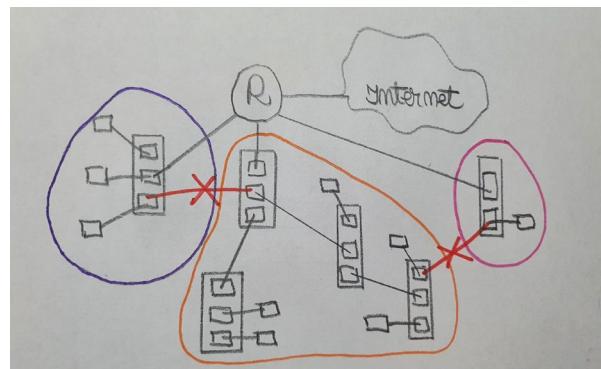
**VPN** – se introduce în serverul de VPN o placă de rețea virtuală (un driver pe placă de rețea care nu există)



router-ul devine un server de VPN odată ce este conectat un cablu virtual

**VLAN** – rețea locală construită pe o infrastructură fizică de rețea locală

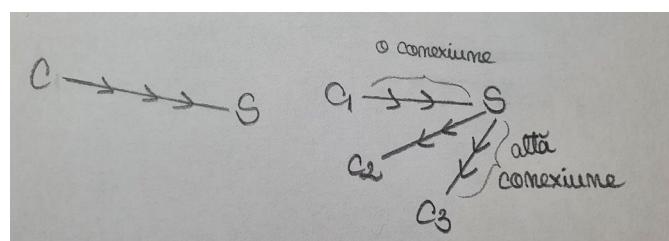
- ajută când rețeaua e prea mare, și trebuie spartă în sub-rețele



### Tipuri de trafic

Întotdeauna când circulă informație în Internet, datele respective aparțin unui anumit pattern de trafic.

- **Unicast** (cel mai folosit)
  - avem un emițător și un receptor (1 : 1)
  - aici intră conexiunile TCP clasice



- **Broadcast**
  - unu trimite și primesc toți (1 : toți)
  - cererile ARP, RARP, DHCP (se întâmplă peste UDP)
  - **nu** se poate desfășura peste TCP (nu putem avea un capăt al conexiunii să fie mai mulți care scriu pe același descriptor de socket)
- **Multicast**
  - unu trimite, dar nu primesc toți neapărat (1 : n)
  - dacă două personane dintr-un LAN intră pe un site, de la server-ul de streaming până la cele două persoane o să ajungă de două ori traficul, chiar dacă ceea ce se redă pe cele două device-uri este același lucru → are loc un trafic unicast între fiecare device și server
  - A.B.C.D – A este maxim 223 pentru ca adresa IP să fie validă (restul sunt rezervate pentru trafic multicast)
- **Anycast**
  - unu trimite, primește cel puțin 1 (1 : cel puțin 1)
  - dacă avem mai multe servere DHCP la nivelul rețelei locale, este important ca cel puțin 1 să răspundă cererii unui client, deci poate fi un pattern de comunicare de tip anycast

### **Cablu direct și cablu cross-over**

8 = 4 perechi de cabluri

#### **Cablu direct:**

- alb portocaliu
- portocaliu
- alb verde
- albastru
- alb albastru
- verde
- alb maro
- maro

#### **Cablu cross-over – se schimbă portocaliu cu verde:**

- alb verde
- verde

- alb portocaliu
- albastru
- alb albastru
- portocaliu
- alb maro
- maro

## CURS 9 + 10

192.168.3.x

x = 0..255

255.255.255.0 → 1111111.1111111.1111111.00000000

Octetii completați cu valoarea 1 exprimă faptul că toate adresele device-urilor noastre trebuie să aibă primii trei octeți egali.

192.168.3. - partea din adresa IP ce îmi identifică rețeaua

192.168.3.x - partea din adresa IP ce îmi identifică un calculator în cadrul rețelei

192.168.3.0 – adresă de rețea (nu se poate folosi) = adresa rețelei

.1 – folosită de obicei pentru router

...

.254

192.168.3.255 – adresă de broadcast (folosită de un device când vrea să transmită ceva tuturor echipamentelor din rețea)

**Netmask** = anulează diferențele dintre adresele IP ale calculatoarelor din cadrul aceleiași rețele.

**Adresa de rețea** → řI (AND, &, &&) pe biți între netmask și o adresă IP

192.168.3.x AND

255.255.255.0

---

192.168.3.0 - adresa de rețea (prima adresă IP din CLASĂ)

