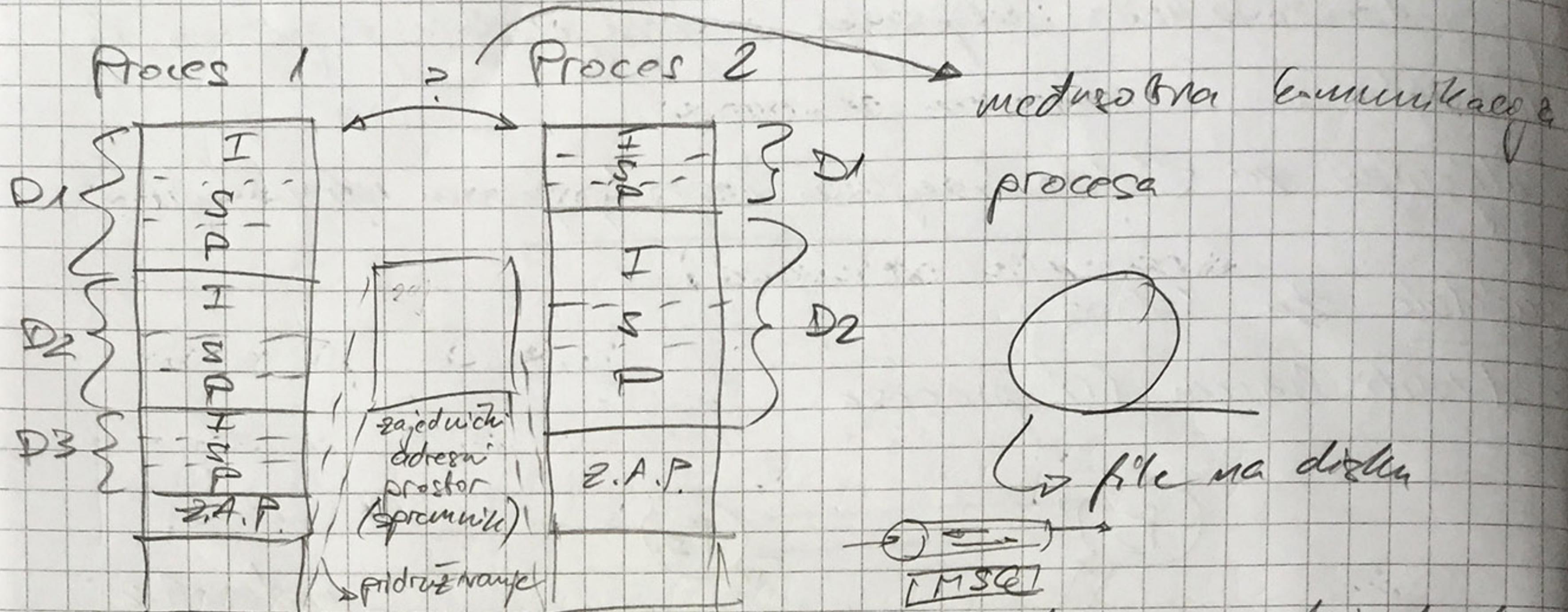


## 10. KOMUNIKACIJA između procesa



- $D_i \rightarrow$  skupni podprostori, podijeljeni na tri dijela
  - I - instrukcije
  - S - stog
  - P - podaci
- komunikacija između dva istog procesa je neovisna
  - sustav (a samim time i vrsta)
  - struktura podataka binarnog scmfata, kernel (jezgra) ...
  - zaledničke varijable Peterson, Lamport... (zastavice); slično - zaledničke strukture podataka)
- zalednički adresni prostor unutar procesa
- međusobna komunikacija
  - stvara se zalednički adresni prostor (uredljivo OS-u)
    - dodjeli rezerviraj
  - funkcije sl.moth, sl.mqget (na slici gore)
  - alternativno otvaranjem zajednog datoteka (jedan proces otvara, drugi piše)
    - pretežno se koristi gotovi mehanizam
  - varijable okoline (nije učinivo OS-a)

- varijable deklaracije
- red poruka (message queue, MQ)
- jevovodi i menovani jevovodi
  - imenovanje jevovoda konzistente datotečku (pravom rednjacem ne može da uveliki jevovoda)
- Windows podžaraju datotečku koja se pojavljuje zapadnicu spremnike (prijevod 10.1. u ožujku)
- potrebam je mehanizam slanja i prihvatanja poruka
  - nema mreže postopati sinchronizacijom mehanizmom (npr. poslušni monitor, primjer G.S. klijenta)

Kommunikacija porukama / jevovodi - pravila za rješenje

→ Kommunikacija porukama

- poruka - "mala" količina podataka (malo - do nekoliko stotina B) koja služi za komunikaciju među procesima
  - poruka se ne salje između procesa, već se salje u red poruka (message queue)
  - može imati tip; nije upravid određen
  - proces koji ima odgovarajuće dozvole (za čitanje, pisanje i sl.) može poslati poruku
  - proces može biti izbrisany - u tom slučaju samo određenog tipa
- red poruka se može iskoristiti kao semafor (samo OSEN)
  - startanje u red = postavi semafor
  - izlazanje iz reda = očekji semafor
  - semafori (BSEN/OS/OSEN; BSEN - dinarni OS - Dijkstra opći semafor, reprezentativno praktički se ne

Koristi: u praksi OSEN - brojčki semafor, brojčki opći semafor)

- ako je red poruka prazan  $\equiv$  semafor je nepratjan (crven)

- zahtijeva čekanje od pristupne poruci

- funkcije u C-u:

| int msgget (key t\_key, int flags);

- vraca -1 za gresku drugo je ID reda  
poruka (int podatak)

- t-key  $\rightarrow$  ključ za pristup redu poruka

- procesi moraju imati zajednički key

zadaje se  
prihvatuje  
stvaranja  
reda poruka | - može biti greska ako više korisnika radi  
na istom računalu

- konkretno, na raspodjeljenom sustavu,  
samo prvi korisnik s određenim, ključem  
neće imati problema

- standardna uporaba

| if (msgid = msgget (ključ, 0600 | IPC\_CREAT) == -1)  
| priši greska.

meja struktura

- sljedeća funkcija:

| int msgsnd (int msgid, struct msghdr \*mstr,  
| int msgsz, int flg);

- msgsz - message size  $\rightarrow$  maxima; kada je veća  
definirana u strukturi; n+1 zbroj kojeg je  
reda

- flags - prava prijetja 0600, a-user, g-group, o-other  
 "7" → IPC-CREATE
- u - 6 → 2 vrata 11  
 r w
- 0600  
 ↳ dezvoljana prijetja drugačije?
- IPC-CREATE - tako red poruka s tim argumentom postavlja ID; inace smrši vrati ID)

```

struct msgbuf {
    long mtype;
    char mtext[1];
};
```

uz malova (prodovljene  
dubine)

- ali je red pun blokiran ce se msgsnd
- IPC-NOWAIT - tako pride dobro, ako ne  
nikom nista (zastavica)
- int msgrcv (int msgid, struct msgbuf \*mstr, int  
msgsz, int msgtype, int flg);  
velicina n+1
- ordje msgsz označava uapređen moguću poruku,  
prekoračenje vraca -1 (odnosi se na char mtext[n])
- msgrcv - primanje poruke (message receive)

| int msgctl (int msgid, int cmd, struct msgid \*buf);

- cmd - komanda
- IPC-RMD - brišeju reda poruka s određenim  
ID-om

## → Okolina (environment)

- okolina (environment) - už znakova "ime = vrijednost" koji se predaje svakom programu prilikom pokretanja
  - ime → ime variable okoline
  - u neprisanom pravilu se piše velikim slovima (npr. PATH)
- korisnička linija:
  - | → print status
  - | PS
- korisnička linija imao može biti sh, csh, bash, zsh, ksh, tcsh ...
- bash - dolra zbog historyja (uz stranicu gore dole, moguće je raditi prethodne upitnice unose) i zbog načina rada s datotekom (prihvata prvič slova i pristupačna tipka TAB)
- linije se odnose na komunikaciju varijablama okoline
- služi projektu razvijati okoline programme (ali programima); ovde će znati varijable okoline koje su prilikom pokretanja
- npr. varijabla okoline KUTAK koja se zahteva prosledjuje sljedećim procedurama
- dva tipa korisničkih linija - sh i csh (ostale pripadaju tim stupinama)

	sh	csh
ime = vrijednost	set <u>ime=vrijednost</u>	
export ime		setenv <u>ime vrijednost</u>
export		setenv

varijabla samo  
za liniju  
predaja programa  
popis

## → Prvstup v programu obolice

| int main();

| int main (int argc, char \*argv[], char \*envp[]);

