19.10.2022

Cursul 3

Client —— $IP_{server} : PORT_{server}$ ——> Server

netstat → afișează să vedem porturile libere

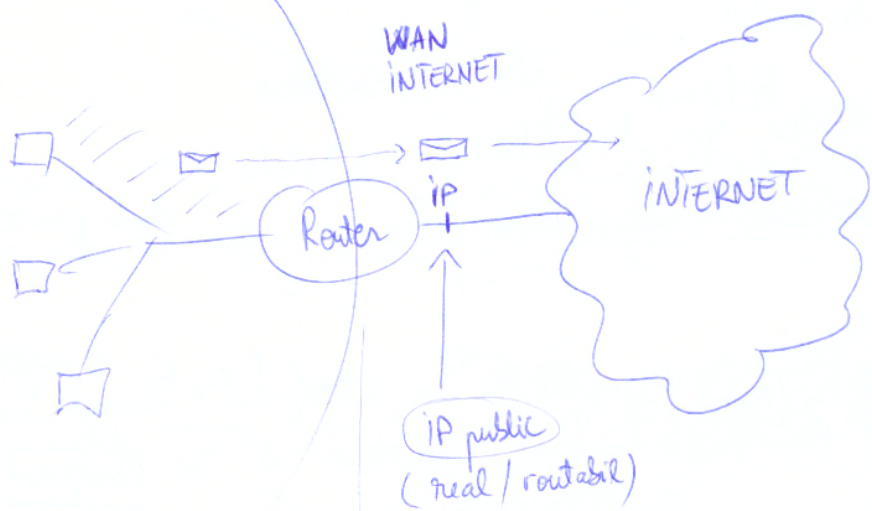
! Trebuie verificat dacă s-a făcut:
- connect
- bind

SS - pe Linux mai nou

Local Area Network

ipconfig - Windows
ifconfig - Linux
ip addr

LAN ← adrese IP private (false) (nonroutable)



192.168.1.X
192.168.0.X
172.30.X.Y
diverse clase de adrese

! Obs: între 2 sau mai multe device-uri din LAN nu este necesar routerul

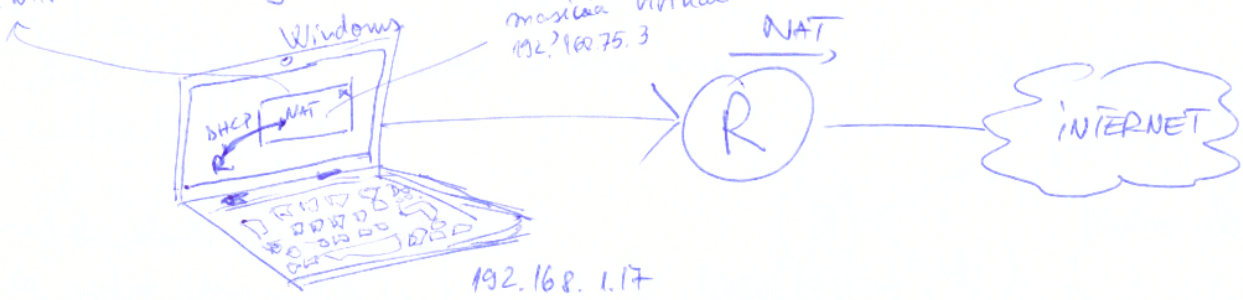
NAT Network Address Translation
SNAT Source —u—

→ dacă merge ping între două IP-uri, va reuși conexiunea

IP C: 192.168.1.17
IP S: 192.168.1.37

Nu e neapărat să fie în aceeași rețea, doar observăm că fac parte din aceeași clasă de adrese IP.

