Dijana Prokic

\*znaci da nije odgovoreno

? nije jasno, ili nije sigurno da je dobro uradjeno

1.2.2018.

1. **Sta se podrazumeva pod DS? Navesti tri prednosti I tri nedostatka DS u odnosu na centralizovane.**

DS cini kolekcija racunara koji su medjusobno nezavisni i prividno rade kao jedan jedinstveni sistem.

Mane:

-Mnogo veća kompleksnost DS od centralizovanog

Softver: teže je razviti sofver za DS nego za centralizovane sisteme

-Umrežavanje:

Povezanost je ključna za funkcionisanje DS: mreža može otkazati ili postati zagušena

-Bezbednosni problemi

Laka dostupnost resursa, dovodi do problema bezbednosti

\*9.6.2021.

1. **a. Koje su prednosti DS u odnosu na centralizovane?**
   * Ekonomski: DS imaju bolji odnos cena/performanse od velikih (mainframe) računara.
   * Brzina: DS može imati veću ukupnu moć obrade nego mainframe (Npr. Jedna velika baza podataka može biti razbijena na manje baze.)
   * Pouzdanost: Ako imate centralizovani sistem i računar otkaže, aplikacija ne može da se izvršava. Kod DS (ako je dobro projektovan) ako 15% mašina otkaže, aplikacija i dalje može da se izvršava
   * Inkcrementalni rast: sistem se može postupno povećavati dodavanjem novih računara.
   * Deljenje resursa: Omogućeno je da više korisnika pristupa zajedničkim bazama podataka i periferijama.
   * Komunikacija: olakšana komunikacija između ljudi.
   * Efikasno korišćenje resursa: omogućava da se opterećenje rasporedi na raspoložive računare na najefikasniji način

**b. Zasto je tesko ostvariti sinhronizaciju u DS?**

**c. Zasto je (nekada) problem detektovati greske u DS?**

28.4. , 30.9.2020.

1. **Sta se podrazumeva pod skalabilnim distribuiranim sistemom? Koje se tehnike koriste za postizanje skalabilnosti?**

Skalabilnost (proširljivost) DS je jedan od najvažnijih projektantskih zadataka.

Skalabilnost se može posmatrati kroz tri dimenzije:

* 1. Skalabilnost u odnosu na broj korisnika i resursa
  2. Skalabilnost u odnosu na geografsku udaljenost resursa i korisnika
  3. Administrativna skalabilnost – sistemom se može lako upravaljati čak i ako se prostire kroz više administrativnih domena.

Postoje 3 tehnike skaliranja:

* + Skrivanje komunikacionog kašnjenja: Koriste se asinhrone komunikacije umesto sinhronih

Kada se uputi zahtev udaljenom servisu, klijent se ne blokira dok čeka da stigne odgovor, već radi neki drugi posao. Download-ovati deo koda na klijent stranu da bi se ubrzala obrada

* + Distribucija: Podrazumeva da se komponente dele na manje delove, a zatim se ti delovi distribuiraju na više mašina u sistemu
  + Replikacija : Pomaže da se poveća dostupnost i balansira opterećenje u sistemu da bi se postigle bolje performanse. U sistemima koji su geografski distribuirani poželjno je da postoji kopija resursa blizu mesta korišćenja kako bi se smanjilo komunikaciono kašnjenje.

2.7.2021.

1. **Sta predstavlka transakcija u D informacionim S? Koje osobine treba da zadovolji svaka transakcija da bi bila uspesno izvrsena?**

* Transakcija predstavlja skup operacija koje se obavljaju kao jedna nedeljiva (atomična) operacija.
  + Sistem za obradu transakcija obezbedjuje da sve ili ni jedna operacija u transakciji budu izvršene bez greške.
  + nakon obavljanja transakcije sistem (obično baze podataka ili fajl sistem) mora da ostane u poznatom, konzistentnom stanju tako što obezbedjuje da se operacija koje su medjusobno zavisne ili sve izvrše ili se ni jedna ne izvrši.

Svaka transakcija mora da zadovolji **ACID** test pre nego što se dozvoli modifikacija sadržaja

A – Atomicity– podrazumeva da se transakcija obavi kompletno ili se uopšte ne obavi

C - Consistency –transakcija ne ugrožava skup invarijanti (ograničenja, konstanti) sistema.

* + - Npr. ako je uslov da sve transakcije nad bazom podataka moraju imati pozitivnu vrednost, bilo koja transakcija sa negativnom vrednošću će biti odbijena. (npr. ako pokušate da podignete više novca nego što imate na računu)

I – Isolation –transakcije su medjusobno izolovane, tj. dve konkurentne transakcije moraju biti serijalizovane (izvršavaju se u nekom nedeterminisanom redosledu, ali taj redosled mora biti vidljiv na isti način za sve transakcije)

D - Durability –kada se transakcija obavi ne može se poništiti!

2.7.2017.

1. **Sta je middleware I koja je njegova uloga u ds?**

Komunikacija između procesa je u srcu svih distribuiranih sistema.

Komunikacija u DS je uvek bazirana na slanju poruka, tj. ne postoji zajednički adresni prostor.

Da bi se pojednostavilo pisanje i razvoj distribuiranih aplikacija potrebno je obezbediti softversku podršku koja će aplikativnog programera osloboditi detalja vezanih za interprocesnu komunikaciju, sinhronizaciju, bezbednost itd.

Da bi se pojednostavilo pisanje distribuiranih aplikacija i njihova integracija u DS na Mrežni OS je moguće je dodati još jedan softverski sloj (MIDDLEWARE) sa ciljem da sakrije heterogenost platforme na kojoj je sistem izgrađen od aplikacije, i da se sakrije komunikacija, tj. distribuciju.

Middleware sistemi nude kompletan skup usluga aplikaciji i ne dozvoljavaju korišćenje ničeg drugog do njihovih interfeisa prema uslugama. Npr. ne može se “preskočiti” middleware sloj i koristiti socketi za komunikaciju.

Osnovni cilj middleware je da se sakrije heterogenost platforme na kojoj je sistem izgrađen od aplikacije.

Middleware sistemi nude kompletan skup usluga aplikaciji i ne dozvoljavaju korišćenje ničeg drugog do njihovih interfejsa prema uslugama

1.2.2018.

1. **Objasniti razlike izmedju sinhronih I asinhronih komunikacija.**

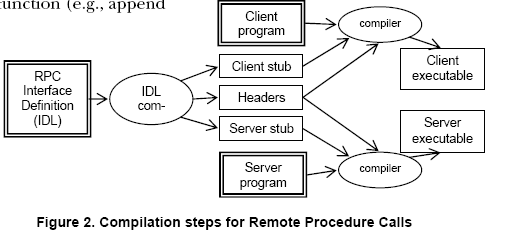
-Kod **asinhronih** komunikacija pošiljalac nastavlja sa radom odmah nakon što prosledi poruku za slanje

Poruka se pamti ili u lokalnom baferu ili u komunikacionom serveru

-Kod **sinhronih** komunikacija pošiljalac se blokira dok se poruka ne zapamti u lokalnom baferu odredišnog hosta (tj. -dok ne stigne do odredišta)

7.9.2021

1. **Kako izgleda poziv Sun RPC komplajlera? Koji fajlovi se dobijaju kao rezultat komplajliranja I sta sadrze? Objasniti na primeru.**



* 1. rpcgen kompajler generiše tri fajla (npr rpcgen primer.x)
     1. header fajl (npr. primer.h)
        1. Header fajl sadrži jedinstveni identifikator interfeisa, definiciju tipova, konstanti i prototipova funkcija.
        2. On treba da bude uključen (korišćenjem #include) i u klijent i u server kod
     2. klijent stub (**primer\_clnt.c)**
        1. Klijent stub sadrži procedure koje će klijent program pozivati
        2. Ove procedure su zadužene za pakovanje parametara u poruke, slanje poruka, prijem poruka, izvalačenje reazultata poziva procedure i prosleđivanje klijentu
     3. server stub (**primer\_svc.c ) –**
        1. Sadrži procedure koje se pozivaju kada stigne poruka do servera i koje zatim pozivaju odgovarajuću serversku proceduru

28.4

1. **Servis IZBORI ima dve udaljene procedure :**
   1. **Glasanje- sa dva parametra preko kojih klijent navodi ime kandidata(string) i svoj JMBG broj(integer)**
   2. **Rezultat- sa jednim parametrom koji predstavlja ime kandidata a kao rezultat vraca broj glasova koji je kandidat osvojio.**

**Definisati interfejs za servis IZBORI pomocu Sun XDR(jezika za definiju interfejsa).**

struct op{ char\* ime, int jmbg;} ;

program IZBORI

{

Version IZBORI1

{

void Glasanje(op) = 1;

int Rezultat(char\*) = 2;

} = 1

} = 0 x 20000001

9.6.2021

1. **Pre nego sto klijent uputi RPC poziv server, server mora da bude registrovan. Kako se obavlja registrovanje servera? Sta je portmapper? Sta se podrazumeva pod tehnikom “binding” (povezivanje)?**

klijent mora znati ime udaljenog servera

server registruje svoje usluge preko portmapper deamon procesa na serverskoj mašini (uvek na portu 111).

Da bi klijent mogao da poziva server, neophodno je da server bude registrovan i spreman da primi poziv.

* + Registracija servera omogućva klijentu da locira server i da se poveže sa njim
  + DCE klijent pronalazi server u dva koraka
    - Lociranje serverske mašine
    - Lociranje servera (tj. odgovarajućeg procesa) na serverskoj mašini.
  + Da bi klijent mogao da komunicira sa serverom (tj. procesom) on mora da zna broj porta (end point) na serverskoj mašini na koji može poslati poruke
    - Server se registruje i u direktorijumskom serveru tako što dostavlja ime serverske mašine i svoje interfeise
* Da bi se realizovao poziv udaljene procedure neophodno je locirati udaljeni host i odgovarajući proces na hostu
  + Povezivanje (binding).

-Dva rešenja su moguća

* + Statičko povezivanje: Klijent zna koji host treba da kontaktira (adresa hosta se nalazi u klijent stub-u), a kada klijent pozove odgovarajuću proceduru, klijent stub jednostavno prosledi poziv serveru.
    - Poseban program (**portmapper**) na udaljenom hostu pamti preslikavanja imena programa i broja verzije u broj porta

1. Dinamičko povezivanje : Drugo rešenje je da postoji centralizovana baza podataka (smeštena u name i directory serverima) koja može locirati host koji obezbeđuje željeni servis. Ako server koji implementira željenu proceduru promeni adresu, dovoljno je promeniti ulaz u serveru imena.

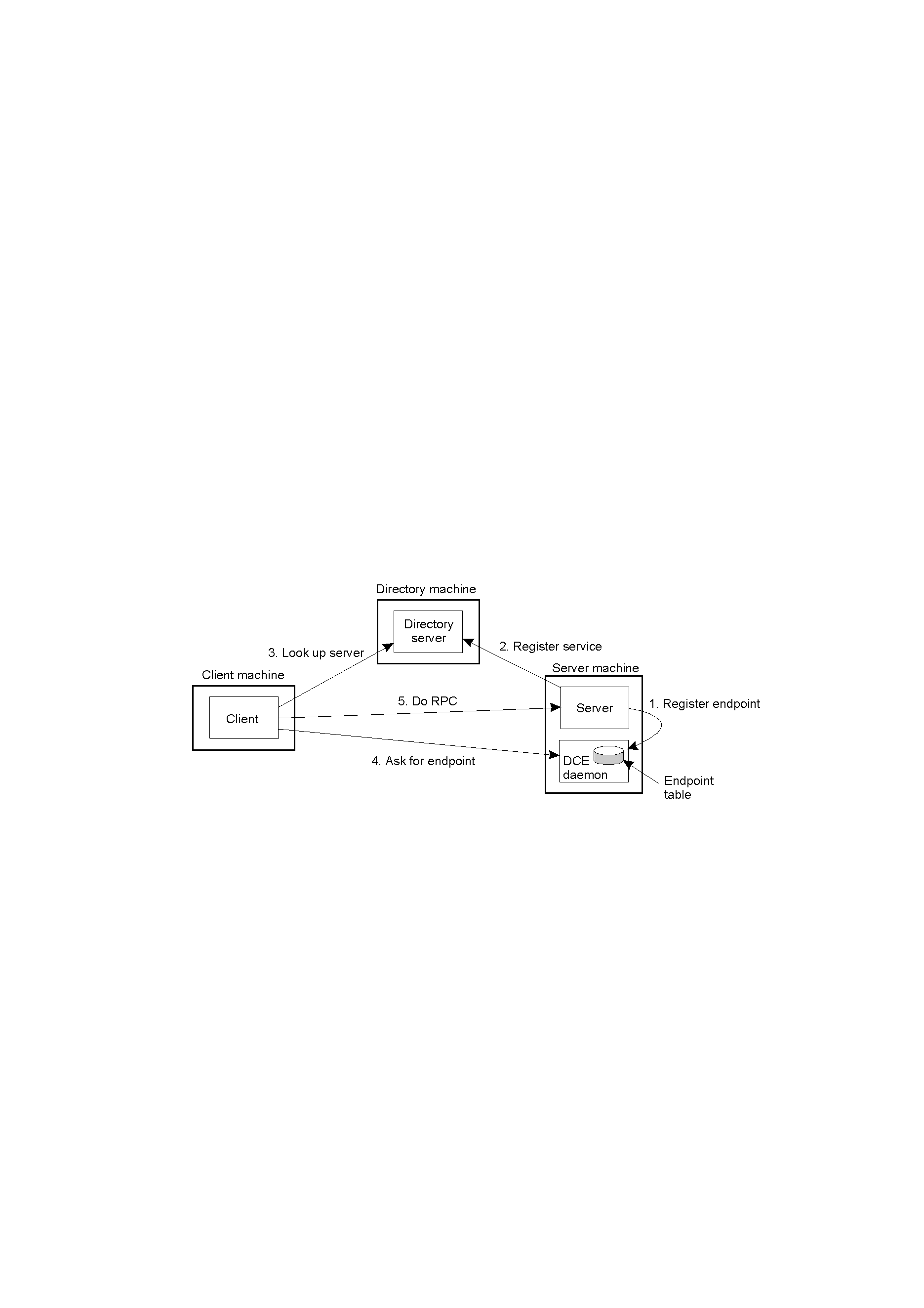
Serverski stub registruje serversku proceduru u serveru imena i direktorijuma prilikom stratovanja servera

28.4.