- argc - argument counter, argv - argument vector,  
envp - niz znakova

- envp - popis svih variabli koje se podesju  
programima

- operacije se envp ne mogu; razlozi:

- nije vidljivim funkcijama main i drugo

- koriste se gotove funkcije!

| char \*getenv (char \*name);

- vraca pokazivac na prvi znak imena = vrijednost

| int putenv (char \*niz);

- niz → ime = vrijednost; ta varijable mora

postojati, možda se može i stvoriti (projekti)

- služi za komunikaciju između procesa kada se imaju

čitavu dodjeliti isti objekti

## → Cjevovodi (pipelines)

- široka uporaba; problematika je shvatiti logiku i  
posadini, dobro projekt

▀ kako povezati procese cjevovodom?

1.

(P1)

( )

→ → →

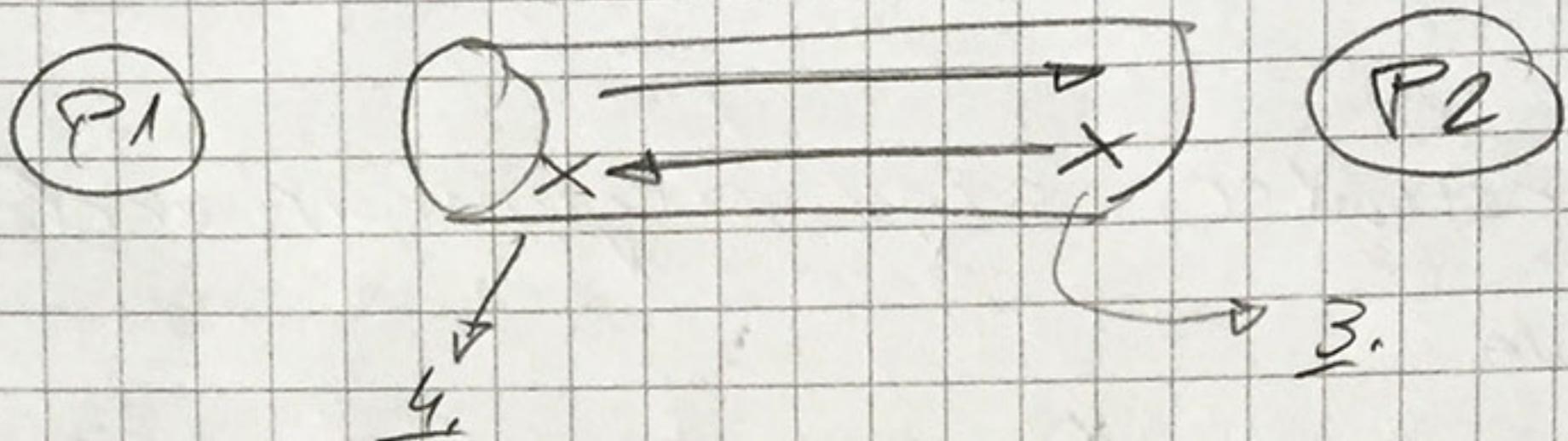
- proces P1 stvara cjevovod

→ →

- poruke mogu teći u oba smjera, ali u praksi se to ne radi (tj. koristi se samo jedan)

2.

- stvoriti proces dijete (koje će da)



3.

- dogovoriti smjer (zatvoriti smjer za pisanje)

4.

- zatvoriti broj čvorova da piše kod P1

5.

- komunikacija

- čvorod je opisan s brojem krajeva (u ovom primjeru 2)

- svaka dijete vidi svoju opciju (ili otvara ili piše)

- zaista prvo čvorod, a tek onda procese koji ga koriste

- stvaranje procesa funkcijom fork → proces dijete je kopija roditelja i dolazi sre što on ima (nije iz prakse)

- ovaj tip čvoroda se ne može realizovati, ali se može implementirati čvorodima

**P** Razlika između čvoroda i datoteka

- datoteka se može citati proizvoljnim redoslijedom, čvorodi nisu predefinirani

- prazan čvorod se može iskoristiti kao sinkronizacijski mehanizam

- pun oporovod zahvjetača čitanje
- datoteku uči
- čitanje je prevozda je destruktivno (odnosno pročitani podatak se tamo više ne nalazi)
- datoteka je dan opisuta; prevoz 2
- učestvatale prevozda:
  - procesi kopir komunikaciju prevozom mrežu biti u sredstvu; nema komunikacije
  - komandna linija

```
| ls | wc
|     word counter je dobiti broj
|     riječi - broj znakova ; linija
iziskuje sadržaj direktorija
| - koristi se prevoz
```

- za oštvarivanje prevozda OS koristi modus premuka (kopi je nečitne stranice, davaš ujčeće 4 KB)
  - to je jedno redateljstvo prevozda, odnosno 4 KB
- stvaranje prevozda

| int pipe (int fd[2]); → dva opisnika!

- fd[0] - izlazna strana
- fd[1] - ulazna strana
  - pišanje u fd[1] je pišanje u prevoz
  - čitanje iz fd[0] je čitanje iz prevoza
  - recipročitet između tim varijabli za P1; P2 je prvi put

→ (also si 25.3. (prije fijdan))

- od 10 do 13 sati, doći uani god
- uploadom (also se ujedno dolaze u određeni termin)
- repozitorije (neodređenosti) u zadatku - rečiti kako god
  - koliko redova poruka treba? - uklju god, kolike god
- 3 procesa, 6 redova poruka
  - siaklizacija Campart, Ricart, Agrawala
  - Campart - redi prvi poruka, drugi način vod prema krajem godine
- prvi zadatci
  - ne radiju u distribuiranom okružju (čaku i sami)
  - ipak, izvodi se distribuirani algoritam zbroj mogućnosti izvodenja da jednom samim
  - instalacija - Wubi vodi uo image file

| int pipe (int fd[2]);

| | ↳ fd[1] - ulazna strana, fd[0] - izlazna strana

- zatvara se generodar

| int close (int fd);

- return -1 → zatvara se neotvorenu operativu
- veli su opisnicu defaultno otvarati

| int dup (int fd);

- duplikacija opisnika generodara

- smjer: radi duplikat, vraca red u kojem opisnik prije slobodni je i novi dopisnik će biti upotrebljen

- jedan se zatvori i konsti se duplikat - ovim
- tričkom se povezuje na standard input/output
- ls/wc → ispisuje na std::out sadržaj direktorija; očekujemo da se povezuju dva procesa gresodom (iz komande linije)
- može : programski: po potrebi se prenosi - rava sa std::out/std::in na gresku
- taj ladije one godine ne postoji
- fdu:
  - 0 → std::ulaz (std::in)
  - 1 → std::izlaz (std::out)
  - 2 → std::izlaz\_greske (std::err)
  - ne mogući one vrijednosti 0, 1 & 2 su uvećane od fd[1] i fd[0]!