**Adresa de broadcast** → SAU (OR, |, ||) pe biți între *not(netmask)* și o adresă IP

192.168.3.x      OR

0. 0.0.255

-----  
192.168.3.255

**Cum se scrie o clasă de adrese?**      **adresă\_de\_rețea / netmask**

Adresa de rețea este întotdeauna prima adresă IP

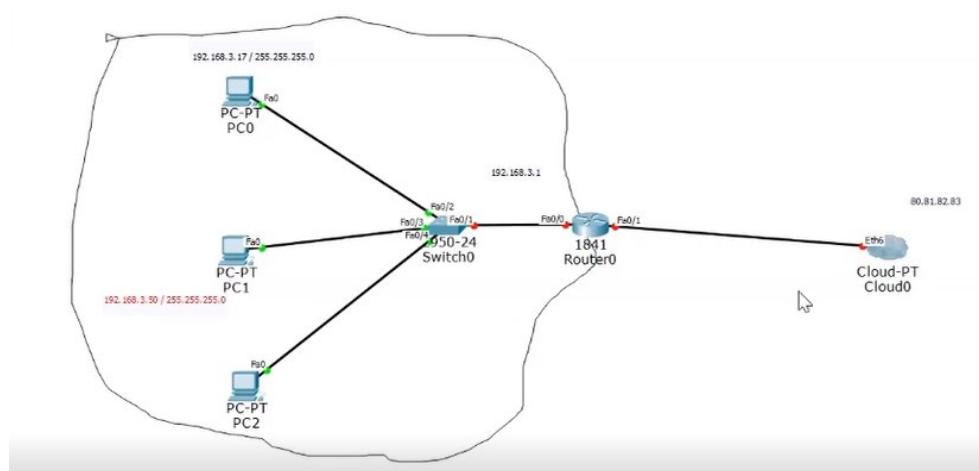
Netmask-ul spune că de la adresa de rețea în sus trebuie considerate toate adresele IP care au în comun primii x biți (x = numărul de biți 1 din netmask)

Biții 0 din netmask identifică cât de mare este clasa de adrese

**Clasă de adrese** = un interval (pool) de adrese IP cu anumite *proprietăți*:

- dimensiunea clasei e putere a lui 2
- adresa de rețea începe la multiplu de dimensiune a clasei

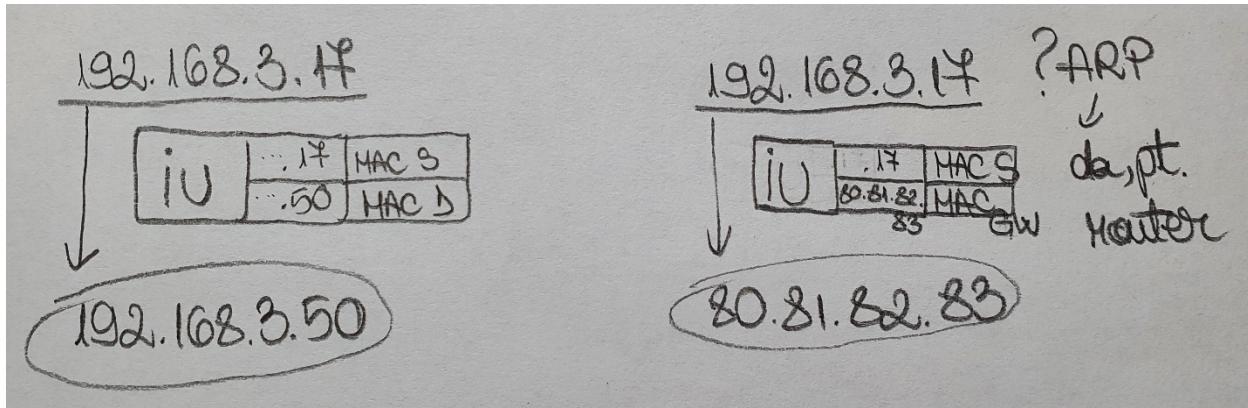
255.255.255.0 – 256 adrese IP



!!! Ca două calculatoare din aceeași rețea să comunice, **nu** e nevoie de router.

192.168.3.17 → 192.168.3.50: se face un ARP request în care întreabă adresa MAC a lui 192.168.3.50

192.168.3.17 → 80.81.82.83: calculatorul trimite mai întâi informația la 192.168.3.1



### De unde știe sursa că destinația nu se află în aceeași rețea locală?

Sursa întotdeauna face un řI pe biți între adresa IP a lui și netmask-ul lui. Apoi, face un řI pe biți între adresa IP a destinației și netmask-ul lui. Dacă rezultatele sunt egale, atunci IP sursă și IP destinație se află în aceeași rețea.

192.168.3.0 / 255.255.255.0

/24 - numărul de biți 1 din netmask

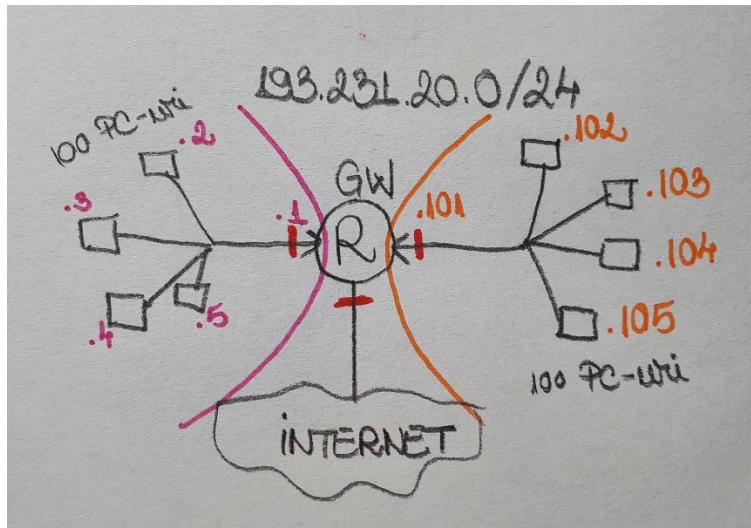
$$256 = 2^8 \quad (8 = 32 - 24)$$

172.30.0.0 / 255.255.0.0

/16

$$65536 = 2^{16} \quad (16 \text{ biți } 0 = 32 - 16)$$

!!! Într-un netmask, biții de 1 trebuie să fie consecutivi, de la stânga la dreapta a reprezentării.



greșit!!!