1. **Kako se moze locirati server koji implementira udaljenu proceduru? (odgovor pitanje iznad)**

8.9.2020

1. **Pobrojati redom korake kojima se ostvaruje povezivanje klijenta I server kod DCE.**



1. Registrovanje broja porta servera (procesa) u DCE deamonu (Registracija servera omogućva klijentu da locira server i da se poveže sa njim)
2. Registrovanje servera u direktorijumskom servisu (adresa serverske mašine, ime servera i interfeisi koje implementira)
3. Klijent kontaktira direktorijumski server da dobije adresu server mašine (IP adresu) na kojoj se izvršava server(DCE klijent pronalazi server u dva koraka:

Lociranje serverske mašine

Lociranje servera (tj. odgovarajućeg procesa) na serverskoj mašini.)

1. Klijent kontaktira DCE deamon da bi dobio broj porta servera kome želi da pristupi
2. Poziv udaljene procedure

11.10.2017

1. **DCE cell directory servis u slucaju RPC omogucava da:**

**a. klijent pronadje na kom server je implementirana udaljena arhitektura T**

b. klijentu da pronadje broj porta na kome je implementirana udaljena procedura

c. server obavi prozivku klijenata

d. distribuciju objekata izmedju vise servera

26.6.2017

1. **Objasniti ulogu klijentskog I serverskog staba.**
   1. Ako je *read* udaljena procedura (koja će se izvršiti na serveru) drugačija verzija read, koja se zove klijent stub se poziva koja na osnovu osnovu primljenih parametara gradi poruku i poziva lokalni OS da prenese poruku do udaljenog servera.
      1. Stub funkcija izgleda kao funkcija koju korisnik želi da pozove, ali ona stvarno sadrži kod za slanje i prijem poruka kroz mrežu.

Klijent stub pakuje argumente za udaljenu proceduru (ovde se može vršiti i konverzija u standardni format, ako je potrebno) i gradi jednu ili više mrežnih poruka. Pakovanje argumenata u mrežnu poruku zove se *marshaling*.

- Klijent stub izvlači rezultat iz poruke i vraća klijentskom procesu

* 1. -Serverska stub funkcija prima poruku od lokalnog OS (izvršava receive primitivu), raspakuje je (unmarshaling) i izvlači argumente za poziv lokalne procedure

-Serverski stub poziva željenu serversku proceduru i predaje joj parametre koje je primio od klijenta (stavljajući ih u stek – kao kod poziva lokalne procedure).

- Serverski stub pakuje rezultat u poruku i poziva svoj lokalni OS (poziva *send* primitivu, a nakon toga ponovo *receive* primitivu čekajući novi zahtev).

-Server stub poziva (preko OS sistemskog poziva) mrežnu rutinu za slanje poruka

U delu:

10 koraka kod poziva RPC:

1. Klijent poziva lokalnu proceduru (klijent stub)
   * Za klijntski proces ona izgleda kao aktuelna procedura koju treba pozvati.
2. Klijent stub pakuje argumente za udaljenu proceduru (ova aktivnost može uključiti i konverziju u standardni format) i gradi jednu ili više mrežnih poruka (messages) i poziva lokalni OS (poziva *send* primitivu, a nakon toga i *receive* primitivu i čeka se dok ne stigne odgovor).
   * + Pakovanje argumenata u mrežnu poruku se zove “parameter marshaling” (uređivanje parametara)
3. Lokalni OS šalje poruku udaljenom OS (korišćenjem transportnog protokola)
4. Udaljeni OS prosleđuje poruku serverskom stub-u. Serverska stub funkcija prima poruku od lokalnog OS (izvršava *receive* primitivu), raspakuje je (unmarshaling) i izvlači argumente za poziv procedure.
5. Serverski stub poziva željenu serversku proceduru i predaje joj parametre koje je primio od klijenta (stavljajući ih u stek – kao kod poziva lokalne procedure).
6. Server izvršava pozvanu proceduru i vraća rezultat stub-u.
7. Serverski stub pakuje rezultat u poruku i poziva svoj lokalni OS (poziva *send* primitivu, a nakon toga ponovo *receive* primitivu čekajući novi zahtev).
8. Serverski OS šalje poruku klijentskom OS
9. Lokalni OS prosleđuje poruku klijnt stub-u
10. Klijent stub izvlači rezultat iz poruke i vraća klijentskom procesu

8.7.2020

1. **Objasniti razliku između tranzijentnih i perzistentnih komunikacija u distribuiranom sistemu baziranom na midleware sa razmenom poruka.**

\*Perzistentne (istrajne): poruka se pamti u komunikacionom serveru koliko je potrebno da bi se isporučila odredištu.

-poruke koje se prenose se pamte u komunikacionim serverima dok se ne proslede sledećem komunikacionom serveru.

-izvor ne mora biti aktivan nakon što dostavi poruku komunikacionom serveru

-prijemnik se ne mora izvršavati u trenutku kada izvor pošalje poruku.

- E-mail sistem je tipičan primer perzistentne (istrajne) komunikacije

\*Tranzijentne: poruka se odbacuje ako komunikacioni server nije u stanju da je isporuči sledećem serveru ili odredištu

-Kod tranzientnih komunikacija poruke se skladište samo dok se izvorna i odredišna aplikacija izvršavaju.

-Ako se agenti ne izvršavaju, poruke se odbacuju. Komunikacioni server u ovom slučaju je a store-and-forward ruter.

29.9.2017.

1. **U trenutku 117 primljen je NTP odgovor sa sledecim informacijama:**

**Zahtev poslat u trenutku 82 , primljen u trenutku 111, odgovor je posslat u trenutku 120. Kako ce se korigovati lokalni casovnik:**

* 1. Uznapredovace za 21
  2. **Uznapredovace za 16 T**
  3. Uznapredovace za 9
  4. Uznapredovace za 32

t0=82, t1=111, t2=120, t3=117

Ө=((t1-t0)-(t3-32))/2 =((111-82)-(117-120))/2=(29+3)/2=32/2=16

t=t3+ Ө=117+16=133

28.4.

**Lamportov algoritam**

1. **Tri računara A, B i C komuniciraju koristeći protokol koji implementira Lamportove vremenske markice. Na početku u svim računarima vrednosti vremenskih markica su 0. Kasnije nastupaju sledeći događaji:**

**• A šalje poruku M1 u B: „Zdravo“**

**• Nakon što je primio M1, B šalje poruku M2 u C: „A mi se javio“**

**• Nakon što je C primio M2, šalje poruku M3 u A: „B mi dosađuje“**

**(a) Uz svaku od poslatih poruka napisati vrednost vremenske markice sa kojom je poslata:**

Send (M1, 1 )

Send (M2, 3)

Send (M3,5 )

**(b) Nakon ovih događaja, poslate su još tri poruke:**

**• Nakon što je primio poruku M3, A šalje poruku M4 u B : „C je nedosledan“**

**• Nakon što je primio poruku M4, B šalje poruku M5 u A: „C mi dosađuje“**

**• A prima poruku M5.**

**Nakon što su sve poruke poslate i primljene, kolike su vrednosti vremenskih markica u svakom od računara A, B i C?**

A 10

B 9

C 5

**(c)Da li opisani slučaj predstavlja primer relativne ili potpuno uređene komunikacije?**

U pitanju je relativno uređenje događaja, tj. nema multicast-a.

\*?7.9.2021.

1. **Tri racunara A, B i C komuniciraju I koriste protocol koji implementira Lamportove logicke casovnike. Na pocetku su svi log. casovnici na 0, kasnije nastupaju sledeci dogadjaji:**

* A salje poruku M1 u B.
* C salje poruku M2 u B.
* B prima poruku M2 pre poruke M1.
* B odgovara porukom M3 prvo racunaru C, a zatim porukom M4 rac. A.
* Nakon primanja M3, C salje poruku M5 racunaru A.
* Nakon primanja M5, A salje poruku M6 racunaru B.
* Nakon primanja M6, B salje poruku M7 racunaru A.
* Poslednja poruka koju A prima pre poruke M7 je M4.
  1. **Oznaciti kako izgledaju vremenske markice poruka M1 do M7 prilikom slanja:**

Send(M1, 1)

Send(M2, 1)

Send(M3, 3)

Send(M4, 3)

Send(M5, 5)

Send(M6, 7)

Send(M7, 9)

Nisam sigurna

* 1. **Navesti bar jedan par dogadjaja koji su medjusobno uslovljeni, a cija je kauzalnost korektno identifikovana Lamportovim markicama.**

???????????????????

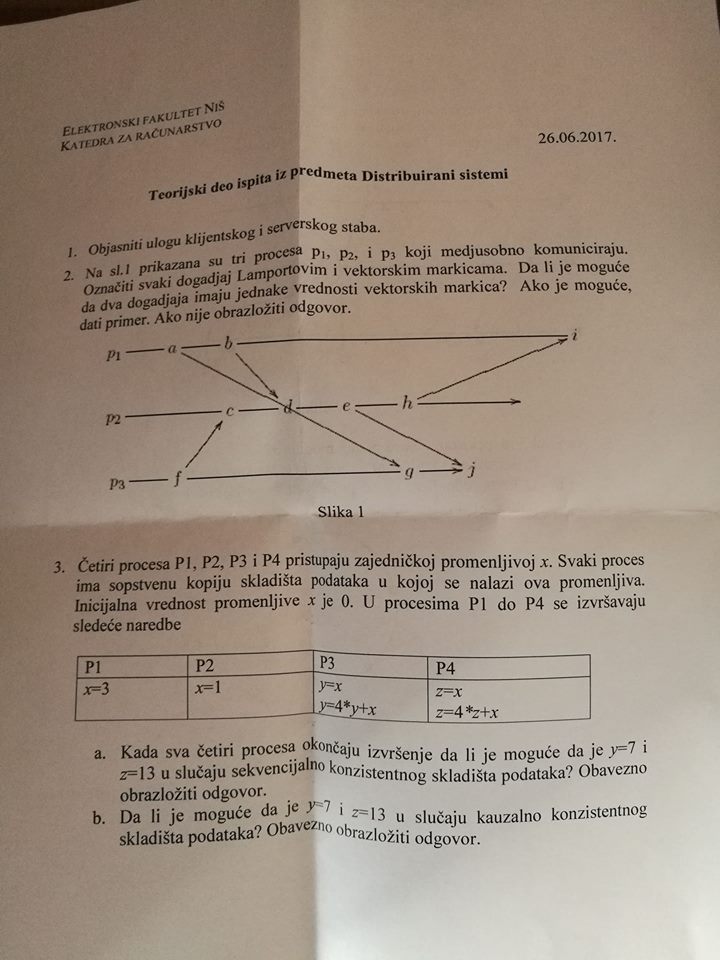
* 1. **Da li se dogadjaj kada B salje poruku M4 I kada C salje poruku M5 mogu identifikovati kao konkurentni dogadjaji koriscenjem Lamportovih markica. Objasniti zasto.**

Dva dogadjaja su konkurentna kada nisu medjusobno uslovljena.

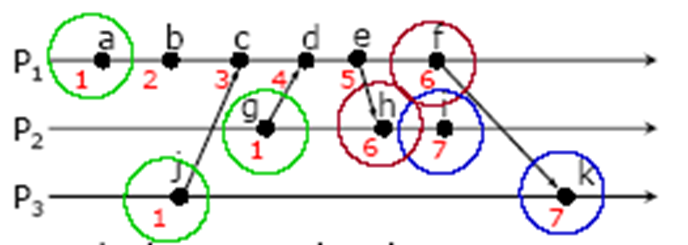
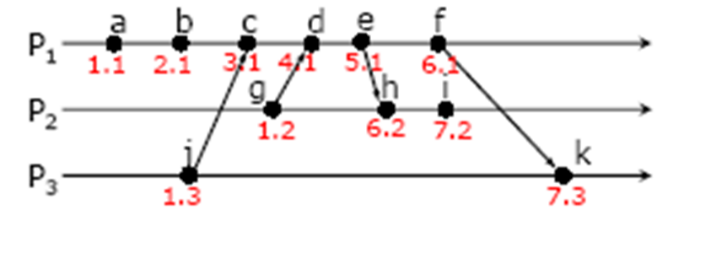
Mogu se identifikovati kao konkurentni, jer ne zavise jedan od drugog.

?????????????????