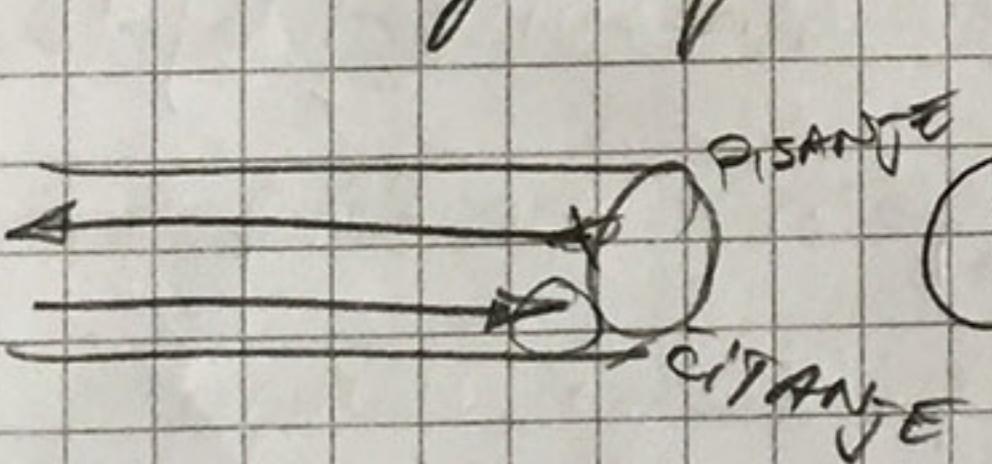
- preporuka:

#define CITANJE 0

#define PISANJE 1

pa krozeti te vrijednosti za fdu (fifile) program

→ Povezivanje greskova na standardni ulaz



postojeći procesi se ne mogu povezati greskom; on mora biti definiran prilikom stvaranja procesa

- redstavljaju objekt greskova
- želimo povezati standardni ulaz na gresku (želimo dati te gresku):
  - | close (fd[PISANJE])

close (STD-IN)

- zatvara se lucrat pisanje

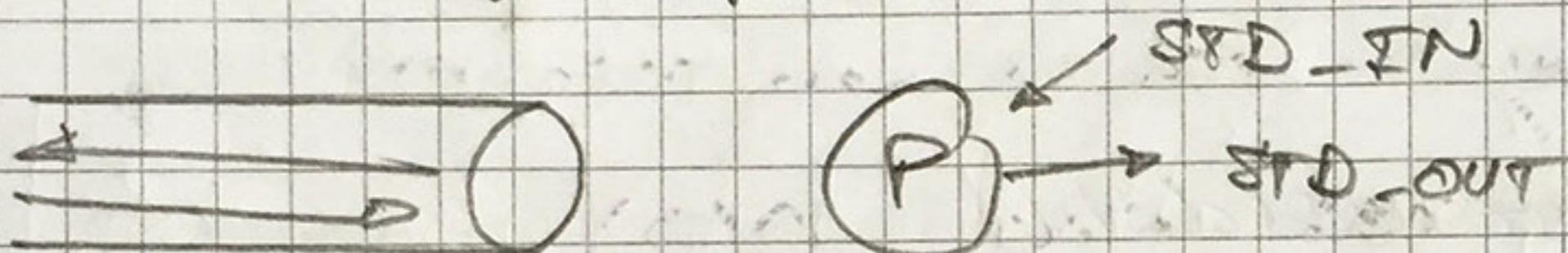
| dup (fd [CITANTE]);

- duplicit se sugesta sa userito 0

| close (fd [CITANTE]);

- isto radi i (swc, ab radi i drugo

→ Povestivane cjevova da na standardni izlaz



- Zatvara -jelaz i sluzi u userito STD-OUT:

| close (fd [CITANTE]);

close (STD-OUT);

dup (fd [PISANTE]);

close (fd [PISANTE]);

- povestivane na stdout/stdin se izvodi u dva odvojena programa

## IMENOVANI ČEBOVOD

- kombinacija čebovoda i datoteka
- čebovod podesiti neće biti etauca u slučaju iniciranja čebovoda (tako će CRT biti obrišao) - za to se koristi datoteka ~~kas obrišu, ali to nije~~
- "obrana" datoteka - označa P na licenčnim
- relacije O
- otvara se dvaput - redom za crtanje, drugi put za pisanje - tek naku ova otvaranje jedan proces može početi pisati
- pipe (čebovod) se mora stvoriti programski, za radnik .D tog se datoteka stvara iz komandne linije
- podaci su privremeni u datoteci - zato relacija O (između jednog procesa zapise drugi proces)
- nema ograničenje podataka (preko veličine stranice) - ograničenje je diskovski prostor
- prednosti:
  1. Rad s analizom čebovodom odgovara radu s datotekom (iste funkcije)
  2. Procesi koji komuniciraju preko im. glos. ne moraju biti u skrodbitu → nevezni
  3. Nema ograničenja na duljinu (možd obrišu do 10 stranica, ≈ 4000)
  4. Može se skoristiti iz komandne linije sa mikrofona

- ukljucde je novedba ili funkcija

| int mknode (char \*path, int mode, int dev);

    -- 1 greska, 0 u redu

    - path - ime

    - mode - S\_IFIFO + 9 bitova za prava  
        pristupa

    - dev - ne konsti se

    - ukljucde je rezerviran za odmiva; zastavice  
        omogucuju pristup drugima

| int open (char \*path, int flags)

    - otvara se datoteka; -1 je error, 0 je  
        u redu (ucljude se uklazi u manualu)

    - flags :

        - O\_RDONLY - samo citanje

        - O\_WRONLY

        - O\_RDWR

| int fcntl (int fd, int cmd, int arg);

    - kontrola nad otvarcim izvorom; dinamička promjena prava pristupa; decje dan  
        predati

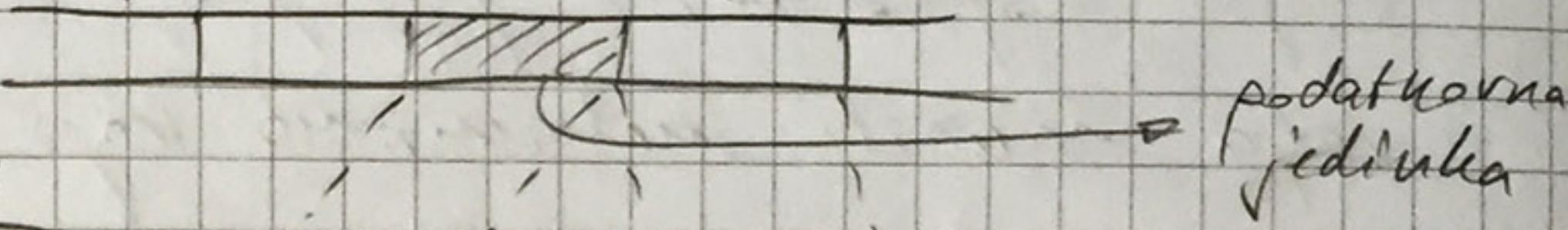
    - citanje i pisanje se izvodi kao ; za normalnu  
        datoteku

- komunikacija između procesa na uređaji - protokolu slog  
= četiri sloja

1. Primenjenska razina (Application layer)
2. TCP - prijenosni protokol (Transmission Control Protocol)

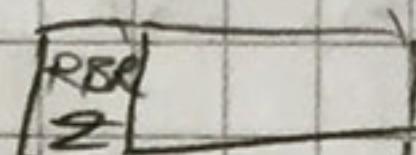
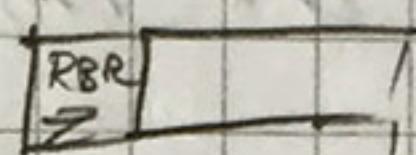
upr. datoteka na iskodistu