193.231.20.17 → 193.231.20.150

255.255.255.0      255.255.255.0

→ 193.231.20.0 = 193.231.20.0, deci din perspectiva sursei, destinația se află în aceeași rețea cu el, așa că face un ARP request să îi determine adresa MAC, și nu o să primească niciun răspuns (destination host unreachable)

Trebuie împărțită clasa:

125	.0	00000000
255.255.255.128	...	0xyzabcd
193.231.20	.127	01111111
125	.128	10000000
255.255.255.128	...	1.....
	.255	11111111

193.231.20.17 → 193.231.20.150  
255.255.255.128      255.255.255.128

---

193.231.20.0 != 193.231.20.128

Clasa de 256:

**193.231.20.0 / 24** → **193.231.20.0 ... 193.231.20.255**

193.231.20.0 / 255.255.255.**0**

*Nu se pot folosi:* 193.231.20.0, 193.231.20.255

---

Clase de 128:

**193.231.20.0 / 25** → **193.231.20.0 ... 193.231.20.127**

193.231.20.0 / 255.255.255.**128**

*Nu se pot folosi:* 193.231.20.0, 193.231.20.127

**193.231.20.128 / 25** → **193.231.20.128 ... 193.231.20.255**

193.231.20.128 / 255.255.255.**128**

*Nu se pot folosi:* 193.231.20.128, 193.231.20.255

---

Clase de 64:

**193.231.20.0 / 26** → **193.231.20.0 ... 193.231.20.63**

193.231.20.0 / 255.255.255.**192**

**193.231.20.64 / 26** → **193.231.20.0 ... 193.231.20.63**

193.231.20.64 / 255.255.255.**192**

**193.231.20.128 / 26** → 193.231.20.128 ... 193.231.20.191

193.231.20.128 / 255.255.255.**192**

**193.231.20.192 / 26** → 193.231.20.192 ... 193.231.20.255

193.231.20.192 / 255.255.255.**192**

*Se pierd 8 adrese IP în total*

Cele mai mici subclase: cele cu 4 adrese IP → netmask 255.255.255.**252** (/30)

193.231.20.145 face parte dintr-o clasă de 64 de adrese

64 adrese =  $2^6$  → netmask 255.255.255.192

193.231.20.145 AND

255.255.255.192

---

193.231.20.128 → deci face parte din clasa 193.231.20.128 / 255.255.255.192

**/32 255.255.255.255 – host only netmask** – mască de rețea care se folosește ca să denotă că mă refer la un singur IP, nu la clasa din care face parte

!!! Adresa de rețea e întotdeauna pară, cea de broadcast e impară

!!! Întotdeauna adresa de rețea e multiplu de dimensiunea clasei

193.231.20.32 -> începe clasa de 64 (/26)

193.231.20.32 ... 193.231.20.95

193.231.20.40      193.231.20.70

255.255.255.192      255.255.255.192

---

193.231.20.0      !=      193.231.20.64

## CURS 11

Se dau două adrese IP: 192.168.1.20 și 192.168.1.40. Care e rețeaua de dimensiune minimă care le conține pe amândouă?

R: Ambele adrese fac parte împreună din clasa 192.168.1.0/24. Problema e că rețeaua nu e de dimensiune minimă. Împărțim în două și ajungem la clasa 192.168.1.0/25. Mai împărțim și ajungem la 192.168.1.0/26.

193.231.20.0/24 (/255.255.255.0) – 256 adrese

193.231.21.0/24 (/255.255.255.0) – 256 adrese

500 calculatoare – pot să am o clasă suficient de mare care să conțină toate calculatoarele?

Putem face “join” celor două clase (**AGREGARE**), să avem o clasă cu 512 adrese IP.

193.231.20.x

193.231.21.x

**193.231.00010100.00000000**

...

**193.231.00010101.11111111**

Adresele au în comun primii 23 de biți → netmask = /23 (255.255.254.0)

la final 9 biți 0 →  $2^9 = 512$  adrese IP

⇒ clasa de adrese 193.231.20.0 / 255.255.254.0  
adresa de bcast: 193.231.21.255

Se dă adresa 193.231.21.0. Se poate folosi adresa respectivă pentru un device?  
**Depinde de netmask.**

!!! Nu putem zice de o adresă IP dacă este adresă de rețea, fără să știm netmask-ul.

193.231.19.0/24 – 256 adrese IP

193.231.20.0/24 – 256 adrese IP

---

? 193.231.19.0/23 - 512 adrese IP

193.231.19.0                    193.231.xxxxxxx1.00000000

---

=

---

512                                1000000000

Tăiem 8 zerouri de la numărător cu 8 de la numitor

193.231.xxxxxxx1

---

10

Deci nu e multiplu de 512, pentru că ne-a mai rămas un 0 la numitor pe care nu îl putem simplifica.

*De ce nu poate să înceapă o clasă de 64 la adresa de rețea 192.168.1.32? Pentru că 32 nu e multiplu de 64.*

*Avem adresa IP 193.231.20.0. Care e cea mai mare clasă de adrese care are această adresă ca și adresă de rețea?*

Cea mai mică clasă: 193.231.20.0 / 255.255.255.252

193.231.20.0 / 255.255.255.248

193.231.20.0 / 255.255.255.240

---

193.231.20.0 / 255.255.255.0

193.231.20.0 / 255.255.254.0                    193.231.21.255 – bcast

? de 1024 adrese: 20 de multiplu de 4 ( $2^2$ ) – descompunerea lui în baza 2 se termină cu 2 biți 0 – adresa IP se termină cu 10 biți 0 – se împarte cu  $2^{10}$  – multiplu de  $2^{10}$  – putem avea clasă de 1024 adrese IP care începe cu 193.231.20.0

⇒ **193.231.20.0/22**

*Precizați clasa de dimensiune minimă care conține următoarele două adrese IP: 80.81.82.83, 84.79.30.129.*

Trebuie să determinăm un netmask care, făcând și pe biți între prima adresă IP și netmask și a doua adresă IP și netmask, să ne dea aceeași adresă de rețea.