1. **Na slici su prikazana 3 procesa p1,p2,p3 koji medjusobno komuniciraju. Oznaciti svaki dogadjaj Lamportovim i vektorskim markicama. Da li je moguce da dva dogadjaja imaju jednake vrednosti Lamportovih markica? Ako je moguce, dati primer, ako nije, obrazloziti.**

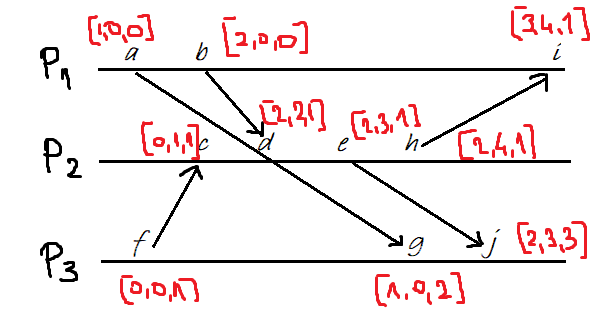
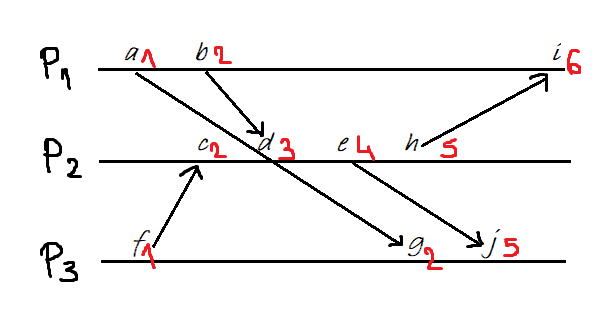
****

Moguće je da više događaja koji nisu međusobno uslovljeni (tj. konkurentni su) imaju identične vremenske markice

. 

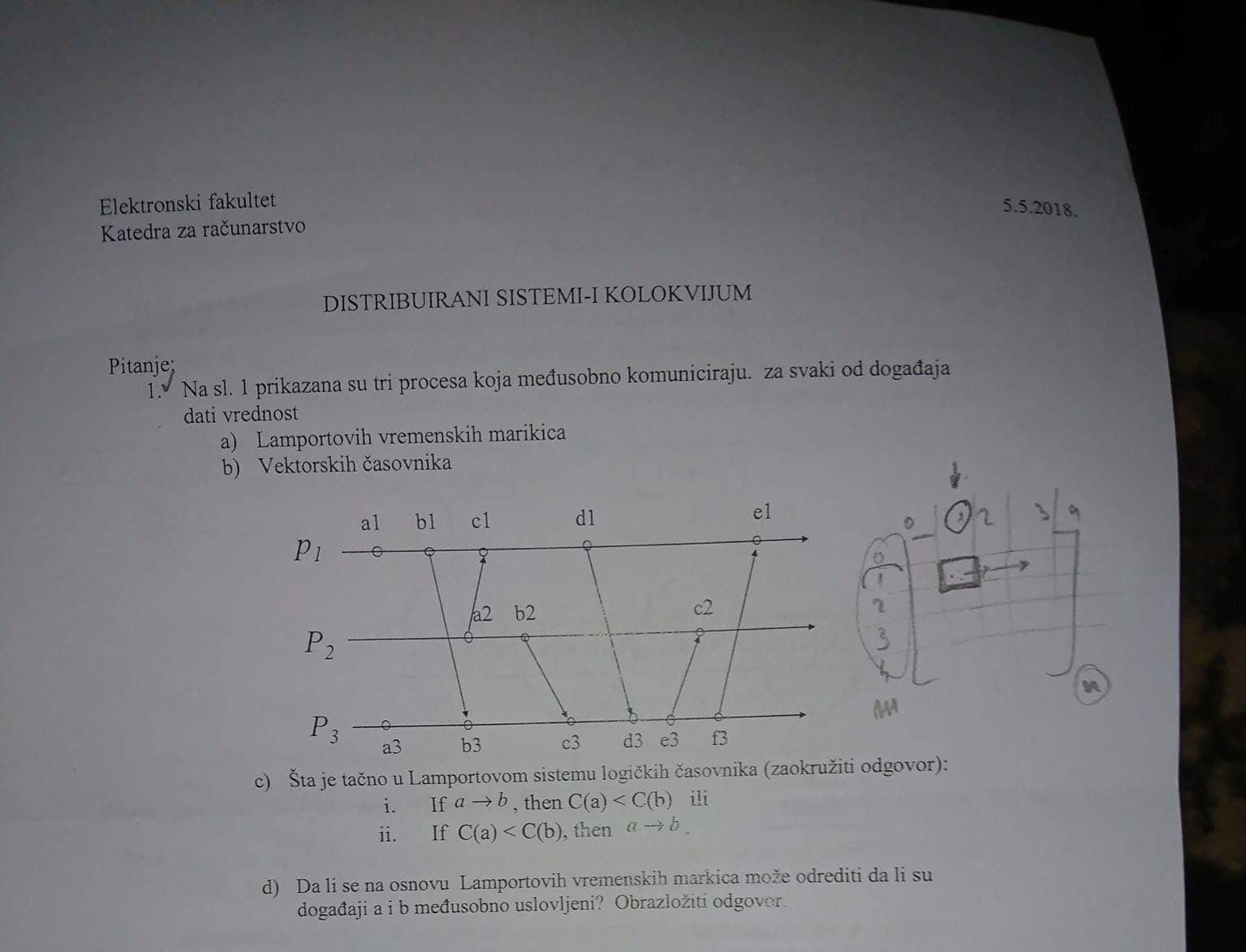
Prisiliti da svaka markica bude jedinstvena

a.Lamportov: b. Vektorski:

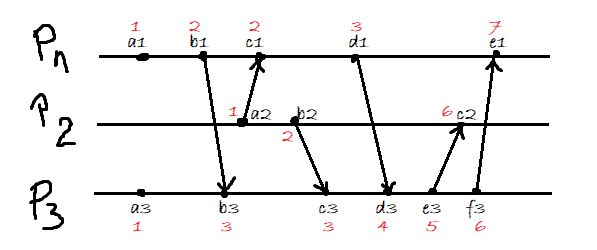


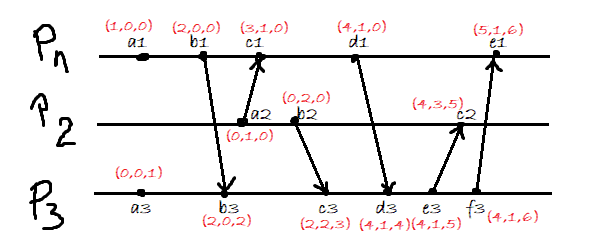
5.5.2018.

1. **Prikazana su 3 procesa koja medjusobno komuniciraju, za svaki od dogadjaja dati vrednost:**
   1. **Lamportovih vremenskih markica**
   2. **Vektorskih casovnika**



Lamportov:

Vektorski:



* 1. **Sta je tacno u lamportovom sistemu log.casovnika?**
     1. **If a->b, then C(a)<C(b) T**
     2. **If C(a)<C(b), then a->b**
  2. **Da li se na osnovu Lamportovih vremenskih markica moze odrediti da li su dogadjaji a i b medjusobno uslovljeni? Obrazloziti.**

Ne moze, zato se koriste vektorski casovnici.

\*?11.9.2017

1. **Simbol -> oznacava Lamportovu relaciju “desilo se pre”(happened-before), a I b su dogadjaji, Va je vektorski casovnik duzine n koji predstavlja vreme kada je dogadjaj a nastupio.**
2. **Koja karakteristika sistema odredjuje velicinu n?**

N odredjuje broj procesa, pa je zbog toga vector duzine N

1. **Definisati sta znaci Va=Vb i Va>Vb.**

Va>Vb znaci da b->a, tj, da b prethodi a. Ako ne moze da se uspostavi Va<=Vb Ili Va>=Vb, to znaci da su konkurentni.

Va=Vb ako Va[i]=Vb[i], za i=1,..N

Va=Vb ?

1. **Ako je a->b, sta znamo o Va i Vb?**

Znamo da je Va<Vb

1. **Ako ne vazi a->b, niti je b->a, sta znamo o Va i Vb?**

?????????

11.10.2017

1. **Sta po definiciji potpuno uredjena grupna komunikacija (total order multicast) znaci?**

- Poruke se isporucuju procesima u Fifo redosledu

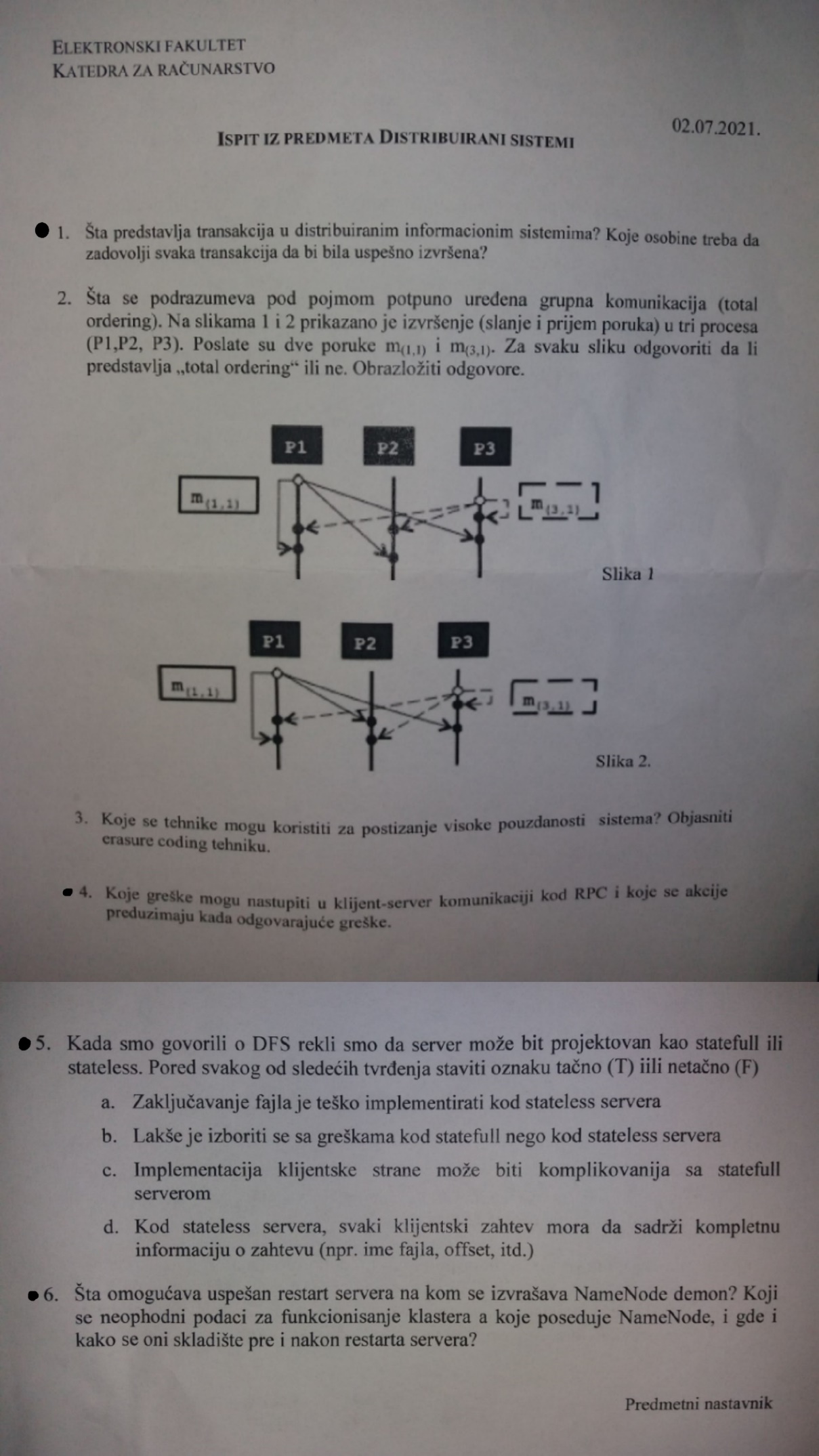
- Poruke se isporucuju procesima onako kako su se desile u realnom vremenu

- Poruke se isporucuju procesima u “happened-before” redosledu

- Poruke se isporucuju procesima u istom redosledu **T**

2.7.2021.

1. **Sta se podrazumeva pod pojmom potpuno uredjena grupna komunikacija(total ordering)? Na slikama je prikazano izvrsenje (slanje i prijem poruka) u tri proseca, P1, P2, P3. Poslate su dve poruke m(1,1) I m(3,1). Za svaku sliku odgovoriti da li predstavlja “total ordering” ili ne. Obrazloziti odgovore.**

****

Potrebno je obezbediti da se operacije ažuriranja obave na obe strane u istom redosledu. Neophodno je obezbediti potpuno uredjenu grupnu komunikacija (multicast), tj. operaciju kojom se sve poruke isporučuju u istom redosledu svim prijemnicima.

NAPOMENA: Multicast se odnosi na slanje poruke iz jednog izvora u više odredišta (prijemnika).