1. PRIMJENSKA RAZINA



2. TCP

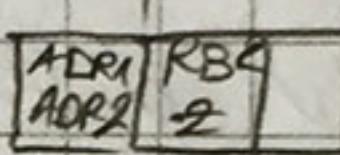
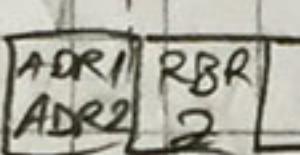
redni broj



podatkovna jedinica

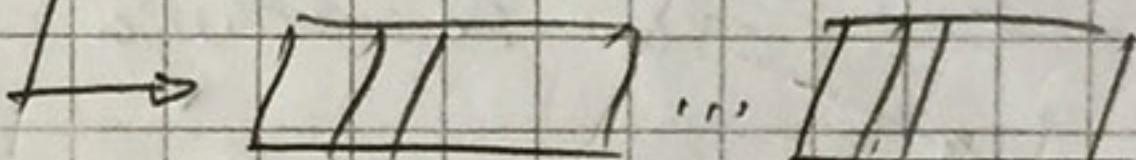
3. IP

zadnjim dijelom



ADR1 - adresa iskodista  
ADR2 - adresa odredista

4. FIZIČKA RAZINA

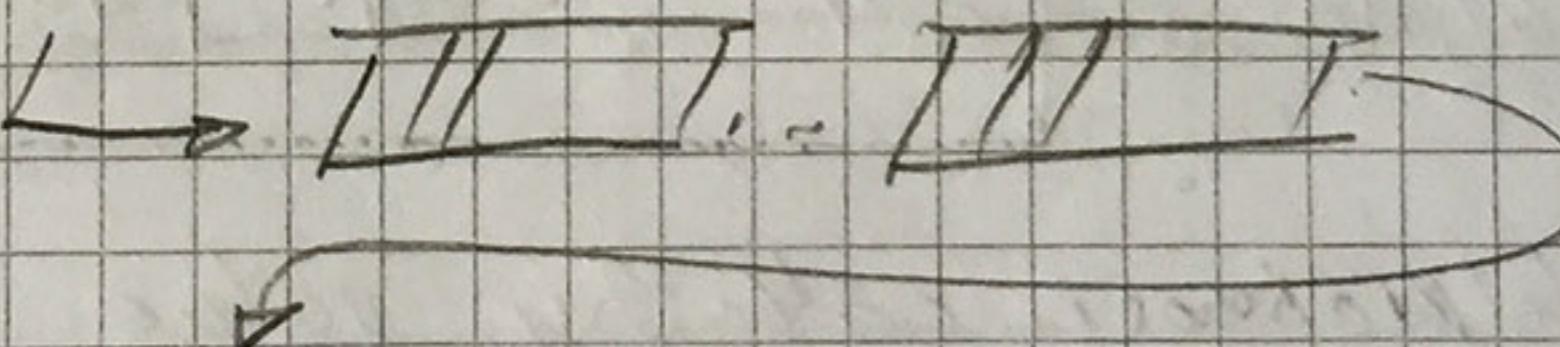


3. IP - Internet Protocol

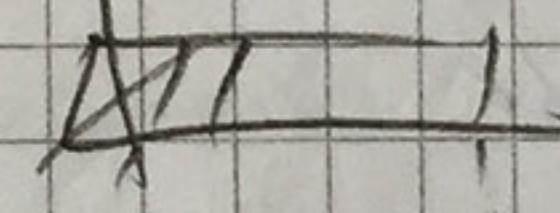
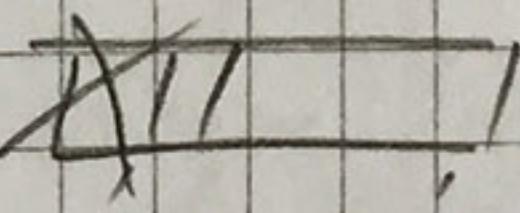
- u osnovi nije imao nikakvu zaštitu (nije bilo takvih zaključeva)

4. Fizička razina

4. Fizička razina

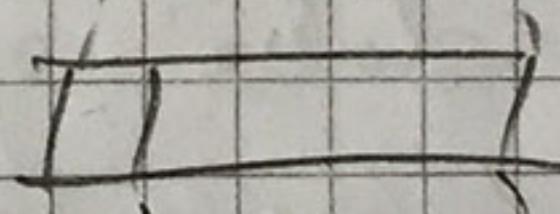
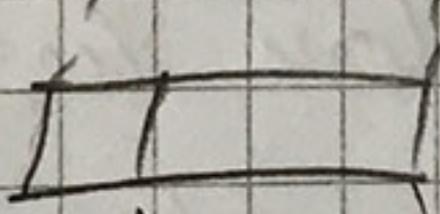


3. IP

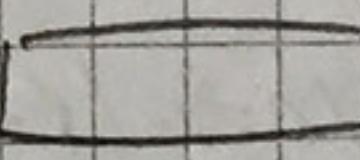


(primarna podataka)

2. TCP



1. PRIMJENSKA RAZINA



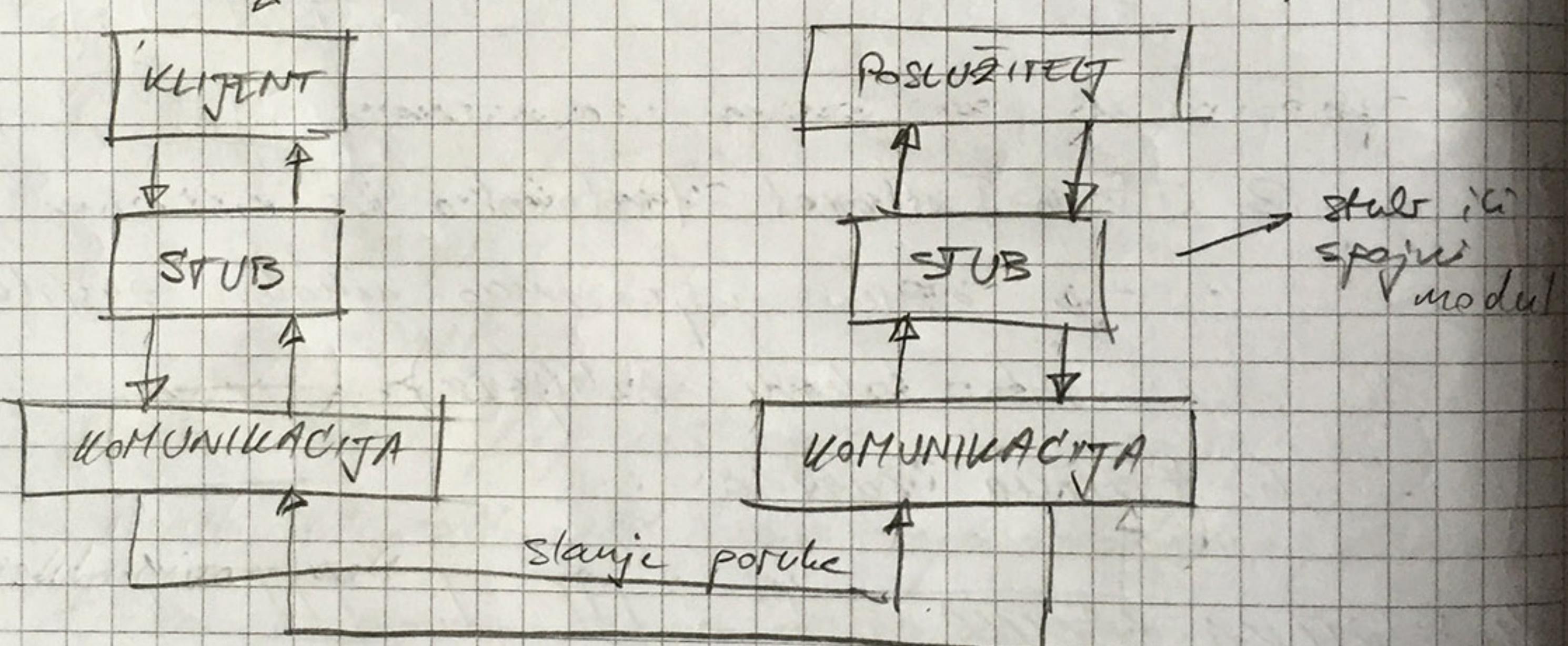
## 10.2. Komunikacija između procesa u raspodjeljenim sustavima

- osnovni model je da prethodnoj: stranici
- program je na projektu, razinu ne mora brinuti o drugim razdrama - aktiviraju se gotovim API funkcijama
- valja uspostaviti poveznicu (socket)
- valja naučiti mehanizme koji se koriste
  - pouča se slično kao komunikacija gospodom