0.0.0.0 / 0.0.0.0 – are ca și dimensiune  $2^{32}$  adrese IP

80.81.82.83 = 01010000.81.82.83

84.79.30.129 = 01010100.79.30.129

-----

netmask: 11111000.0.0.0 = 248.0.0.0 – punem 1 pe pozițiile unde se aseamănă biții celor două adrese IP

Acum trebuie să calculăm adresa de rețea:

01010000.81.82.83 AND

11111000.0.0.0

-----

01010000.0.0.0 = 80.0.0.0 = rețea

bcast: 87.255.255.255

R: **80.0.0.0 / 5**

### Clase A, B, sau C

193.168.0.1 / 255.255.255.0 / 24 – clasă C ( $2^8$  adrese IP)

172.30.0.0 / 255.255.0.0 / 16 – clasă B ( $2^{16}$  adrese IP)

10.0.0.0 / 255.0.0.0 / 8 – clasă A ( $2^{24}$  adrese IP)

0.x.y.x	128.x.y.z	192.x.y.z	224.0.0.0
127.x.y.x	191.x.y.x	223.x.y.z	.....
-----	-----	-----	-----
clase de tip A (/8)	clase de tip B (/16)	clase de tip C (/24)	clase D (de multicast)

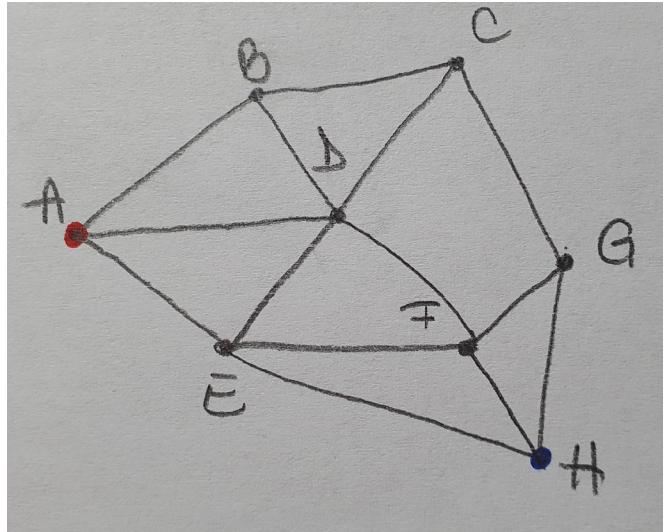
Să se reprezinte intervalul [ 79.85.23.48 ... 94.16.25.63 ] în clase de adrese.

Plecând de la 79.85.23.48 ca adresă de rețea și până la 94.16.25.63 care e adresă de broadcast, să exprimăm tot intervalul în clase de adrese (și numărul de clase să fie minim).

- 79.85.23.48 / 255.255.255.240 / 28      bcast: 79.85.23.63 → maxim 16 adrese pentru că 48 e multiplu de 16
  - 79.85.23.64 / 255.255.255.192 / 26      bcast: 79.85.23.127
  - 79.85.23.128 / 255.255.255.128 / 25      bcast: 79.85.23.255
  - 79.85.24.0 / 255.255.248.0 / 21      bcast: 79.85.31.255
  - 79.85.32.0 / 255.255.224.0 / 19      bcast: 79.85.63.255  
 $32 * 256 = 2^5 * 2^8 = 2^{13}$
  - 79.85.64.0 / 255.255.255.192.0 / 18      bcast: 79.85.127.255
  - 79.85.128.0 / 255.255.128.0 / 17      bcast: 79.85.255.255  
 $128 * 256 = 2^7 * 2^8 = 2^{15}$
  - 79.86.0.0 / 255.254.0.0 / 15      bcast: 79.87.255.255
  - 79.88.0.0 / 255.248.0.0 / 13      bcast: 79.95.255.255
- ...

## CURS 12

Internetul e organizat prin interconectarea mai multor rețele.



Dacă nodul A vrea să trimită la nodul H, ruta ar fi  $A \rightarrow E \rightarrow H$ .

**Reguli de dirijare** = informații care îi spun unei surse la ce vecin trebuie să trimită informația ca să ajungă la destinație

Se păstrează o **tabelă de dirijare**

Tabela de dirijare (de rută) a nodului A:

Destinație	Vecin	Lungimea drumului
H	E	2
A	A	0
B	B	1
C	D B	2
D	D	1
E	E	1
F	D D	2
G	D E	3

Nu e bine să suprasolicităm o linie de comunicație (un nod).

Din perspectiva lui A, câte noduri în graf avem, atâtea rute avem în tabelă.

În rețeaua Internet, există niște **algoritmi de dirijare** care permit construirea unei tabele de dirijare în mod *dinamic*.