Prva slika nije total ordering, druga jeste:

* Poruke se prosledjuju svim procesima u grupi uključujući i samog sebe.
  + usvajamo da poruka koju proces šalje samom sebi stiže trenutno (tj. nema kašnjenja)
* Nakon slanja poruka redovi čekanja u procesima sadrže sledeće poruke:
  + P1: (m,1.1), (m,3.1)
  + P2: (m,1.1), (m,3.1)
  + P3: (m,1.1), (m,3.1)
* P1 će proslediti svima u grupi (multicast) potvrdu prijema za poruku (m,1.1) ali ne i za (m,3.1).
  + Zašto? Identifikator procesa P1 je niži od identifikatora procesa P2, a P1 je izdao poruku ažuriranja
  + 1.1 < 3.1
  + P1 neće poslati potvrdu za (m,3.1)dok se operacija (m,1.1) ne izvrši.
* P2 će proslediti svima u grupi potvrdu za (m,1.1) i (m,3.1)
  + Zašto? Identifikator procesa P2 je viši od identifikatora procesa P1
  + 1.1 < 3.1

…

* Ažuriranje koje se zahteva porukom (m,1.1) može biti obavljeno i u P1 i u P2 i u P3.
* Kada P1 završi sa ažuriranjem (m,1.1) *,* on šalje svim procesima u grupi potvrdu za (m,3.1) .
* Kada potvrdu prime P1, P2 i P3, znači da je potvrda za (m,3.1) stigla od svih procesa iz grupe
* Sada ažuriranje koje se zahteva porukom (m,3.1) može da se obavi u svim procesima.

?1.kolokvijum

1. **Postoje tri procesa P1,P2,P3 u grupi. Svaki process salje dve poruke svim ostalim procesima, ukljucujuci I sebe(multicast) sa sledecim vrednostima vremenskih markica:**

**-P1 salje m1 sa vremenskom markicom (9,1) I poruku m2 sa markicom (11,1)**

**- P2 salje m3 sa vremenskom markicom (9,2) I poruku m4 sa markicom (10,2)**

**- P3 salje m5 sa vremenskom markicom (10,3) I poruku m6 sa markicom (12,3)**

**Ako je potrebno ostvariti potpuno uredjenu grupnu komunikaciju, u kom redosledu ce svih 6 poruka biti primljeno u svakom procesu?**

(9.1), (9.2), (10.1), (10.2), (11.1), (12.3)

**Kristijanov algoritam**

\*7.9.2021.

1. **Casovnik na gradskom tornju u gradu A je bio pokvaren. Sad je popravljen, ali mora da bude podesen. Voz napusta stanicu u gradu A I dolazi u B brzinom od 200km/h. Cetri sata kasnije, voz se vraca sa informacijom da vreme po casovniku u gradu B iznosu 6:15. Ako se koristi Kristijanov algoritam, na koju vrednost treba da bude postavljen casovnik u gradu A?**

???????????????????????????????

1. **Sinhronizacija casovnika se vrsi Kristijanovim algoritmom. Poruka sa racunara A je poslata u trenutku 1:10.100 (po lokalnom vremenu rac.A). Server B je primio poruku u 1:15.000(po svom lokalnom vremenu), procesirao zahtev I poslao odgovor u kome stoji vreme 1:15.005, koja je stigla do rac.A u 1:10.150. Na koju vrednost ce biti postavljen casovnik u rac.A?**

****

T=1:15.005 + (1:10.150 – 1:10.100)/2 = 1:15.005 + 0:0.050/2 = 1:15.030

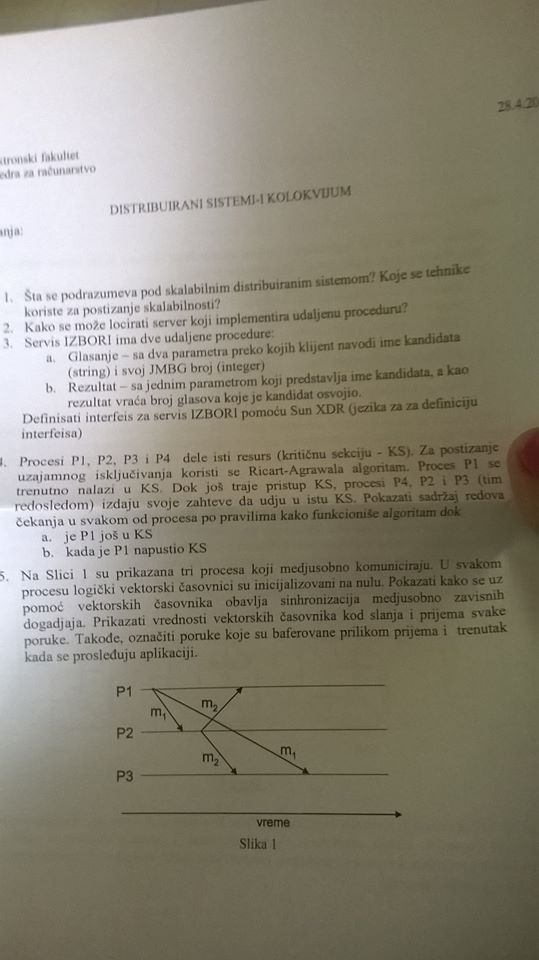
**Vektorski casovnik**

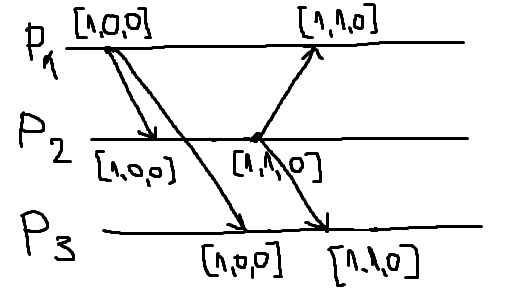
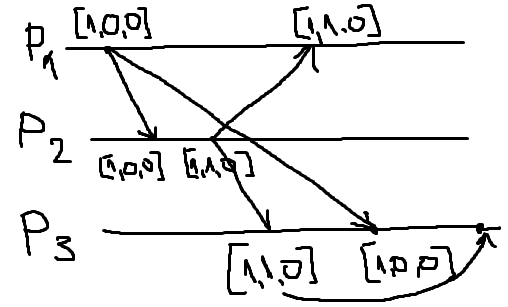
**2.7.2018. dodatno**

1. **Na slici su prikazana 3 procesa koji medjusobno komuniciraju.U svakom su logicki vektorski casovnici inicijalizovani na 0. Pokazati kako se uz pomoc vektorskih casovnika obavlja sinhronizacija medjusobno zavisnih dogadjaja. Prikazati vrednosti vek. casovnika kod slanja i primanja svake poruke. Takodje, oznaciti poruke koje su baferovane prilikom prijema i trenutak kada se prosledjuju aplikaciji.**

**Dodatno: navesti koji se uslovi testiraju kod prijema svake poruke.**

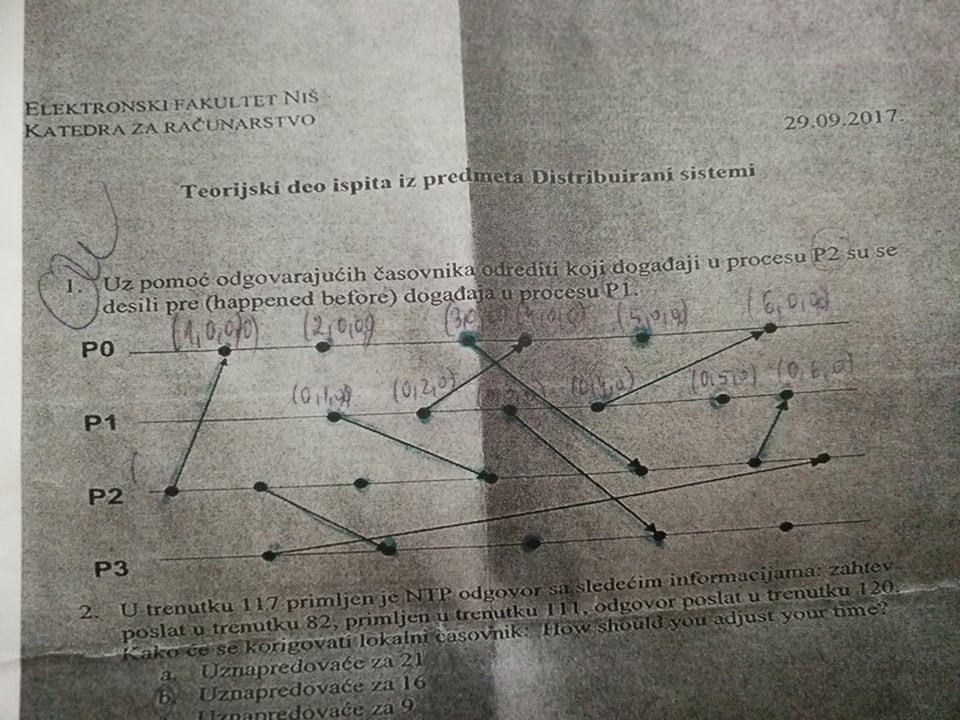
Kada Pj primi poruku poredi lokalni vektor sa primljenim, element po element, i postavlja element u lokalnom vektoru na veću od vrednosti: 

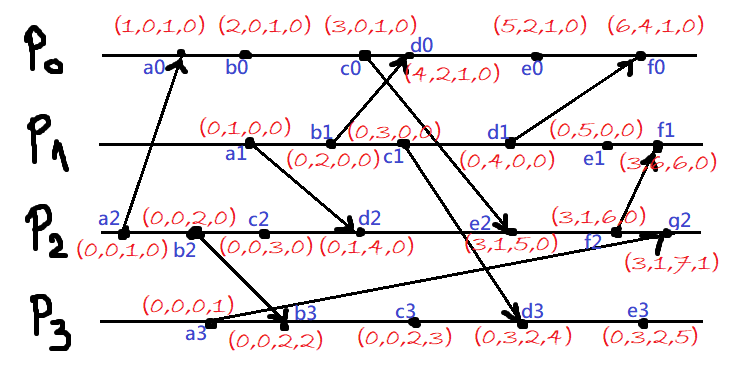




29.9.2017.

1. **Uz pomoc odgovarajucih casovnika, odrediti koji dogadjaji u procesu P2 su se desili pre(happened before) od dogadjaja u procesu P1.**





P2-> a2: (0,0,1,0) b2: (0,0,2,0) c2: (0,0,3,0) d2: (0,1,4,0) e2: (3,1,5,0) f2: (3,1,6,0) g2: (3,1,7,1)

P1-> a1: (0,1,0,0) b1: (0,2,0,0) c1: (0,3,0,0) d1: (0,4,0,0) e1: (0,5,0,0) f1: (3,6,6,0)

d2 -> f1

e2 -> f1

f2 -> f1

10.9.2018

1. **Vektorski casovnici:**
2. **Navesti pravila za implementaciju vektorskih casovnika.**

Pravila:

1. Vektor se inicijalizuje na 0 u svim procesima: Vi[j]=0, za i,j=1,…,N
2. Proces Pi inkrementira i-ti element vektora u lokalnom vektoru pre nego što obeleži događaj: Vi[i]=Vi[i]+1

Poruka se šalje iz procesa Pi zajedno sa Vi.

1. Kada Pj primi poruku poredi lokalni vektor sa primljenim, element po element, i postavlja element u lokalnom vektoru na veću od vrednosti



1. **Na osnovu vrednosti vektorskih casovnika, koji dogadjaj je prethodio dogadjaju ciji je vektorski casovnik (4, 2, 8, 5)?** Dati kratko obrazlozenje.

- (3, 1, 7, 7)

- (5, 1, 6, 2)

**- (4, 2, 8, 4)**

- (4, 3, 8, 5)

(4, 2, 8, 4) -> (4, 2, 8, 5), jer za svako i, vazi Vi[i] <= Vj[i]

30.9.2020

1. **a. Kako se uz pomoc vektorskih casovnika moze ostvariti uredjenje medjusobno povezanih dogadjaja?**

1.Pre emisije poruke m, proces Pi inkrementira svoj vektorski časovnik

Vi[i]=Vi[i]+1

Markica tm=Vi se šalje sa porukom m

2.Na prijemnoj strani poruka m se ne prosleđuje procesu Pj dok se ne zadovolje sledeća dva uslova

A) tm[i]=Vj[i]+1

Ovaj uslov obezbeđuje da proces Pj primi sve poruke koje je poslao proces Pi pre slanja poruke m

Neka je Vj[i]=5. Ovo znači da Pj zna da je Pi poslao 5 poruka i da od njega očekuje 6. poruku.

Ako je tm[i] = 8 tada poruke 6,7 još nisu stigle

B) tm[k]≤Vj[k], za svako k≠i

Obezbeđuje da proces Pj primi sve poruke koje je primio Pi pre slanja poruke m

3.Kada se poruka prosledi procesu, njegov lokalni časovnik se ažurira



**b.U DS tri procesa p0, p1, p2 medjusobno komunciciraju. U p0 vek. casovnik ima vrednost (8, 2, 4). Koja poruka iz p2 moze odmah da bude prosledjena aplikaciji u p0 ako je neophodno ocuvanje uredjena medjusobno zavisnih uredjaja? Obrazloziti kako se doslo do odgovora**

- (8, 2, 6)

**- (9, 3, 5)**

- (4, 1, 3)

- (1, 1, 5)

(9,3,5), jer za svako i, vazi Vi[i] <= Vj[i], a ako se povecava u p0, onda treba da je prvi 9.