→ Mechanizmi za komunikaciju

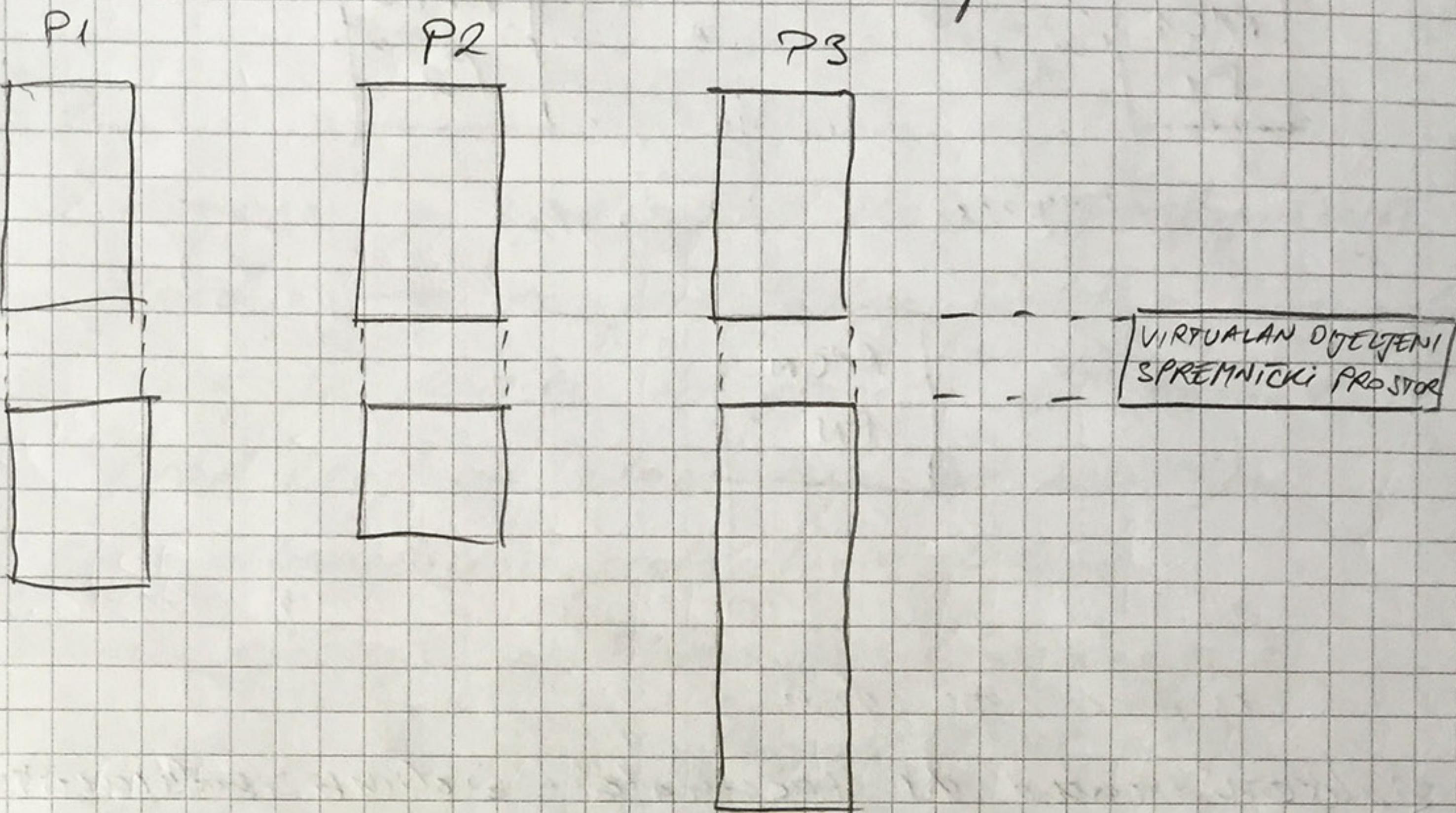
- 1. RPC - poziv udaljene procedura (Remote Procedure Call)

tu se odvija problem, poziva se fja s poslužitelja



- isključivo call by value (call by address nema smisla)
- počeo stvara se ustanovi komunikacije sa svim protokolom predaje poslužitelju
  - rezultat funkcije se prenosiće stubu i vrada se klijentu
- klijent preko stuba dohvata raspoređenu vrijednost koju vraca funkcija pozvana u poslužitelju

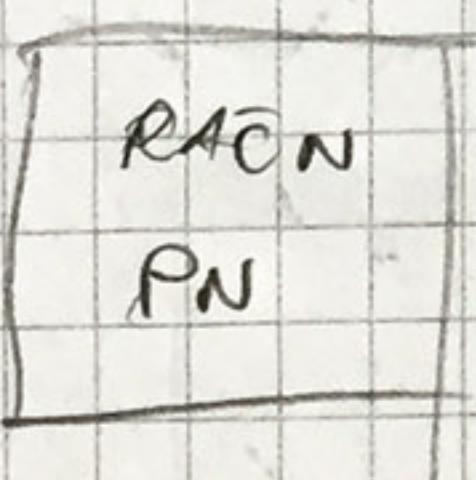
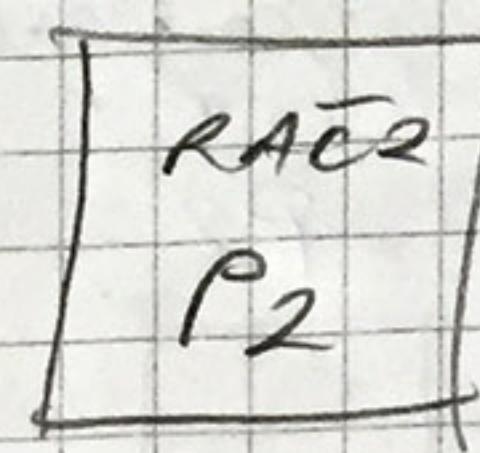
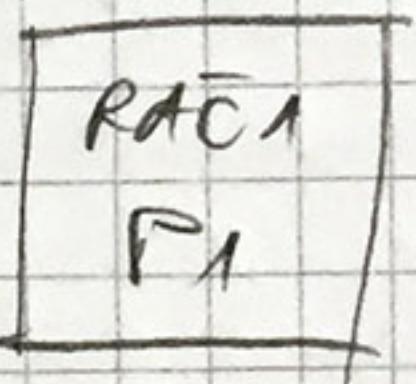
→ Račpo dopoljni dijeljeni spremnički prostor



- $P_i$  - procesi na zasebnim računalima
- svaki proces sebi stvara sliku kako bi virtualni dijeljeni spremnički prostor
- komunikacija među poslužima se realizira raspodijeljenim dijeljenim spremničkim prostorom
  - potrebna je sinkronizacija - sinkronizacijski mechanizmi

→ Metodološko ispitivanje u račodjeljivom sustavima  
vremensko uređenje

- N računala, svaki proces vodi dvije direkte (prva i druga)
- direkte imaju istički odstojanja - može se računati samo na jednom od N računala
- samo paralele; giovad je rezerviran za jedno računalo



- synchronizacija N racionala - jedno zadrzavati  
istovremeno, zero dugje deliti stran
- vremensko uređenje - lokalni logički sat  $C_i$ 
  - svaki proces ima svoj, svaki ima svoju perpektivu na globalni logički sat
  - kada se dogodi dogadjaj  $a \Rightarrow C_i++$
  - $C_i(a) < C_i(b) \iff a \rightarrow b$   
(ako  $a$  prethodi  $b$ )
  - vrijedi transitivity (a → b → c)

### Globalni logički sat

- pravila:

1. Proces  $P_i$  ne može uveljavljati lokalne logičke sata  $C_i$  između svakih dva dogadjaja
2. Kada proces  $P_i$  zatreba poslati ceo svoj dogadjaj vremensku oznaku  $T_m$ , t.d.  $T_m \leq C_i$  (svaki raspodjeljuje lokalne logičke sate)
3. Kada proces  $P_i$  prima poslatu (to je dogadjaj) on ce povećati svoj  $T$  za vrednost već je

postavlja na vrijednost  $(\max_i c_i) + 1$ , tj.

$$C_v = \max \{ C_j, T_m \} + 1$$

- moguća je redukcija lokalnih logičkih zatorova  $\rightarrow$  ne predstavlja problem, umjesto sata se konstruiše id procesa (ili thread id (broj na ulazu uje od koristi))

- Lamportov protocol  $\rightarrow$  sljedeći:

### Programski rješenje problema synchronizacije:

1. Tri programa iz OS-a (Lamport et al.)

2. Sklopovska potpora - koristeći "praksi"

- vsegrado funkcije moraju biti zaštićene na napićem nivou

- instrukcije: TAS (Test And Set), SWAP, CAS (Compare And Set)