Există două clase mari de algoritmi de dirijare:

- bazați pe vectori distanță
- bazați pe starea legăturilor

Ca și *metrici de cost*, se află:

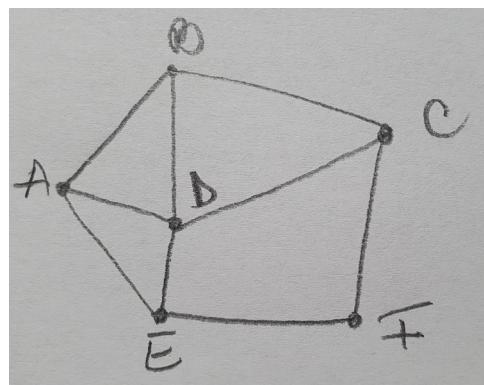
- lungimea
- viteza liniilor de comunicații
- delay-ul liniilor de comunicații
- costul (\$\$)
- lungimea cozilor de așteptare existente la nivelul routerelor

Când într-un router cu  $n$  linii de comunicație intră foarte multe date, se poate ajunge la situația în care vin date la router pe  $n - 1$  linii, toate datele trebuie trimise pe router pe o singură linie de comunicație. Ca să facă față, routerele țin o coadă de așteptare în care se pun, temporar, pachetele care vin pe liniile de comunicații.

Cu cât coada este mai plină, cu atât linia de ieșire asociată cozii e mai congestionață (aglomerată).

Când se umple coada de așteptare, pachetele sunt “*aruncate la gunoi*” și nu vor fi livrate la destinație.

### Algoritmi bazati pe vector distanță



Fiecare nod își populează tabela de dirijare cu ce știe. După ce un nod își pune ruta spre el însuși, două routere adiționale încep și schimbă tabelele de dirijare. De exemplu, A și B își trimit unul celuilalt propria tabelă de înregistrare. Astfel, A află că B ajunge la B cu un drum de lungime 0, și B învață același lucru despre A. Se întâmplă asta cu toți vecinii.

Fiecare router pleacă de la o tabelă minimală și ajung să aibă informații despre cât mai multe destinații și își populează tabela de dirijare cu tot mai multe înregistrări.

!!! Protocoale de dirijare care implementează acest algoritm: **RIP** (Routing Internet Protocol), **BGP** (Border Gateway Protocol – protocolul router-ului de frontieră).

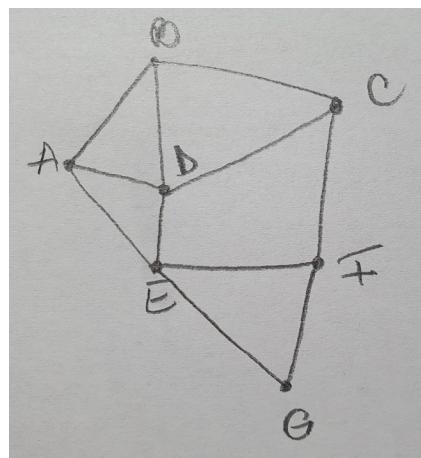
### Avantaje:

- se pot implementa în rețele mari
- fiecare nod, făcând niște calcule parțiale, și bazându-și determinarea tabelei de dirijare pe niște calcule făcute de vecini, fiecare nod nu are de făcut atâtea calcule, algoritmul fiind **mai puțin intens** computațional

### Dezavantaj:

- apare **problema numărării la infinit**

### Algoritmi bazați pe starea legăturilor (se bazează în spate pe Dijkstra)



Fiecare router din rețea pregătește niște pachetele: (**sursă, destinație, cost, timestamp**). Acest pachetel descrie starea legăturii dintre sursă și destinație: la momentul de timp *timestamp*, era linie de cost *cost* de la sursă la destinație.

Aceste pachete sunt trimise de la sursă la fiecare vecin. Astfel, vecinii află starea legăturilor adiacente ale sursei.

La fel, fiecare vecin trimite pachetele respective trimise de sursă, prin tehnica de *inundare*, la vecinii lor.

În final se ajunge ca fiecare nod să aibă informații despre starea legăturilor adiacente tuturor celorlalți noduri din graf.

### **Avantaj:**

- nu mai apare problema numărării la infinit

### **Dezavantaj:**

- este mai intens computațional, fiecare nod face calculele individuale și pentru el
- un astfel de algoritm nu poate fi implementat și deploy-at într-o rețea de dimensiuni mari

În Internet, se face dirijarea pe două nivale:

- la nivel macro, rețeaua fiind o rețea de rețele – se face dirijare folosind *BGP*
- la nivelul unui sistem autonom (AS – Autonomous System) – se pot utiliza algoritmi de dirijare *IGRP* (Interior Gateway Routing Protocols) – se pot baza pe algoritmii cu testarea legăturilor

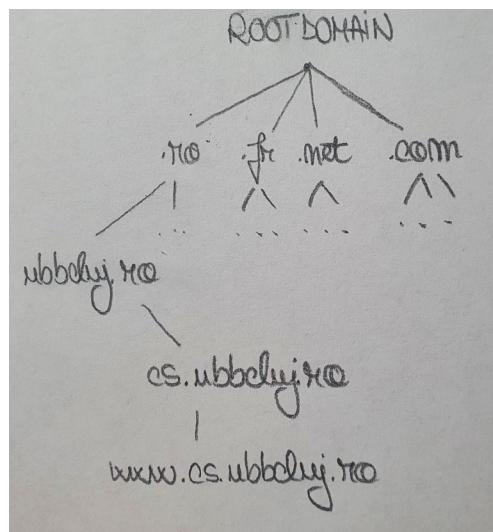
**Propagarea rutei** = un router spune că el “*tine în spate*” unele clase de adrese, iar această informație se propagă de-a lungul rețelei

## CURS 13

### Server DNS (Domain Name System)

Responsabil să traducă nume de calculatoare în adrese IP.