9.6.2021

1. **Navesti pravila za implementaciju vek. casovnika?(bilo) Da li se na osnovu vek. casovnika moze utvrditi da li su dva dogadjaja medjusobno uslovljena? Obrazloziti.**

Moze se utvrditi:

* Poređenje vektora



* Za bilo koja dva događaja e i e’ važi sledeće
* 
* Ako se ne može uspostaviti relacija V(e)≤V(e’) ili V(e)≥V(e’) , tada su događaji konkurentni (tj. nisu međusobno uslovljeni)

**Fizicki casovnik**

1.2.2018.

1. **Objasniti kako se obavlja sinhronizacija fizickih casovnika ako:**
   1. **Postoji eksterni casovnik u sistemu**

Kristijanov algoritam – eksterna sinhronizacija

-U sistemu postoji jedna mašina sa WWV prijemnikom (server vremena – time server), a cilj je da sve ostale mašine budu sinhronizovane sa tom mašinom.

-Periodično svaka mašina šalje poruku serveru vremena tražeći informaciju o trenutnom vremenu.

-Server odgovara porukom u kojoj se nalazi informacija o vremenu, CUTC.

-Kada primi informaciju o vremenu klijent jednostavno može postaviti svoj časovnik na vrednost CUTC.

* 1. **Ne postoji eksterni casovnik u sistemu**

Berkeley algoritam – interna sinhronizacija

-Berekely UNIX koristi suprotan prilaz.

-Server vremena (u suštini time deamon) je aktivan.

-Obavlja prozivku svake mašine periodično da dozna koliko je vreme na datoj mašini.

-Na osnovu dobijenih odgovora, server izračunava srednje vreme i saopštava ostalim mašinama kako da podese svoje časovnike.

-Ovaj metod je pogodan za sisteme u kojima ne postoji WWV prijemnik

**Ricart & Agrawala algoritam**

28.4, 8.9.2020

1. **Procesi P1,P2,P3,P4 dele isti resurs (kriticnu sekciju-KS). Za postizanje uzajamnog iskljucivanja koristi se Richart-Agrawala alg. P1 se trenutno nalazi u KS. Dok jos traje pristup KS, P4, P2 I P3, tim redom, izadju svoje zahteve da udju u KS. Pokazati sadrzaj redova cekanja u svakom od procesa po pravilima kako funkcionise alg. dok:**
2. **Je P1 jos u KS**

P1: P4, P2, P3

P2: P3

P3: /

P4: P2, P3

1. **Kada je P1 napustio KS**

P1: /

P2: P3

P3: /

P4: P2, P3

**\*1.kolokvijum**

1. **Za izbor koordinatora koristi se “Bully” algoritam. Sta ce se desiti ako dva procesa jednovremeno pokrenu izbor novog koordinatora?**

**Pitala sam profesorku**

**Greske**

8.7.2020

1. **a) Definisati pojmove defekt (fault), greška (error) i otkaz (failure) i navesti primer koji ilustruje ova tri pojma.**

**Defekt** je trajni ili privremeni nedostatak koji se javlja u nekoj hardverskoj ili softverskoj komponenti

**Greške** su manifestacija defekta i predstavljaju odstupanje vrednosti podatka od očekivane (defekt je uzrok greške)

**Otkaz** nastupa kada sistem više ne može da obavlja funkciju za koju je projektovan

Npr. u softverskom sistemu, nekorektna instrukcija u programu koja dekrementira vrednost neke promenljive, umesto da je inkrementira, je defekt. Kada se ova instrukcija izvrši ona će generisati (upisati) pogrešnu vrednost promenljive. Ako druge naredbe u programu koriste tu vrednost, ceo sistem će odstupiti od željenog ponašanja. U ovom primeru nekorekta instrukcija je defekt. Individualna vrednost promenljive je greška, a otkaz je ponašanje sistema koje je posledica greške.

Napomenimo da ako se generisana pogrešna vrednost promenljive nikad ne koristi u sistemu, neće doći do otkaza. takodje, ako se defektna instrukcija ne izvrši neće doći do otkaza.

b) **Distribuirani sistem se sastoji od 4 replicirana servera. Pouzdanost svakog pojedinačnog servera je 0.9.**

* 1. **Ako je sistem projektovan tako da može da funkcioniše ako je bilo koji od servera u funkciji, kolika je pouzdanost celog sistema?**

Odgovor je u prezentacijama od 27.04-30.04, slajdovi 7 i 8. Pouzdanost sistema je P=1-(0.1)4 =0.9999

* 1. **Ako je sistem projektovan tako da može da funkcioniše samo ako su sva četiri servera u funkciji (ispravna), kolika je pouzdanost celog sistema?**

Pouzdanost sistema je P=(0.9)4 =0.6561

11.10.2017

1. **Objasniti problem dve armije I kako se on moze prevazici.**

-Dve divizje, A i B, jedne armije, treba da koordinišu napad na drugu armiju, C.

-A i B imaju po 2000 vojnika, a armija C 3000.

-Ako se A i B dogovore da zajedno napadnu, mogu da pobede C

A i B su fizički odvojene i koriste kurira za razmenu poruka.

* + A šalje poruku B u kojoj piše “izvršimo napad u zoru”.
  + B prima poruku i slaže se sa predlogom (šalje ok – potvrdu)
  + Kurir stiže u A sa povratnom porukom (potvrdom), ali tada A zaključuje da B ne zna da li je A primio potvrdu i neće biti siguran da li da obave napad.
  + A može da odluči da pošalje poruku B-u “primio sam potvrdu”, ali tada A neće biti siguran da li je ta poruka stigla u B, itd.

Problem je poznat pod nazivom problem višestrukih potvrda.

* Ne postoji protokol koji može da reši problem nepouzanih kanala!
* Može se pokazati da generali plave armije nikada ne mogu postići konsenzus

10.9.2019, 1.2.2018.

1. **Koji tipovi gresaka u odnosu na trajanje postoje? Dati primer za svaku od njih. Objasniti razliku izmedju “mirnih” I Vizantijskih gresaka. Koju vrstu gresaka je teze detektovati? Obrazloziti.**

* Kategorizacija grešaka u odnosu na trajanje
  + Prolazne (transient faults)
    - Pojave se jednom i nestanu
      * Npr. Istekao time out za prenos poruke, ali nakon ponovnog slanja poruka je uspešno otposlata
  + Peridične (intermittent)
    - Greška se pojavi, zatim nestane, pa ponovo pojavi. Najnezgodniji tip grešaka
      * Npr greška u mrežnom kablu kod koga postoji neki prekid
  + Stalne (permanent)
    - Greške koje postoje sve dok se komponenta ne zameni ispravnom
      * Greška u hard disku, pregorela komponenta, bagoviti sw.
* Druga klasifikacija grešaka:
  + Tvz. “mirna” (fail-silent, ili fail-stop) greška– komponenta prestaje sa radom i ne generiše nikakv izlaz ili generiše poruku o grešci
  + Vizantijske greške – komponenta nasatavlja sa radom i generiše pogrešan rezultat

Teze je detektovati Vizantijske greske jer generisu rezultat koji je neispravan, dok mirne ne generisu rezultat ili daju poruku o greskci.

2.7.2021.

1. **Koje se tehnike mogu koristiti za postizanje visoke pouzdanosti sistema? Objasniti erasure coding tehniku.**

Kako sistem učiniti otpornim na greške?

Opšti prilaz u projektovanju sistema otpornih na greške je redundansa.

Može se primeniti na nekoliko nivoa

* + - Informaciona
      * Otpornost na greške se postiže repliciranjem ili kodiranjem podataka (npr. Hammingovi kodovi, ABFT)
    - Vremenska
      * Tolerantnost na greške se postiže izvršenjem operacija više puta u vremenu. Npr. timeouti i retransmisija poruka.
      * Ova vrsta redundantnosti je korisna u slučaju prolaznih ili periodničnih grešaka.
    - Fizička redundansa fizička redundansa je široko prihvaćena tehnika za postizanje visoke pouzdanosti tako što će se sistem učiniti tolerantnim na defekte. Ova “tehnika” je prisuta u prirodi (npr. sisari imaju dva oka, dva uha, dve plucna krila, dva bubrega,...); u avio industriji – 747 ima 4 motora a može da leti sa tri; u sportu – više sudija u slučaju da jedan previdi dogadjaj..

Ne postoji jedan opšte prihvaćeni metod za postizanje otpornosti na greške.

Najjednostavnija varijanta- TMR

* + - * hardverska
        + Bavi se uređajima a ne podacima. Hardver se umnožava da bi sistem mogao tolerisati otkaze. Npr. RAID diskovi (redundant array of independent disks) i backup servri.
      * softverska
        + replicirani procesi

Tehnika za postizanje otpornosti na greške (fault tolerance)

Tehnika kodiranja kod koje se skup od k podataka kodira skupom od n, n>k podataka tako da se na osnovu bilo kog skupa od k podataka mogu regenerisati originalni podaci.

* 1. (n,k) erasure code omogućava tolerisanje n-k grešaka.
  2. Tehnika koja je široko prihvaćena kod HDF (Hadoop file system)
     1. Hadoop je sistem za skladištenje, pretraživanje i procesiranje velike količine podtaka (big data)

**Primer upotrebe:**

* Posmatrajmo fajl veličine *M*
* Podeliti fajl na k blokova veličine *M*/*k*
* Primeniti *(n, k)* code na ovih *k* blokova da bi se dobilo *n* blokova, svaki veličine *M/k*.
  + Na taj način fajl je proširen *n/k* puta
  + Potrebno je da *n* bude veće ili jednako k.
    - Ako je *n* jednako *k*, fajl je samo podeljen na blokove, ali nema nikakvog kodiranja
* Bilo koji podskup *k* od *n* blokova se može uzeti da bi se rekonstruisao fajl
  + to znači da kod može tolerisati (n-k) grešaka

\*? 8.9.2020

1. **Koje tipove gresaka moze tolerisati sistem koji koristi TMR (trostruku hardversku redudansu)? Obrazloziti.**

ako hardver repliciramo 3 puta, pouzdanost sistema će se povećati

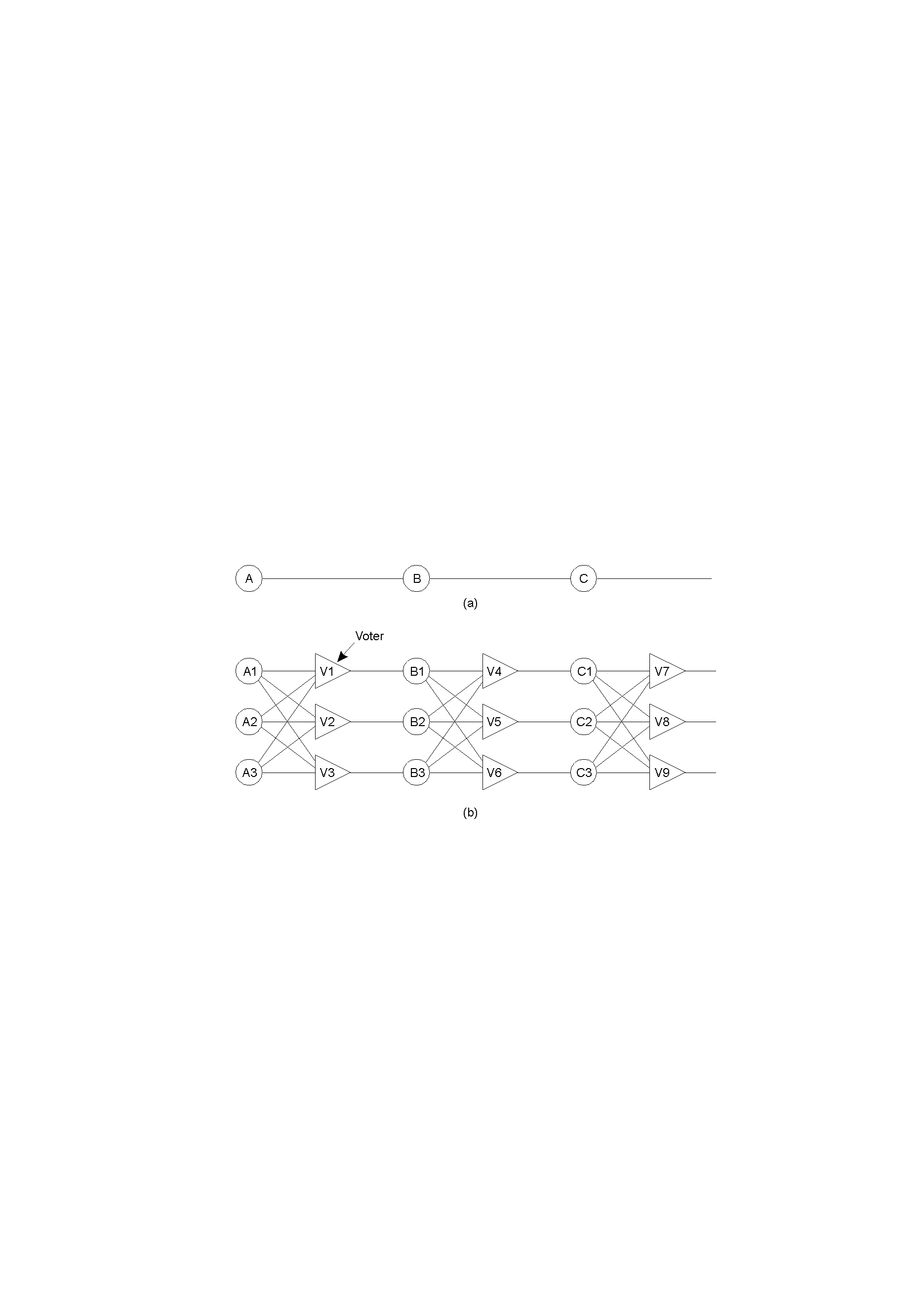
-Sada postoje tri paralelna puta.