3. OS nudi: API!

- semafori

- monitori (koriste binarni semafor)

- time je u potpunosti rješen problem synchronizacije jednog računala; za više:

$\rightarrow$  Centralizirani protokol

- jedan ovor je centralni

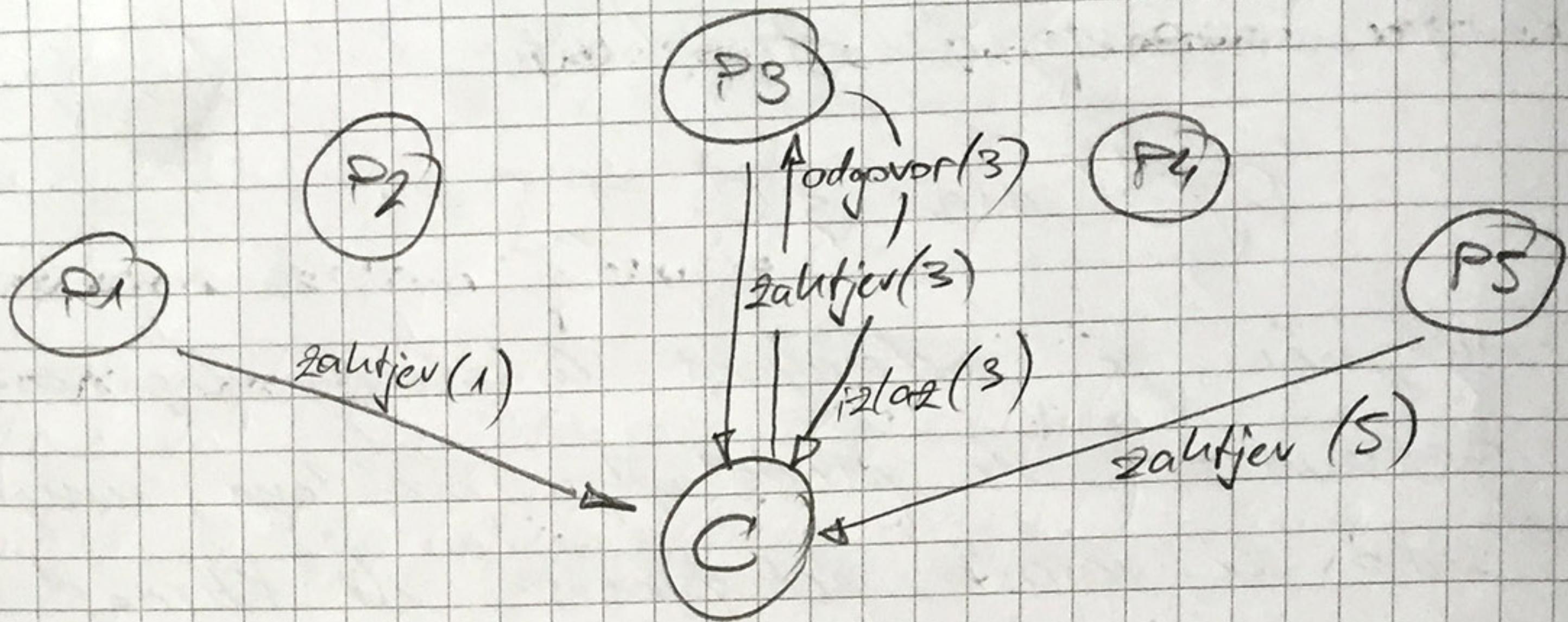
- ako  $\exists$  želi ući u vrtnicu određenih sajta zaštite

C

- first come, first serve (u primjeru 3)

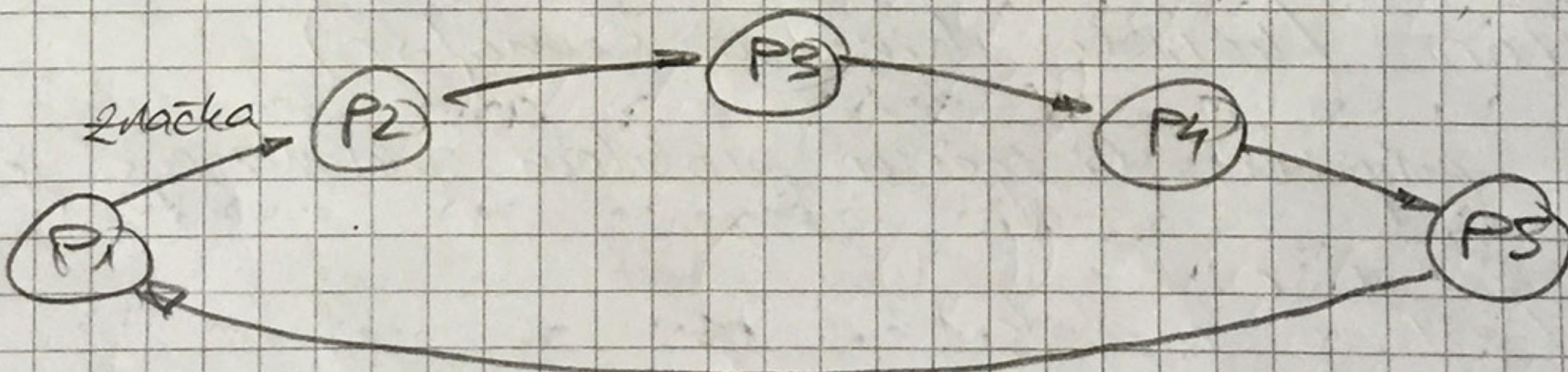
- 1. očekuje

- kad je 3. gotov salje sajtu (3)



- mogući su i prioriteti ; ako S ima visoke prioritete  
1 će i daje zeljstvo
- inačica centralizirajuog protokola - koristi se znakci
- kod centralizirajuog je čvor koji nema zeljstva
- po zeljstvu šalje znakci i čeka ugovornost po potvrđenju
- manja centralizirajuog protokola - moguće veliko  
oprećenje centralizirajuog čvora uslijed velikog broja  
čvorova

→ Protokol s putujućim znakcima



- znakci putujuće uveću učinkovitost
- znakci je poruka ; direkta mora biti konzistentna konfiguracija
- kod Pi želi ući u kontakt sa određenim određenim  
i preseganje ju po zavrsi

## → Lamportov raspodjeljeni protokol

### - pravila:

1. Kada  $P_i$  šali u K.O. ou poruka zahtjev ( $i, T(i)$ ), gdje je  $T(i)$  vrem. oznaka od  $P_i$  (lokalni logički sat)

- zahtjev je poruka

- $P_i$  ju postavlja u svoj red čekanja

- istu poruku šalje svim ostalim procesima

2. Kada proces  $P_i$  primi zahtjev ( $i, T(i)$ ) on uskladi svoj logički sat:

$$C_j = \max \{ C_j, T(i) \} + 1$$

- u svoj red čekanja stavlja novi zahtjev (zahtjev ( $i, T(i)$ ))

3. Bez obzira da sve  $P_i$  šalje

odgovor ( $i, T(j)$ ) → vrijednost u sljedećoj logičkoj  
satnici se na proces  $i$  podnosi

procesu  $P_i$

3.  $P_i$  smije u K.O. kada:

3.1. Njegov sljedići zahtjev je prvi u redu

3.2. Ako je prvi odgovor svih ostalih procesa  
čije su vremenske označke veće od  $T(i)$

4.  $P_i$  izlazi iz K.O., i otklanja se svima na učinku:

4.1. Odstrani svoj zahtjev iz svog reda

4.2. Posalje svim ostalima IZLAZAK ( $i, T(i)$ )

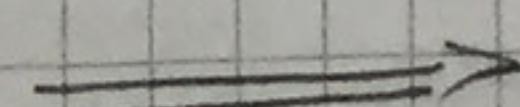
5. Kad proces  $P_i$  primi IZLAZAK ( $i, T(i)$ )

on odstranjuje zahtjev iz svog reda

↓  
prvo  
vjeme  
(jednako kao  
 $i$  u 1.)