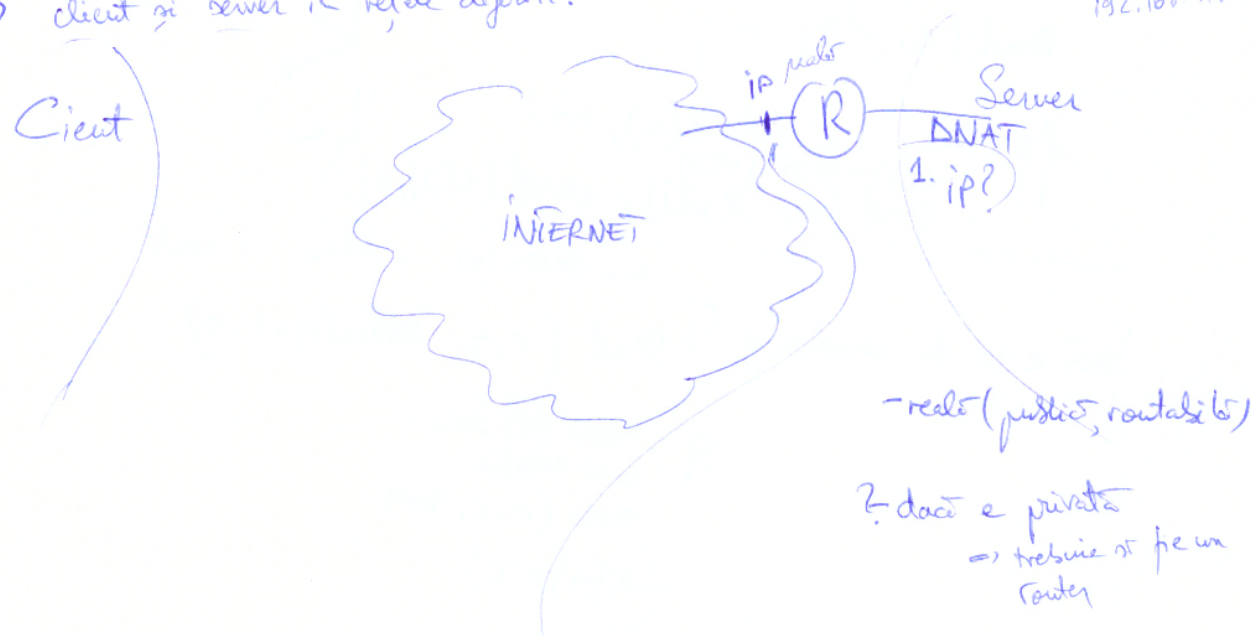
Modif. NAT cu modul Bridge



- Mașina fizică „joacă rolul de router" pentru mașina virtuală
- Dacă avem modul Bridge pt. mașina virtuală (nu NAT, ca mai sus), mașina fizică nu mai e router pt. aceasta, căci însuși Routerul (R) îi va furniza adresă IP. (forma 192.168.1.X)
- Nu este obligatoriu ca serverul DHCP să ruleze pe router.

(când un device vrea să se conecteze la un router/server, trimite un mesaj de tip broadcast)

→ client și server în rețele diferite:



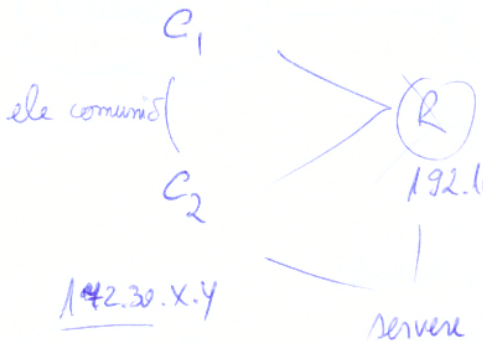
- rețea (publică, routabilă)

? dacă e privată
⇒ trebuie să fie un router

dacă cineva dorește să se ~~conecteze~~ la adresa ip a serverului ~~și dă redirect la adresa privată a serverului~~ trebuie să specifice adresa IP reală a routerului (extern) → clientul

DNAT
Port forwarding
Virtual Server

172.30.4.5 → nu va avea acces la internet !



! Obs: Adresele IP ale device-urilor dintr-o rețea trebuie să aparțină de cea a router-ului.

C → S

```
uint16_t n;
n = htons(n)
send(c, &n, ...
```

```
uint16_t *x;
x = malloc(...)
send(c, x, n*sizeof(uint16_t), 0)
```

```
for(i=0; i < ntohs(n); i++)
    x[i] = htons(x[i])
    send(c, &x[i], sizeof(x[i]), 0);
```

n = strlen(x) → htons → send

Dacă e șir de caractere: C) send(c, x, strlen(x)+1, 0)

```
S( recv(c, &n, ..
   n = ntohs
   recv(c, x, n+1
   (x[n] = 0)
```

! Important: câte date trimitem, atâtea citim + de același tip de date (important pt. size)

ex:

```
int c;
char S[100];
uint16_t i, j;
int x;
c = socket ; connect --
scanf( "%hu", &i )
```

↓ se citesc 2 octeți

(hu - short unsigned)

```
// scanf( "%d", &i )
```

! → pune la adresa lui i 4 octeți, însă i are doar 2 octeți

⇒ se scriu 4 octeți care pot afecta alte date / variabile

strace / ltrace

./strace ./client