-Pouzdanost na jednom putu je Ri, a nepouzdanost Qi=1-Ri

-TMR tehnika se primenjuje u sprezi sa glasačima (voterima)

a) Sistem bez redundantnosti: signal prolazi kroz uređaje A, B i C sekvencijalno. Ako je jedan od uređaja neispravan, konačni rezultat će verovatno biti nekorektan.

b) TMR: svaka komponenta je replicirana 3 puta. Izlaz iz svake komponente se vodi na glasač (voter). I voteri su replicirani. Ako su tri ili dva ulaza ista, izlaz je jednak ulazu. Ako su tri ulaza različita, izlaz je nedefinisan.



\*11.9.2017

1. **U procesu mnozenja dve matrice koriscena je tehnika ABFT za detekciju i korekciju gresaka. Dobijen je sledeci rezultat: 1 2 3 6**

**2 3 4 9**

**3 3 3 8**

**6 7 10 23**

1. Utvrditi da li je nastupila greska u toku izracunavanja
2. Ako je nastupila greska, naci njenu lokaciju
3. Izvrsiti korekciju greske, ako je greska nastupila

11.9.2017

1. **Kada nastupi greska u DS, potrebno je izvrsiti opiravak od greske i vratiti sistem u korektno stanje. Koje strategije za oporavak od greske se koriste u DS? Koje su prednosti i nedostaci ovih strategija?**

To se može postići na dva načina:

\*Povratak u stanje pre nastupanja greške (Backward Recovery):

-vratiti sistem u neko prethodno korektno stanje korišćenjem tačaka provere (checkpoints), a zatim nastaviti sa radom

-npr retransmisija u slučaju gubitka paketa ili greške u primljenom paketu

-Nedostatak oporavka sa povratkom u nazad: beleženje stanja sistema ubacivanjem tačaka provere može biti skupo, uprkos ceni, ova tehnika se češće koristi od forward.

\*Prelazak u novo stanje (Forward Recovery):

-prevesti sistem u korektno novo stanje nakon čega se može nastaviti sa radom

-Erasure coding, (n,k)–tehnika kodiranja podataka koja omogućava korekciju u slučaju da je nastupila greška (skup od k poruka se kodira u n poruka tako da bilo kojih k poruka na odredištu mogu da se iskoriste da se rekonstruiše k originalnih poruka).

-da bi ova tehnika funkcionisala neophodno je predvideti sve potencijalne greške

kada nastupi greška, mehnizam za oporavak zna šta treba da uradi da bi doveo sistem u naredno korektno stanje.

2.7.2018.

1. **U grupi postoje 5 repliciranih procesa. Ako mogu nastupiti samo mirne greske, koliko maksimalno procesa moze otkazati, a da se ipak dobije korektan rezultat? Ako mogu nastupiti greske Vizantijskog tipa, koliko procesa moze maksimalno otkazati a da se ipak dobije korektan rezultat? Sta ako mogu nastupiti greske Vizantijskog tipa, a procesi moraju postici konsenzus? Koliko u ovom slucaju maksimalno procesa moze otkazati? Svaki odg. obrazloziti.**

Mirne -> m+1=5, m=4

Vizantijske ->2m+1=5, m=2

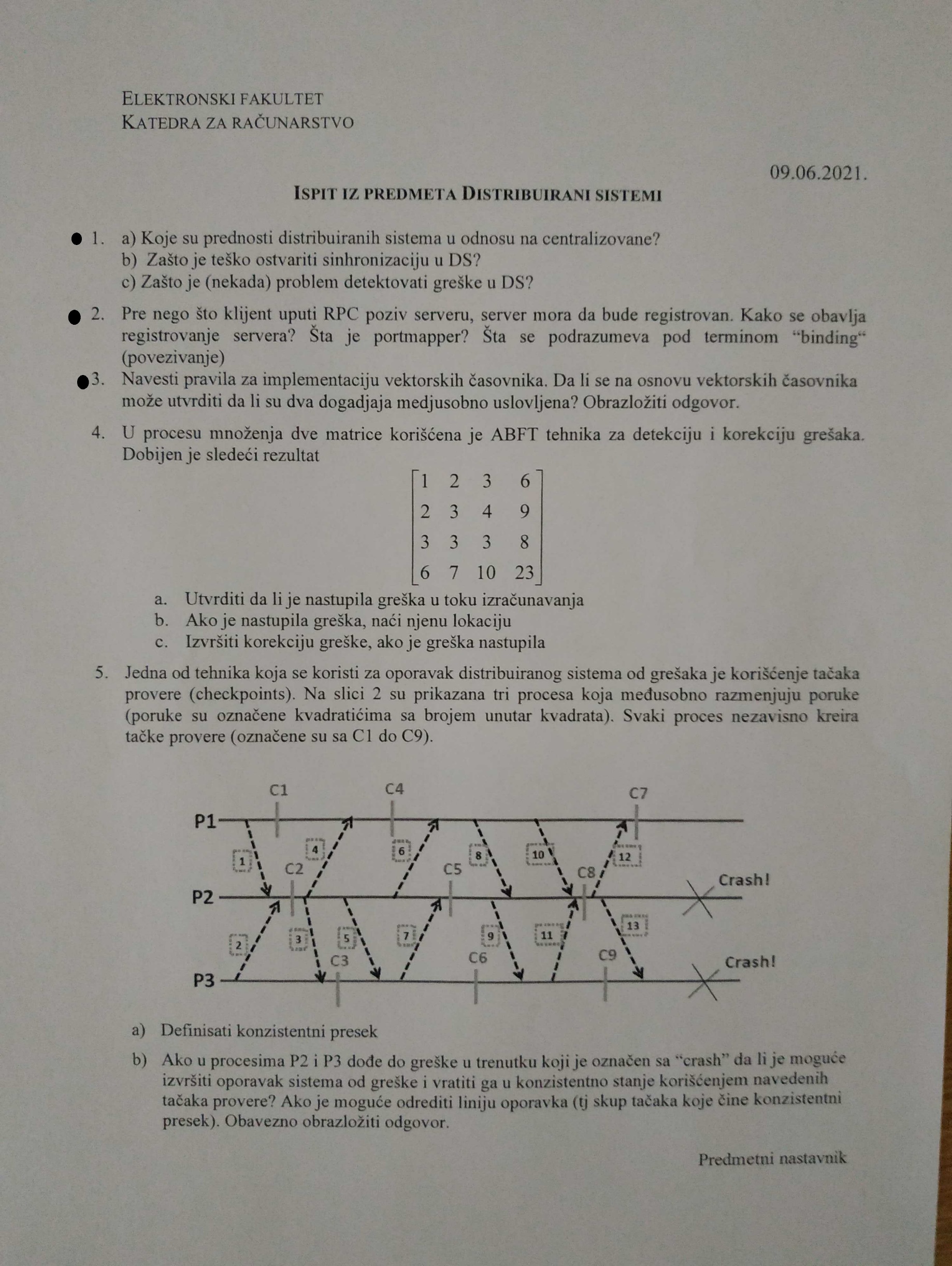
Vizantijske + konsenzus -> 3m+1=5, m=1

\*2.7.2021

1. **Koje greske mogu nastupiti u klijent-server kod RPC-a I koje se akcije preduzimaju kada one nastupe?**

\*9.6.2021

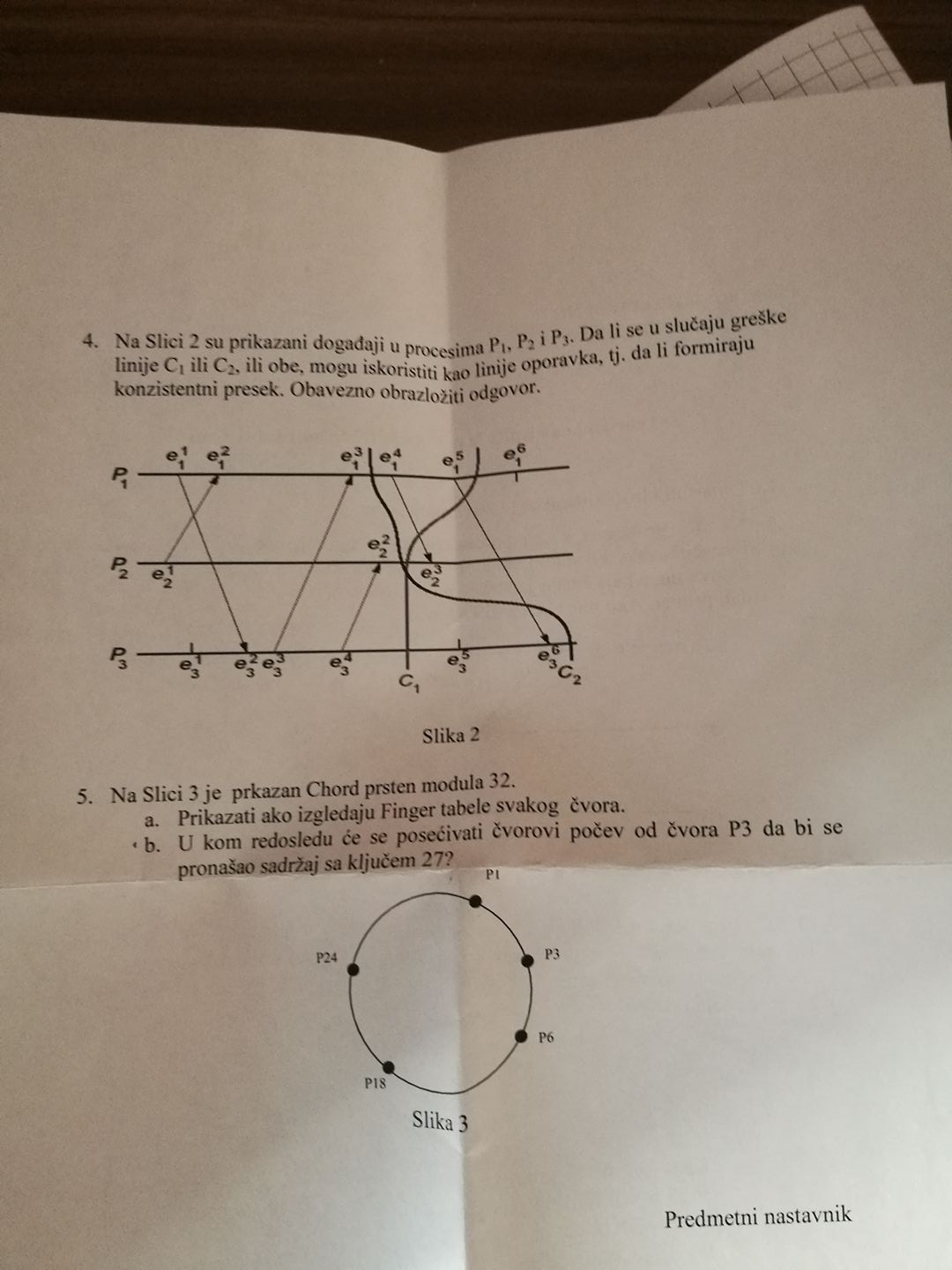
1. **Jedna od tehnika koje se koriste za oporavak DS od gresaka je koriscenje tacaka provere(check points). Na slici su prikazana 3 procesa koja medjusobno razmenjuju poruke(poruke su oznacene kvadraticima sa brojem unutar kvadrata). Svaki process nezavisno kreira tacke provere(od C1 do C9).**

****

* 1. **Definisati konzistentni presek**
  2. **Ako u procesima P2 I P3 dodje do greske u trenutku koji je oznacen sa “crash”, da li je moguce izvrsiti oporavak Sistema od greske I vratiti ga u konzistentno stanje koriscenjem navedenih tacaka provere? Ako je moguce, odrediti liniju oporavka(tj. skup tacaka koji cine konzistentni presek). Obrazloziti.**

26.6.2017

1. **Na slici su prikazani dogadjaji u procesima P1,P2,P3. Da li se u slucaju greske linije C1 ili C2 ili obe, mogu iskoristiti kao linije oporavka, tj. da li formiraju konzistentni presek? Obrazloziti.**

****

* U DS potrebno je da se svi procesi vrate u stanje odakle je moguće izvršiti oporavak, tj. potrebno je pronaći liniju oporavka (konzistentni presek ).



* Skup tačaka provere C je konzistentan ako za sve događaje e i e’ važi

to znči da ako je npr. proces P zabeležio prijem poruke iz procesa Q, tada u skupu mora postojati i tačka u kojoj je proces Q zabeležio slanje poruke procesu P (poruka je morala da stigne od nekog).

C1 moze, C2 ne moze

8.7.2020.

1. **U distribuiranom sistemu klijent koristi RPC za komunikaciju sa serverom. Klijent šalje zahtev serveru (npr. operation x( )) sve dok ne dobije odgovor od servera.** 
   1. **Koja RPC semantiku u slučaju greške je u ovom slučaju implementirana?**

U pitanju je at least one semantika jer klijent šalje zahtev sve dok ne primi odgovor od server

* 1. **Objasniti kako mora da bude implementiran server da bi se ovakava semantika obezbedila? Navesti bar dve mogućnosti.**

Dve su mogućnosti na strani servera: Da implementira idempotentnu proceduru operation x( )

ili da server bude statefull i da vodi računa o klijentskim zahtevima i prepoznaje duplikate

zahteva i u tom slučaju vrši samo retransmisiju poruke u kojoj je odgovor (ne izvršava ponovo

proceduru operation x( ) )

\*10.9.2018.