- razumevacmo je ukupno  $3(N-1)$

poruka - napom: nedostatak ovog protokola



- primatak je dogodaj pa se sat poscočava; drugo nije važno (konkurenčija)
- redukcija broja poruka:

→ Protokol Ricarta i Agrawala

- Ricart i Agrawala (doktrinacija!) - dva redova
- pravila:

1.  $P_i$  želi ući u kritični odsečak:

1.1. Salje poruku  $ZAHTEV(i, T(i))$  svim ostalim procesima (ne stavlja u svoj red)

2.  $P_j$  primi zahtjev i ugraniči razlika!

$$2.1. C_j = \max\{C_j, T(i)\} + 1$$

2.2.  $P_j$  salje poruku  $ODGOVOR(j, T(j))$  ako ne želi ući u kritični odsečak

2.3. Ako želi ući u kritični odsečak ne

salje odgovor

- preciznije: ako  $P_i$  želi u K.O. tada ima  $ZAHTEV(j, T(j))$  i  $(j, T(j))$  prethodi  $(i, T(i))$ , tj.

$$(j, T(j)) < (i, T(i))$$

sto znači:

$$\begin{cases} T(j) < T(i) \\ \text{ako } T(j) = T(i) \Rightarrow j < i \end{cases}$$

inace i ima prioritet

3. Kada proces  $P_i$  primi sve odgovore može u K.O.

4. Kada  $P_i$  izlazi iz K.O. salje  $ODGOVOR$  svim procesima koji su izrazili zahtjev za ulazak u K.O. →  $ODGOVOR(i, T(k))$

- ukupno  $2(N-1)$  poruka, gdje je  $N$  broj procesa
  - primanje odgovora je sekviranje logičkog sata ( $i$  kod  $P_i$ , i  $i+1$  kod  $P_{i+1}$ ), tj. odgovor je dogodaj <sup>colealnog</sup>
  - poslednji odgovor (korak 4.):
    - ODGOVOR ( $i; T(k)$ )
    - $T(k)$  je vremenska oznaka onog koga je poslao zahtjev na  $P_i$  (bitko kog je proces specijalno)
    - Šalje vlastiti indeks i individualna forma kao i odgovor iz 2.2. (analogija, samo se radi o razlikoj u procesima)
- 

- kriptografija radionica - prijevoz krajem sezone, raspodjeljuje se u razne slične padove

### → labeš

- što je dogoda prihvatanja ulaska procesa  $P_i$  u K.O.?
  - ispis u svim mrežama odgovarajućim mrežama; mreža bit će tako da jedan proces paralelni s drugim ispisuje da je u konkretnom odgovoru
  - jedan proces druge mreže "update" u svakom kod njegovi u K.O., ali se ne smije dogoditi da su istovremeno u K.O.
  - isporučati kerke.c i spode.c → kerke šalje poruku, spode ju čita
    - ne zatravljaju red poruka, ostaju spajati; valja isporučiti red poruka!
    - ipos-status; iposm - q x - err-sauje

## - labos ryčetki Ricartom i Agrawalom

- red poruka - mala kolveta saformacija (parceto 8)  
koja se šalje u kom drugom procesu
  - poruka se šalje u red poruka
  - proces zna da je poruka za nešta prava tipa poruke
- red poruka i gorenje se može upotrijebiti kao semafor
- okolina - uz znakova oblike "me = vrjednost"; djele se u dva skupine:
  - za gresku
  - za svaki proces problem potreban
- TCP dodaje redni broj i sistemu; IP - daje adresu
- briči gorenje - dva opseka, menovani jedan
- povezivanje gorenja na STD IN među satravci standardni ulaz
- uobičajeni komunikacijski izmjeni procesa: gaf. spremnik, poruke, var. oblike, datoteke, gorenje
- različita ravnala: samo poruke (drugi uklanjanju se oslanjaju na poruke), to su: RPC, raspodjeljeni računari, adr. prostor
- lokalni logički sat se pojavlja kao brojko izmjeni svaka dva dogadjaja (prvi tak poruke je dogadjaj, ostalo opisualno)
- kampart -  $3(N-1)$  poruka

Zadatak 10.1. U sustavu se nalaže tri dosta na

zajednačiće odavje po redom procesi:  $P_1, P_2, P_3$ .  
Vi procesi u produktivnosti trenutku imaju u  
svakom ček. log. zatvaraju vrijednosti 12, 7, 20  
(redom).

→ Import

a) Ako u tom trenutku proces  $P_1$  želi ući u k.o.  
ispitati što će se biti dogoditi u sustavu.

- red kojim će dovršiti posluge je generalno  
neodređen; mogući su različiti rezultati kao  
posljedica različitih  $C_i$  ne uspostavljenih  
rezultata - da uobičajeno, zatočen su faktovi da  
postoji više raznih mogućnosti

$P_1: C_1 = 12$  red zatijera (red):

$P_2: C_2 = 7$  red:

$P_3: C_3 = 20$  red:

(strelice odgovaraju  
stavu poruka)

dobiti se oblikuje rezultat  
za  $C_1$  (oboga  
ispravio)

(P1)

(P2)

(P3)

generalna

DATIJEV (1, 12)

i id. u red

(P2)

u red

ODGOVOR (1, 13)

u red

$C_3 = \max \{20, 12\} + 1 = 21$

ODGOVOR (1, 21)

$C_1 = \max \{12, 21\} + 1 = 22$

$C_1 = \max \{22, 13\} + 1 = 23$

primo sve odgovore  
i uobičajeno

K.O.

izgadaju iž K.O.:

- briši smj. zatijer u  
svog reda

IZLAZAK (1, 12)

prototip  
 $T(c)$ !

$C_2 = \max \{13, 12\} + 1 = 14$

brisi 2. MJEV (1, 12)  
iz reda

$C_3 = \max \{21, 12\} + 1 = 22$

brisi 1. reda

STANJE na krajnju faza (ako su sto  $P_1$  i  $P_2$  u K.O.)