Tot sistemul numelor de domeniu poate fi privit ca o **bază de date distribuită cu structură arborescentă**.



www.cs.ubbcluj.ro

- ubbcluj.ro – domeniu, subdomeniu al domeniului .ro (**TLD – Top Level Domain**)
- cs.ubbcluj.ro – subdomeniu
- www.cs.ubbcluj.ro – nume de device din cadrul subdomeniului cs.ubbcluj.ro

Browser-ul, ca și client, trebuie să se conecteze la adresa IP al lui www.cs.ubbcluj.ro. Ca să facă connect-ul, browser-ul trebuie să îi determine adresa.

*ping www.abc.ro* – obține adresa IP a lui www.abc.ro

www.cs.ubbcluj.ro e tradus într-un alt string: cs.ubbcluj.ro – acel string e tradus în adresă IP

## Două tipuri de TLD-uri:

- country-based: .ro, .hu, .fr, .it etc.
- generice: .com, .org, .net

Există ceva și mai sus de TLD-uri: un domeniu generic . – **root domain**

⇒ www.cs.ubbcluj.ro.

Browser-ul, ca să afle adresa IP a site-ului, trebuie să apeleze la sistemul de operare care rulează pe calculator, care are setat un DNS oferit de provider. Sistemul de operare întreabă DNS-ul provider-ului adresa IP a site-ului, dar acesta nu știe cine e site-ul. Așa că întreabă alte servere DNS.

**!!!** Orice domeniu din internet (orice subdomeniu) trebuie să aibă un server DNS responsabil de domeniul respectiv (trebuie să aibă un server DNS care să cunoască informații despre domeniul respectiv).

Servelele DNS responsabile pentru root domain (13 la număr) țin minte ce servere DNS sunt responsabile de domeniile copil (de TLD-uri).

Servelele DNS responsabile de TLD-uri țin minte ce servere DNS sunt responsabile de copii lor.

...

La final de tot, serverul DNS al cs.ubbcluj.ro ține minte că www.cs.ubbcluj.ro are o anumită adresă IP.

## Interogările care se pot face unui server DNS

- **iterative** – interogările care le face DNS-ul provider-ului către restul serverelor DNS
- **recursive** – interogarea pe care o face clientul către DNS-ul provider-ului

*De ce serverele DNS responsabile de root domain nu sunt interogate recursiv?*

**Dacă ele ar fi interogate recursiv, ar trebui să țină minte starea interogării. Tot traficul DNS din Internet ar trece prin serverele DNS responsabile de root domain, iar ele nu ar face față.**

### **Tipuri de înregistrări care se țin minte de către un server DNS:**

- **A** (address): www.cs.ubbcluj.ro → 193.0.225.34
- **AAAA** (adrese IPv6)
- **NS**: un server DNS de la nivel superior “memorează” serverul DNS responsabil de un subdomeniu
- **MX**: servere de mail responsabile de un anumit domeniu (user@yahoo.com)
- **PTR** (pointer): 193.0.225.34 → www.cs.ubbcluj.ro
- **CNAME** (alias-uri de tip string):  
www.cs.ubbcluj.ro -----CNAME-----> cs.ubbcluj.ro  
cs.ubbcluj.ro -----A-----> 193.0.225.34
- **TXT** (text): se poate memora orice, spre exemplu adresa fizică unde e localizat un server, numărul de telefon al adminului, etc.

Un server DNS are două **roluri** bine-definite, cu o diferență fină între ele:

- răspunde la interogări care vin dintr-o zonă limitată de utilizatori; cererile de rezolvare pot fi legate de orice → rol de server DNS clasic (cele de la provideri)
- rol de server DNS responsabil de un anumit domeniu (cele TLD, root domain); aceste servere pot fi interogate de oricine, dar interogările trebuie să fie foarte punctuale → rol de Name Server (NS)

**!!!** Toate interogările, toată comunicarea între serverele DNS, se face pe portul 53 peste UDP.

Toate răspunsurile pe care le primește server-ul DNS al provider-ului și clientul de la care pleacă întrebarea sunt însotite de un câmp **TTL** (Time To Live) – timp de viață. TTL-ul se exprimă în secunde, de obicei. Acest câmp spune calculatorului că

informația respectivă este valabilă un anume timp. Calculatorul o să cache-uiască informația primită timp de cât este specificat în acel TTL.

Acest lucru se întâmplă ca să nu se tot trimită interogări când accesezi un site (de exemplu).

După ce trece perioada de timp, browser-ul uită cine e acel site, și o să întrebe din nou DNS-ul.

Și serverul DNS ține minte cine e site-ul, nu numai calculatorul. Astfel că, dacă un device face o interogare pentru un site, iar apoi alt device face aceeași interogare, DNS-ul o să trimită direct cine e acel site, însă TTL va specifica timpul rămas de când a trimis informația primului device.

Ce trebuie făcut ca să existe site-ul *măseluțaveselă.ro*?

E nevoie de domeniu. Ca să existe domeniu, trebuie plătit ca, la nivelul TLD-ului .ro să se pună o înregistrare de tipul NS cum că serverul DNS responsabil pentru domeniul *măseluțaveselă.ro* este blabla. Serverul DNS trebuie să țină minte printr-o înregistrare de tipul A care e adresa IP. (DNS hosting)

!!! E nevoie ca pe un calculator cu adresă IP reală să ruleze un server DNS care să fie responsabil de domeniul *măseluțaveselă.ro* și care să țină minte toate înregistrările care au de-a face cu domeniul.

Pe lângă DNS hosting, mai este nevoie de WEB hosting. Trebuie să se țină minte paginile web ale site-ului.