1. **Objasniti sledece semantike kod poziva udaljenih procedura:**

**a.mozda(maybe)**

**b.At-least-once**

**c.At-most-once**

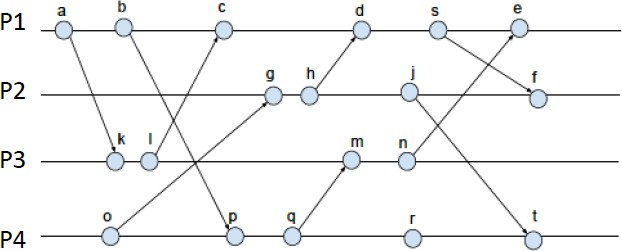
a. ???

b.”Najmanje jednom”- oznacava da funkcija moze da se izvrsi najmanje jednom ili vise puta

c.”Najvise jednom”-ili jednom ili nijednom

8.7.2020.

1. **Na slici su prikazana 4 distribuirana procesa koji međusobno komuniciraju. Prikazati konzistentni presek koji uključuje događaj j i najmanji mogući broj događaja u ostalim procesima.**



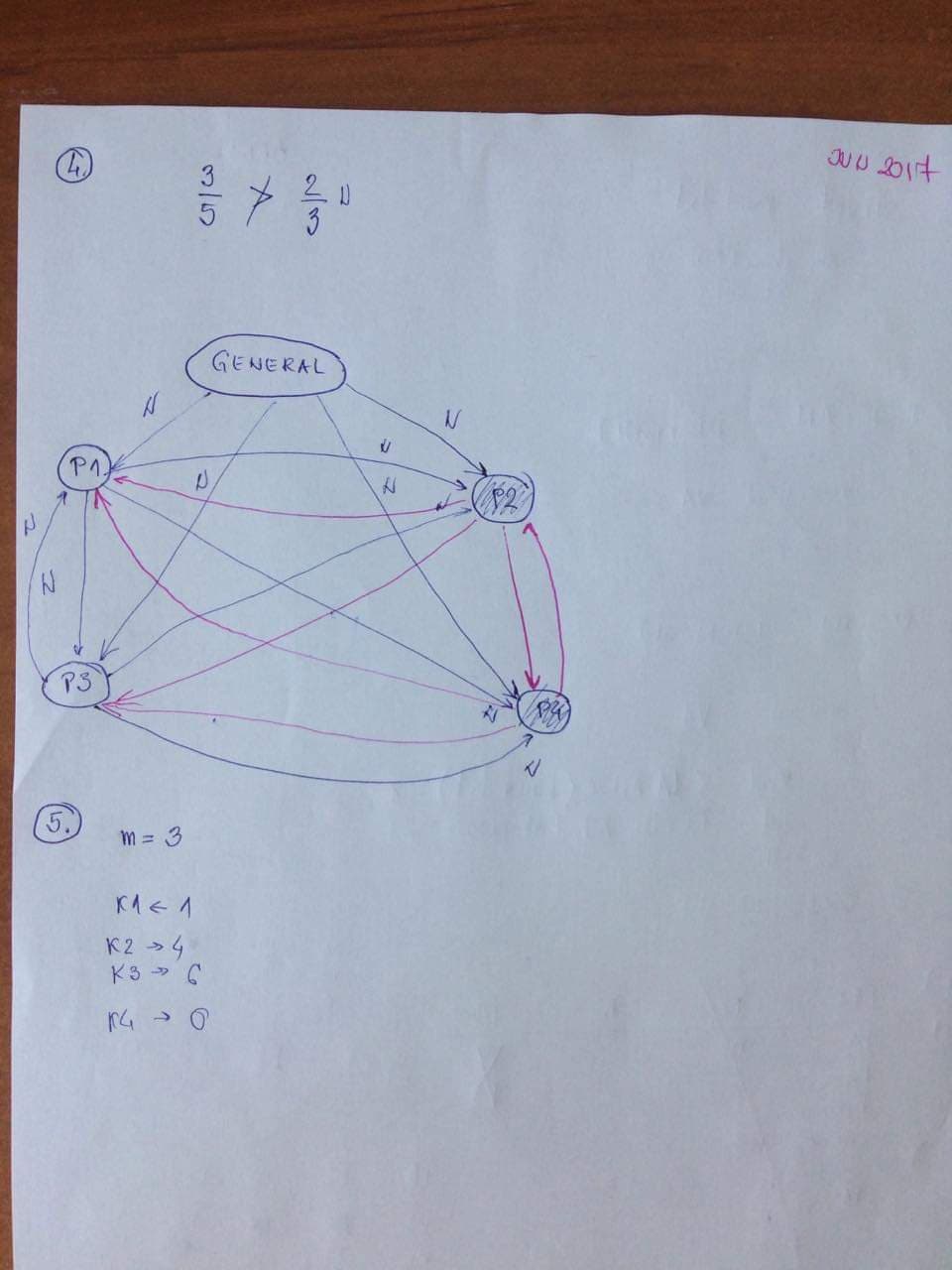
Uslov koji mora biti zadovoljen da bi imali konzistentni presek je da ako je u nekom procesu zabeležen prijem neke poruke iz drugog procesa, tada mora konzistentnim presekom biti zabeležen trenutak slanja te poruke. U konkretnom primeru, događaj g predstavlja prijem poruke iz procesa P4, pa konzistentni presek mora da obuhvati događaj O u procesu P4.

16.6.2017.

1. **U DS postoje 5 procesa koji treba da se usaglase kod donosenja odluke. Ako dva generisu pogresne rezultate, da li je moguce doneti odluku? Ilustrovati primerom sa 1 generalom I 4 porucnika.**

Više od 2/3 generala mora biti lojalno. Za *m* generala izdajnika zahteva se najmanje 3m+1 učesnika (2m+1 lojalnih generala), tako da za 5 procesa i dva iznadnika nije moguce doneti odluku.

Da bi bilo moguce sa dva izdajnika doneti odluku, potrebno nam je najmanje 3\*2 + 1, tj. 7 ucesnika.

****

**Konzistentnost i replikacija**

\*11.6.2017

1. **Koje vrste replika postoje? Koje se informacije mogu proslediti replikama kada se obavlja azuriranje?**

Tri tipa replika:

* + Permanentne replike
  + Replike inicirane od strane servera
  + Replike inicirane od strane klijenta

\*1.2.2018.

1. **Navesti moguce nacine azuriranja replika.**

\*30.9.2020

1. **Server je repliciran na 20 cvorova. Za odrzavanje konzistencije koristi se protocol baziran na kvorumu.** 
   1. **Ako se citanje obavlja samo sa jedne kopije, koliko kopija se najmanje mora azurirati u alucaju modifikacije, da bi se obezbedila korektnost?**
   2. **Ako se prilikom upisa azurira 11 kopija, koliko najmanje kopija se mora procitati da bi se obezbedila korektnost?**

**Obrazloziti.**

?29.9.2017.

1. **P1 I P2 se konkurentno izvrsavaju. Navesti sve rezultate tri print naredbe, npr print x=0, print y=0, print z=0, koji nisu moguci pod uslovima striktne egzistencije.**

**P1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_P2**

**A=1 print y**

**x=A print x**

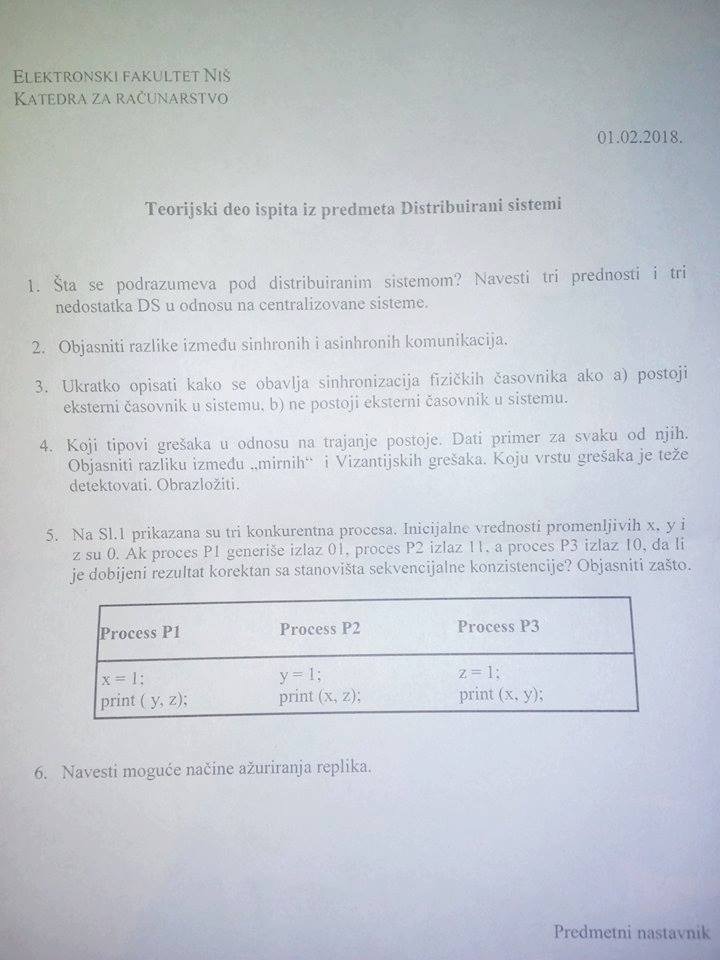
**y=A print z**

**z=y**

001 i 101

1.2.2018.

1. **Na slici su prikazana tri konkurentna procesa. Inicijalne vrednosti za x, z i y su 0. Ako P1 generise izlaz 01, P2 11, a P3 10, da li je dobijeni rezultat korektan sa stanovista sekvencijakne egzistencije? Obrazloziti.**

****

P1: y, z -> 01

P2: x, z -> 11

P3: x, y -> 10

Zadovoljava jer svi citaju iste vrednosti.

?11.10.2017

1. **Procesi P1, P2 I P3 se konkurentno izvrsavaju. Navesti sve rezultate tri print naredbe, npr print x = 0, print y = 0, print z = 0 koji nisu moguci pod uslovima sekvencijalne egzistencije.**

**P1\_\_\_\_\_\_\_\_P2\_\_\_\_\_\_\_\_\_P3\_\_**

**A=1**

**x=A y=x print y**

**z=y print x**

**print z**

x y z

0 0 0

0 0 1

0 1 0

0 1 1

1 0 0 -ne mogu

1 0 1 -ne mogu

1 1 0

1 1 1

100

101

Zasto???

26.6.2017.

1. **Cetiri procesa P1, P2, P3, P4 pristupaju zajednickoj promenljivoj x. Svaki process ima sopstvenu kopiju skladista podataka u kojoj se nalazi ova promenljiva. Inicijalna vrednost promenljive x je 0. U procesima od P1 do P4 se izvrsavaju sledece naredbe:**

P1 P2 P3 P4

x=3 x=1 y=x z=x

y=4\*y+x z=4\*z+x

1. **Kada sva 4 procesa okoncaju izvrsenje, da li je moguce da je y=7 i z=13 u slucaju sekvencijalno konzistentnog skladista podataka? Obrazloziti.**

Nakon Write operacije, procesi NE vide isto.

P3: P2, P1

P4: P1, P2

1. **Da li je moguce da je y=7 I z=13 u slucaju kauzalno konzistentnog skladista podataka? Obrazloziti.**

Moguce je, jer nigde nemamo Read.

X=3, y=1 => y=4\*1+3=7

X=1, z=3 => z=4\*3+1=13

P1: - m(x)3

P2: - m(x)1

P3: - R(x)1 – m(y)1 – R(x)3 – R(y)1 - m(y)7

P4: - R(x)3 – m(z)3 – R(x)1 – R(z)3 - m(z)13

\*7.9.2021.

1. **Nad objektom x obavljene su sledece operacije u distribuiranom skladistu:**

* **P1: write(x)A**
* **P2: write(x)B, read(x)C**
* **P3: read(x)B, read(x)A, write(x)C**
* **P4: read(x)B, read(x)A**

**Nacrtati kako izgleda stvari redosled dogadjaja ako je skladistenje podataka sekvencijalno konzistento.**

11.9.2017

1. **Objasniti protokole konzistencije zasnovane na postojanju primarne kopije. Koji je najjaci vid konzistencije koji se ovim protokolima moze postici?**

* Protokoli konzistencije opisuju implementaciju određenog modela konzistencije.
* Tri vrste:
  + Protokoli zasnovani na postojanju primarne kopije (i backup kopija)
    - Write operacija se može izvršiti samo na primarnoj kopiji
  + Protokoli sa više ravnopravnih replika.
    - write operacija može biti inicirana u bilo kojoj replici.
  + Keš koherentni protokoli
    - inicirani od strane klijenta, a ne servera
* Kod ovih protokola svaki podatak X u skladištu podataka ima pridruženi primar koji je zadužen za koordinaciju upisa u X.
* Može se napraviti razlika u odnosu na to da li je primar fiksiran na udaljenom serveru - Remote-write protokoli (protokoli sa udaljenim upisom)
* ili se write operacija može obaviti lokalno nakon što se primar pomeri na proces koji je inicirao write - Local-write protokoli (protokoli sa lokalnim upisom)
* Najjednostavniji protokol baziran na postojanju primarne kopije je protokol kod koga se sve read i write operacije obavljaju na udaljenom serveru.

