12.10.2022

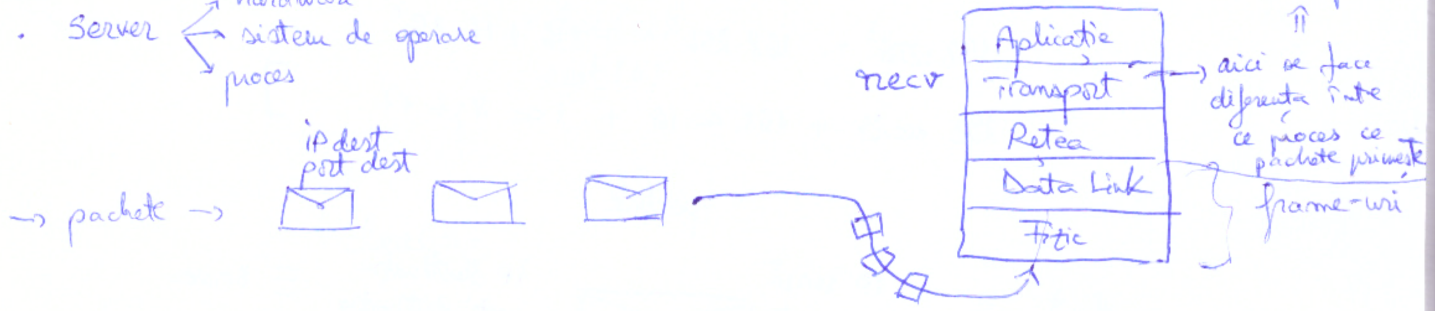
Cursul 2

CLIENT ——————————→ SERVER

ip, portul serverului

↓ identificarea procesului cu care se comunică

· Server ⟨ hardware / sistem de operare / proces



<u>CLIENT</u>

c = socket(,)

↳ e vorba de protocoalele:

⟨ Transport — TCP
Retea — IPv4

SERVER

s = socket(,)

→ pagina urmatoare

```
                  - tip structură *
if ( connect(c, {- ip server }, sizeof(structură)) < 0)
                {- port server }                      ↓
                       ↓                             esuează
             sub forma unei structuri

             * tip structură : AF_INET
```

```
send(c, &i, sizeof(i), 0)
recv(c, &j, sizeof(j), MSG_WAITALL)
close(c)
```

SERVER

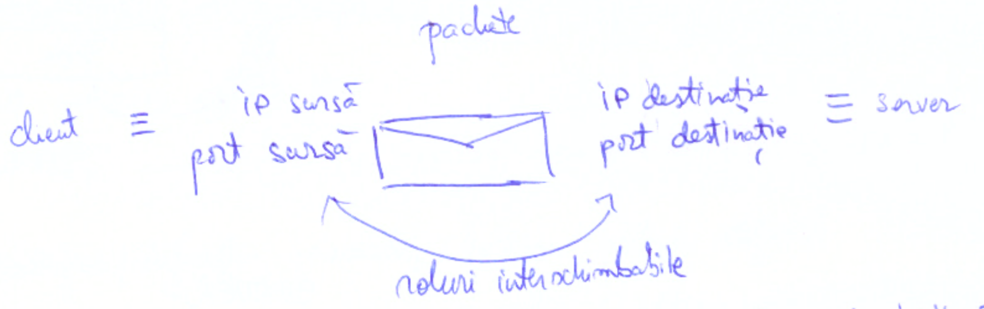
```
S = socket ( , )                              ↱ 0.0.0.0

if( bind (s, { - INADDR_ANY
               - port         } , sizeof(structură)) < 0)

listen (S, 5)
```

aceasta îi conferă atributul de server ←

["192.168.1.17"   inet_addr
                                inet_ntoa

$192 \cdot 256^3 + 168 \cdot 256^2 + 1 \cdot 256^1 + 17 \cdot 256^0$

$192 \ll 24 + 168 \ll 16 + 1 \ll 8 + 17$]



⇒ socket-urile sunt bidirecționale

→ pentru a verifica ce porturi sunt ocupate : netstat

bind → la client ~~este~~ este opțional (în general, nu se face), pentru că portul clientului nu este important și, de asemenea, dacă s-ar face bind la client există riscul de a ajunge pe un port deja ocupat

```
while (1) {
    c' = accept ( s, { - ip client
                       - port client } , "sizeof"
    recv ( c'
    send ( c'
    close ( c')
}   close (S)
```

deservirea clientului respectiv

} un server iterativ

! Obs: ordinea pt. send și recv, semantica + alte detalii / reguli de comunicare → RFC

→ transformarea serverului iterativ în server concurent:

```
while (1)
{
        c' = accept
        if( fork() == 0)
        {
                recv
                send
                close(c')
                exit(0)
        }
}
close(s)
```

înainte de trimitere

htons

htonl

ntohs

ntohl

după primire

fie send la client → send (c, &i, sizeof(i), 0)

$\hookrightarrow$ $\boxed{E8 | 03 | 00 | 00}$

recv la server

} este necesar ca arhitecturile să fie aceleași (dif. little endian vs. big endian)

- datele, în rețea, circulă în format big endian