!!! HTTP și HTTPS merg peste TCP, pe porturile 80, respectiv 443

8.8.8.8 – DNS pe care îl oferă Google, care poate să îl întrebe oricine legat de orice (îl oferă ca să ne facă profiling)

*ipconfig /displaydns* – arată toate înregistrările pe care le știe calculatorul

*ipconfig /flushdns* – golește cache-ul

*nslookup* – folosită pe post de client DNS pentru debugging

*De ce nu merge www.ferarri.it dacă îl cauți în browser, dar serverul DNS zice că există?*

**E trecut în DNS, dar pe calculatorul cu adresa IP a lui nu există niciun server WEB care să ruleze pe 80 sau pe 443.**

*De unde știe serverul DNS de la provider care e punctul de plecare al interogărilor în arbore?*

**Orice server DNS are un fișier de configurare unde sunt trecute toate serverele DNS responsabile pentru root domain.**

## CURS 14

Propagarea rutelor spre clasele de adrese e specifică claselor de adrese IP reale.

Două categorii de clase IP:

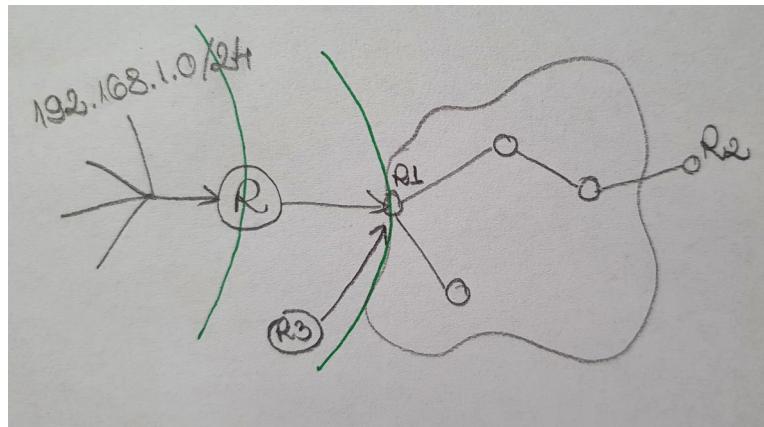
- reale (routabile, publice)
- false (non routabile, private)

### Clasele IP false:

- **10.0.0.0 / 255.0.0.0 (/8)**
    - 1 clasă falsă A cu  $2^{24}$  adrese IP
  - **172.16.0.0 / 255.255.0.0 (/16)**  
**172.17.0.0 / 255.255.0.0**  
...  
**172.31.0.0 / 255.255.0.0**
    - 16 clase false B cu  $2^{16}$  adrese IP fiecare
- ⇒ 172.16.0.0/12 (agregare)
- **192.168.0.0 / 255.255.255.0 (/24)**  
**192.168.1.0 / 255.255.255.0**  
...  
**192.168.255.0 / 255.255.255.0**
    - 256 clase false C cu  $2^8$  adrese IP fiecare
- ⇒ 192.168.0.0/16 (agregare)

Router-ul nu o să spună niciodată router-ului vecin că el ține în spate clasa de adrese false. Informația despre clasa respectivă nu se propagă în Internet din două motive:

- router-ul nu anunță că ține în spate această clasă
- clasa mai e folosită și în alte colțuri din Internet



Clasa falsă, care inițial e vizibilă numai în rețeaua locală deservită de R. Dacă ar exista pe router-ul R1 o rută cum că spre clasa falsă se ajunge prin R, clasa s-ar propaga și ar fi vizibilă în toată rețeaua provider-ului respectiv.

Dacă 192.168.1.7 vrea să trimită un pachet în Internet, de exemplu la 80.81.82.83, va trimite pachetul cu IP sursă 192.168.1.7 și IP destinație 80.81.82.83 la router-ul de la provider (R1). El se uită în tabela de dirijare să vadă unde trebuie să trimită mai departe pachetul.

Când ajunge pachetul la 80.81.82.83, el trebuie să trimită un răspuns, aşa că acum IP sursă devine 80.81.82.83, iar IP destinație devine 192.168.1.7. Router-ul nu va fi în stare să livreze pachetul, din două motive:

- ruta unde e localizat 192.168.1.7 nu e propagată în Internet
- orice router din Internet va fi dat peste cap, pentru că o astfel de clasă va fi folosită în foarte multe locuri

În mod normal, dacă R1 nu ar face *SNAT* (Source Network Address Translation), pe calculatoarele din rețeaua locală deservită de R1 nu merge netul. Ele pot comunica ele între ele, dar nu cu exteriorul. Motivul nu este pentru că nu ar putea trimite pachete, ci pentru că destinația nu ar fi în stare să răspundă unor astfel de pachete.

Când un router din Internet primește un pachet care are ca și adresă IP sursă o adresă falsă, router-ul aruncă pachetul la gunoi.