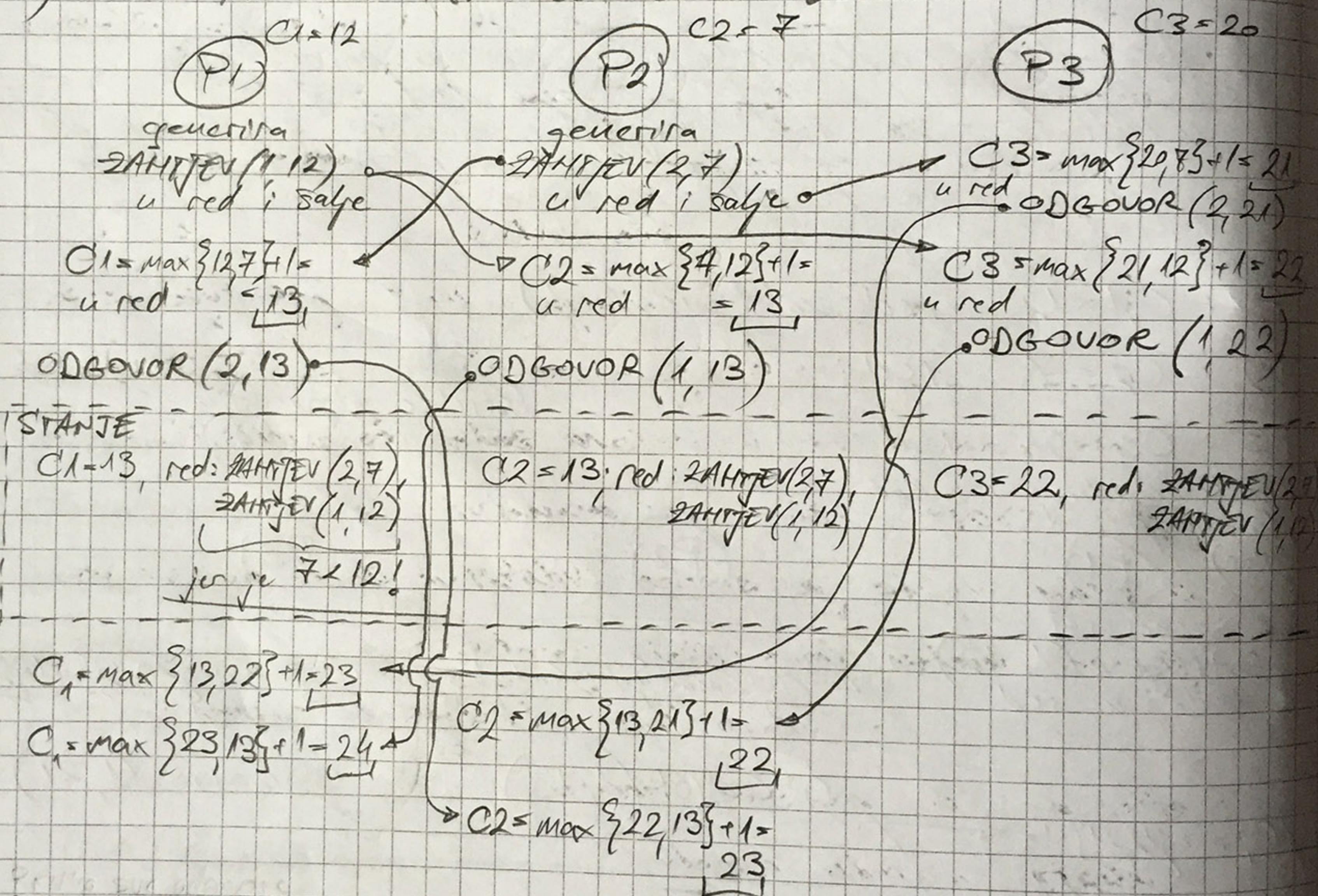
$P_1: C_1 = 23$ , red: -

$P_2: C_2 = 14$ , red: -

$P_3: C_3 = 22$ , red: -

- azurirajuće  $C_i$  uvek potrebuju; algoritam će raditi individualno; ako je dogodaj primitak svake poruke tada ga je potrebno azurirati

6)  $P_1$ ;  $P_2$  žele rastorenuće u K.O.



PRIMIO SVE ODGOVORE, ali  
njegov zadatak nije da  
prvom mijestu  
 $\Rightarrow$  ne može u K.O.  
mora očekati

$C_1 = \max\{24, 7\} + 1 = 25$   
bitiće iz reda  
red:  $ZAHTEV(1, 12)$

ide u K.O.

PRIMIO SVE ODGOVORE,  
1. da redu, ide u K.O.

IZLAZAK (2, 7)  
bitiće iz reda  
red:  $ZAHTEV(1, 12)$

$C_3 = \max\{22, 7\} + 1 = 23$   
bitiće iz reda  
red:  $ZAHTEV(1, 12)$

(P1)

$$T_{\text{ZATA}}(1, 12) \\ \text{red} = - \\ (C_1 = 25)$$

(P2)

$$C_2 = \max \{23, 12\} + 1 = 24 \\ \text{red} = -$$

(P3)

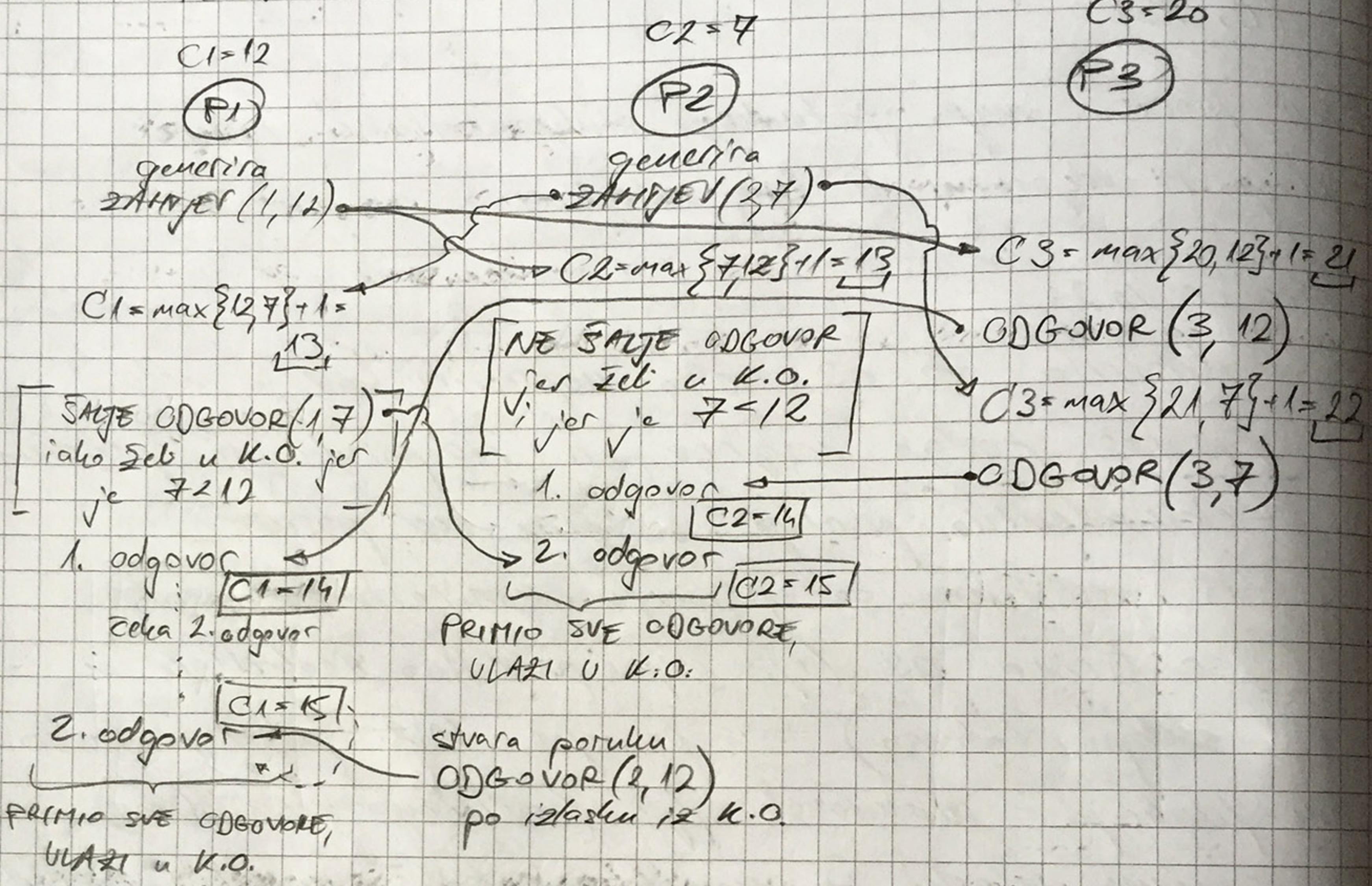
$$C_3 = \max \{23, 12\} + 1 = 24 \\ \text{red} = -$$

- svr' procesi imaju v'edualne, "stukarou izbrane" redove
- projekti odgorata u broj stranica
- prijevjer 10.2:

- ilustrira sto os velja napraviti kad je uveli' proces postao uskoren, a red posluka je prešao ekvivalentna pravina u formi red posluka
- tv problemi se resavaju monitoriraju - za to se brine os (fj. uve potrebu realizirati u sklopu labosa)  $\rightarrow$  prijevjer 10.2. potrebije tako
- on-funkcija - monitoriraju fja (funkcija monitorom)
- nema relacije iz monitora zato sto unisti - u red-a uvrsta sadrži izlaz iz monitora
- u redu uvrsta je proces blokiran dok ne se ne popavi direkt na logu tko suprotno
- dobitka procesa  $P_i$  može biti: (petka u kujuci)
- aktivacija u u.o. djelu
- očekati u redu uvrsta
- aktivacija u u.o.
- po izlasku iz u.o. se može zadiviti u monitoru dok želite posluć drugim

Zadatak 10. 1.

C) Isto kao : b), ali Ricart ; Agrawala  
- desna reda!



- U ovom zadatku je dogovor da se dešavira logički sat po primljenju odgovora; u krajnje suprotnu
- stavlja se ažuriranu  $C_i$  sa oznacena gore kas

$$[C_i = n]$$

- za MI ; 2T  $\rightarrow$  svaka primljena poruka je dogadaj, svaki put treba ažurirati logički sat (skolska varijanta)
- do 2. zadatka ovog tipa će se pojaviti na MI