

26.10.2022

Cursul 4

TCP < protocol orientat pe conexiune
reliable

Socket-uri UDP

Emițător
(client)

Receptor
(server) (roluri "mai"
egale; interschimbabile)

c = socket (AF_INET, SOCK_DGRAM

s = socket (AF_INET, SOCK_DGRAM)

- * UDP - nu se poate face deosebire între cazurile când
 - = se pierd date de la E la R
 - = nu există un proces cu care să se comunice
 - = se pierde rezultatul / răspunsul
- * 64 Kb - dimensiunea maximă a unui packet
(MTU - Maximum Transfer Unit)

Situație UDP (neverificate, ca în cazul TCP) < pierdere de date
gim. datele, neordonate
(out of order)

↓
este mai importantă rapiditatea
(modelul interogare-răspuns)

~~connect~~

sendto (c, de unde
din memorie, cat, flg,

structura, sizeof(structura)
- ip dest
- port dest

bind (s,
~~accept~~ ~~listen~~

recvfrom (s, unde, cat,
flg, structura, sizeof(structura))
- ip emițător
- port emițător

recvfrom (

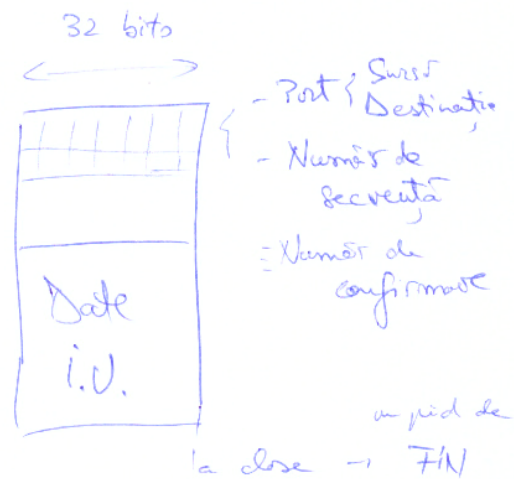
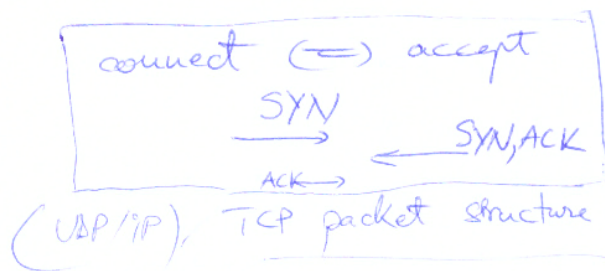
close (c)

Sendto (

* sendto trimite datele unui
partener ale cărei date sunt
de identificare
completate de recvfrom

(Dacă fac serverul concurrent, după recvfrom generezi un port
random (pt. identificare), faci un fiu, iar "comunicarea" se face
pe baza portului nou generat.)

TCP:



~> pachete de control -> fără date
(ex: se trimite pt. a confirma că încă
are loc conexiunea și ce primire
date din sens invers)

-> confirmările nu se fac instant (se supraglomerează fiul de conexiune / rețeaua
cu pachete goale / de control)
↓
se confirmă calea de date

ex:

C = socket (AF_INET, SOCK_STREAM)

connect (c) --

close (c)

C = socket (AF_INET, SOCK_DGRAM)

"reci la seara" lui c -> c nu e în
acelasi timp
socket TCP
și UDP

Sendto (C, 6000

recvfrom (C

Sendto (C, 6001

← 6001

bind (S, 6000

recvfrom (S, 6000

if (pack == 0) {

bind (S', 6001

recv

Sendto (S, 6001

}