## (S)NAT

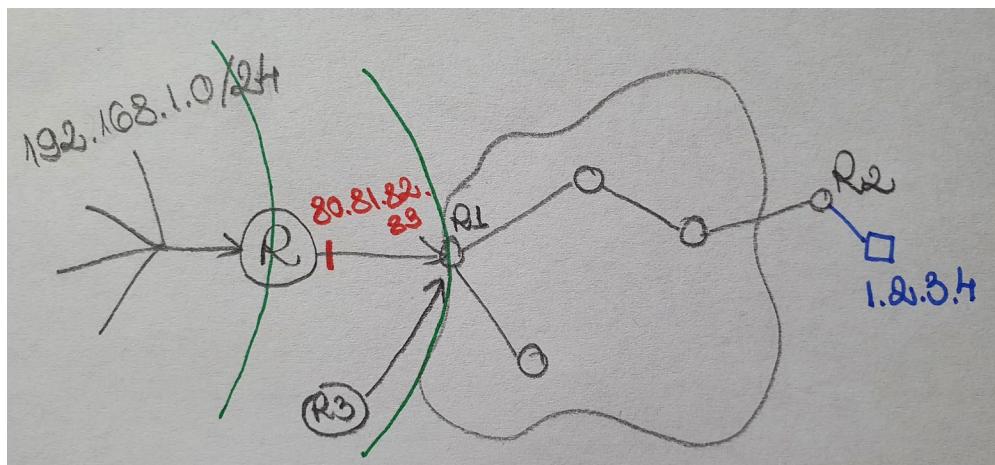
Întotdeauna când un pachet e expediat din rețeaua locală în exterior, adresa IP *sursă* a pachetului (adresă IP falsă) este **înlocuită cu adresa IP publică pe care o are routerul pe interfața externă** (interfață de Internet/de WAN).

Mecanismul se află, în stiva TCP/IP, la nivel *Rețea*.

**Legături PPP** (Point to Point Protocol) – folosită pentru conexiunile unde în spatele unui router este sigur 1 device; se pune netmask 255.255.255.255 (/32)

**PPPoE** (PPP Over Ethernet) – are loc un fel de VPN între router-ul de acasă și router-ul de la provider (e un fir virtual unde e doar calculatorul în spatele lui)

Router-ul păstrează o **tabelă NAT** în care ține minte toate conexiunile care tranzitează un anumit pachet. Atunci când vine un răspund din Internet la router, router-ul o să știe la care din device-urile din rețeaua locală trebuie să transmită răspunsul.



**Tabela NAT**

IP Sursă	IP extern router pt SNAT	IP Destinație
192.168.1.7:21322	80.81.82.83:21322	1.2.3.4:80
192.168.1.15:12323	80.81.82.83:12323	1.2.3.4:80

În momentul în care un router trebuie să translateze înapoi adresa IP destinație a pachetului, o să se folosească inclusiv de portul sursă și portul destinație.

Când un pachet merge de la o sursă spre o destinație, pachetul conexiunii are un port sursă (random printre cele libere). Serverul destinație nu o să știe că conexiunile sunt inițiate de două calculatoare diferite din rețeaua locală. El le vede venind de la aceeași adresă IP: 80.81.82.83. Dar pentru că porturile sunt diferite, nu îl încurcă cu nimic.

Pentru că device-urile sunt independente, cu sisteme de operare independente, se poate întâmpla ca, atunci când se creează conexiunile, ambele calculatoare să meargă pe port cu aceeași valoare.

IP Sursă	IP extern router pt SNAT	IP Destinație
192.168.1.7: <b>25000</b>	80.81.82.83: <b>25000</b>	1.2.3.4:80
192.168.1.15: <b>25000</b>	80.81.82.83: <b>25000</b>	1.2.3.4:80

Problema e că, dacă pachetele curg în exterior cu același port, nu se mai poate face diferențierea între ele, nici la nivelul server-ului care primește pachetele, nici la nivelul router-ului.

În momentul în care router-ul detectează că vine o conexiune nouă, care în momentul în care se face SNAT-ul, dacă conexiunea arată identic cu o altă conexiune pe care o are, trebuie să aibă loc o **translatare de port** sursă.

**!!!** Deci SNAT, în sine, nu este exclusiv la nivel Rețea. Când se face translatarea de port, se bagă și la nivel *Transport*

**!!!** Este posibil să trebuiască să se facă translatare de port chiar dacă nu există o conexiune identică cu cea curentă. Asta se întâmplă când portul folosește același port când inițiază o conexiune client spre aceeași destinație.

### **DNAT (Destination Network Address Translation)**

Toate pachetele care vin din exterior spre 80.81.82.83, router-ul înlocuiește adresa IP destinație și trimit pachetele respective în adresa locală căruia îi erau destinate.

Ex: Dacă din exterior vrem să accesăm un proces server care rulează în rețeaua locală pe un calculator cu adresă IP privată, trebuie spus router-ului, că orice vine din

exterior, specificând pe ce port și dacă e TCP/UDP, trebuie trimis înăuntru la calculatorul potrivit.

Tot aşa, trebuie să se facă diferențierea pe porturi diferite.

### **Avantaje și dezavantaje ale claselor de adrese IP private:**

#### **Avantaje:**

- permit economia de clase de adrese IP reale
- securitatea

#### **Dezavantaje:**

- e nevoie de SNAT ca să meargă internetul
- nu se pot rula servere pe ele care să fie accesibile din alte părți din Internet fără să se facă DNAT

În momentul în care un router face SNAT, se poate să nu facă cu o singură adresă IP reală pe care o are router-ul, ci poate cu un pool de adrese IP (de ex. de la 80.81.82.83 la 80.81.82.90). Această chestie e utilă când avem aşa de multe calculatoare în rețeaua locală și se stabilesc aşa de multe conexiuni, încât e nevoie să se facă translatare a porturilor sursă.

Acest pool nu trebuie neapărat să fie configurat pe interfața exterioară a router-ului. E important doar, ca pe următorul router să existe o rută cum că spre clasa din care fac parte IP-urile de la 83 la 90 se ajunge prin acel prim router.

**!!!** E o percepție greșită că NAT-ul înlocuiește adrese IP false cu adrese IP reale. Se poate înlocui și fals cu fals, real cu real etc.