\*8.9.2020.

1. **a. Sta je striktna konzistencija I zasto je nije moguce postici u DS?**

**Striktna konzistencija:**

-Najjači model konzistencije:

*-Bilo koja operacija nad podatkom X vraća rezultat poslednje write operacije nad X.*

-Definicija je prirodna i očigledna, ali implicitno usvaja postojanje globalnog vremena, tako da određivanje poslednje write operacije bude nedvosmisleno.

-U sistemu u kome se podaci nalaze na više mašina, kojima može pristupati više procesa, situacija je mnogo složenija:

\*Pretpostavimo da proces Pi ažurira vrednost promenljive x sa 4 na 5 u trenutku t1 i šalje novu vrednost svim replikama u grupi (multicast)

\*Proces Pj ‚čita (read) vrednost promenljive x u trenutku t2 (t2 > t1).

\*Proces Pj treba da pročita vrednost 5, bez obzira kolika je razlika (t2-t1).

\*šta ako je t2-t1 = 1 nsec i koriste se optička vlakna izmedju dve host mašine (na kojima se izvršavaju ova dva procesa) koje se nalaze na rastojanju od 3 metra?.

\*Poruka kojom se zahteva ažuriranje morala bi da putuje 10 puta brže od brzine svetlosti

\*Po Ajnštajnovoj specijalnoj teoriji relativiteta to nije moguće!

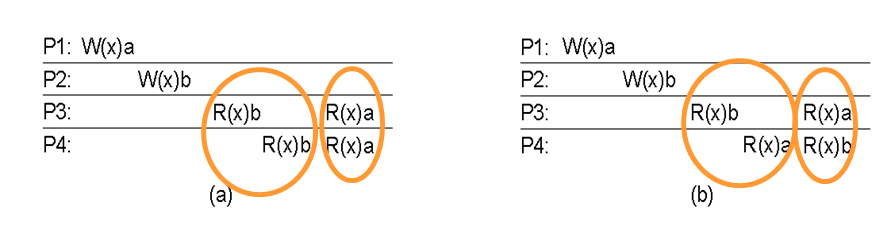
-Zaključak: striktnu konzistenciju u distribuiranom sistemu je nemoguće postići!

**b.Definisati sekvencijalnu I kauzalnu konzistenicu. Dati jedan primer koji ilustruje razlike.**

-Sekvencijalna konzistencija je slabiji model od striktne:

-Rezultat bilo kog izvršenja je isti kao da su (read i write) operacije svih procesa na skladištu podataka izvršene u nekom sekvencijalnom redosledu i operacije svakog pojedinačnog procesa pojavljuju se u ovoj skvenci u redosledu koji je određen njihovim programom.

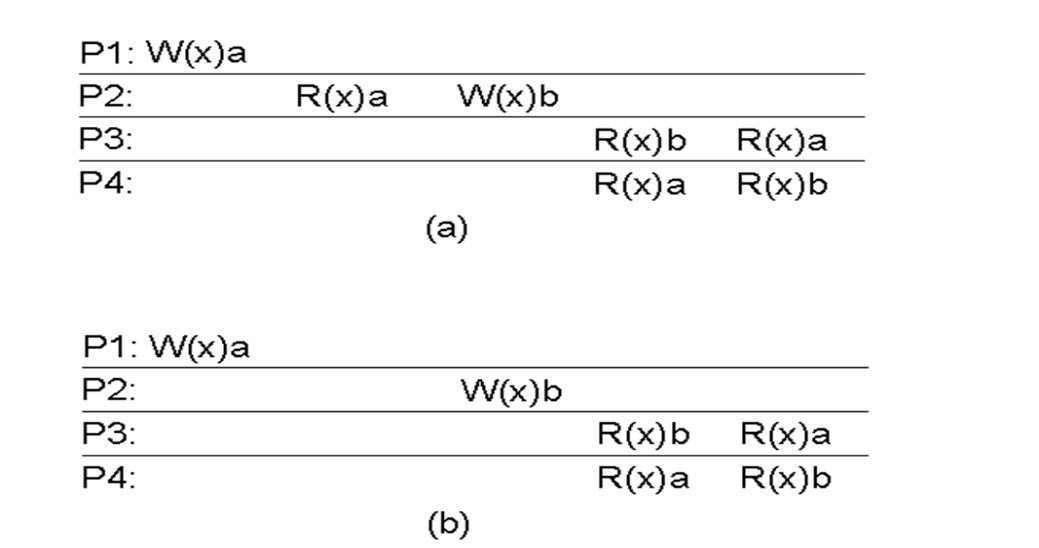
-Def. kaže da kada se procesi izvršavaju konkurentno na (moguće) različitim mašinama, bilo koje validno preplitanje read i write operacija je prihvatljivo, ali svi procesi vide isto preplitanje operacija.



Sekvencijalno konzistentno skladište Narušena sekvencijalna konzistentnost jer svi procesi ne vide na isti način operacije.

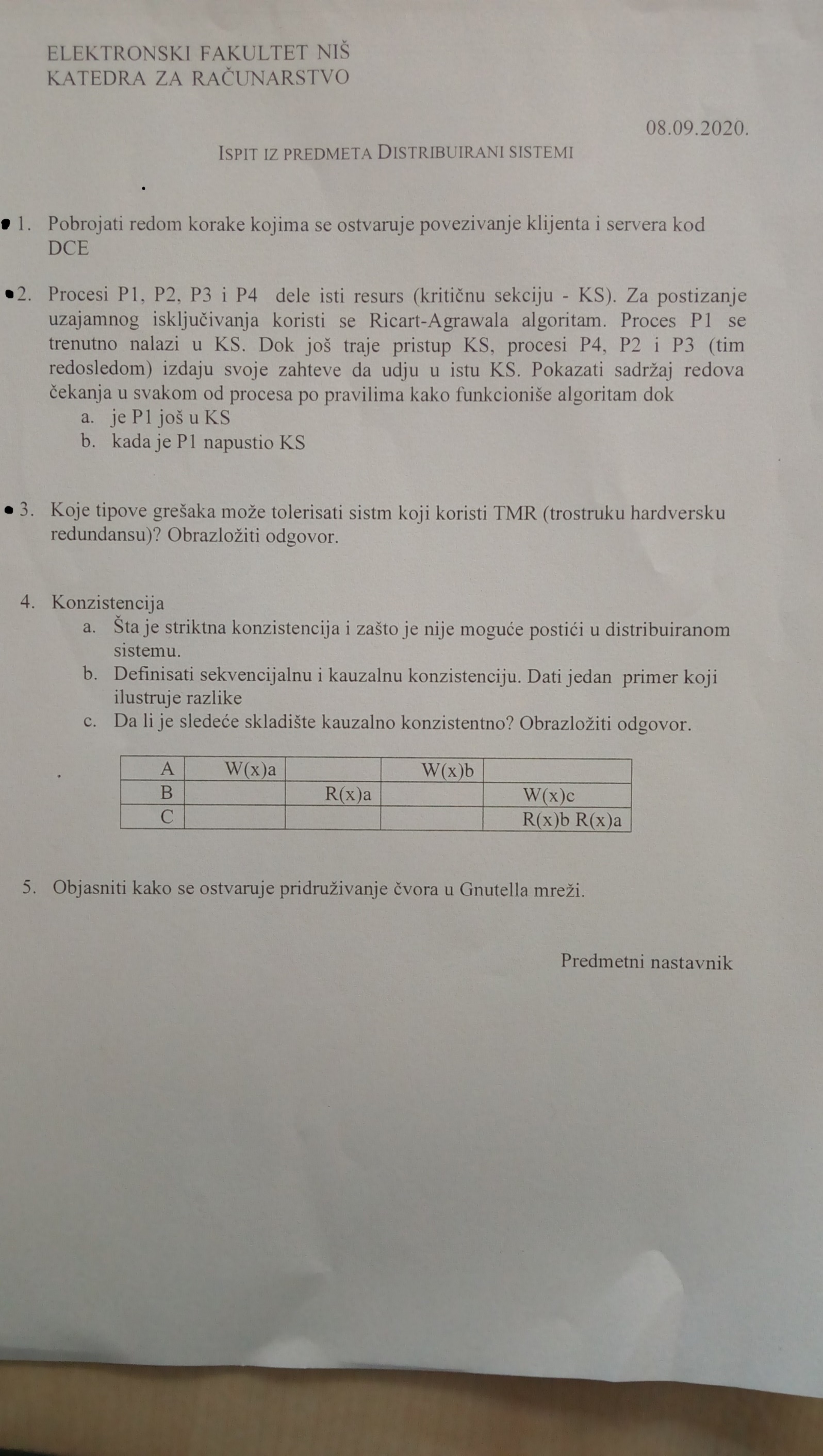
-Uslovna (kauzalna) konzistencija- Upisi (write) koji su potencijalno uslovljeni moraju se videti u svim procesima u istom redosledu

-Konkurentni upisi se mogu videti u različitom redosledu u različitim procesima



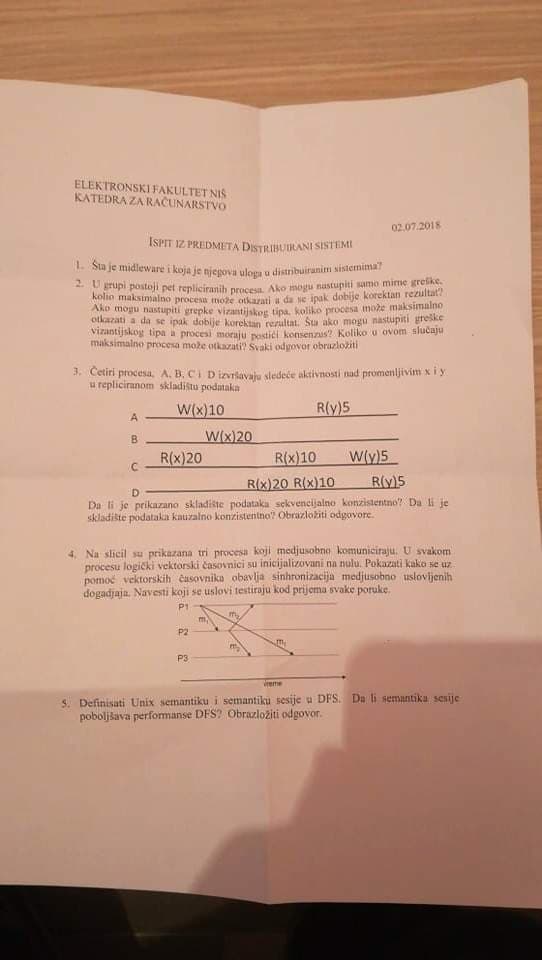
1. nije ispoštovana uslovna konzistencija
2. ispoštovana je uslovna konzistencija jer W(x)a i W(x)b nisu medjusobno uslovljeni

**c. Da li je sledece skladiste kauzalno konzistentno? Obrazloziti.**



\*2.7.2018.

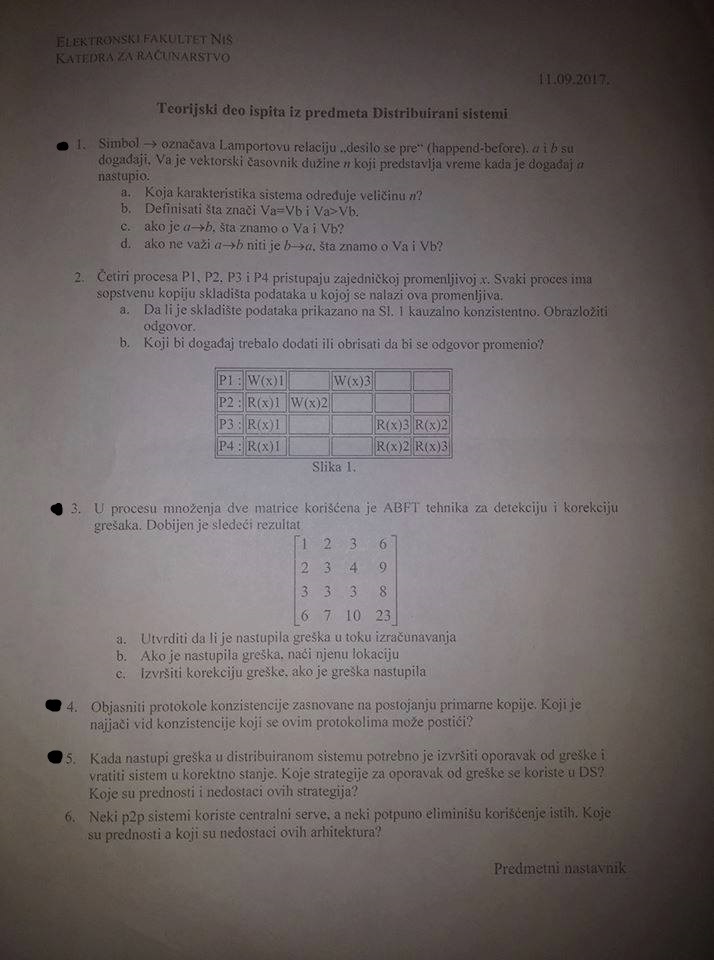
1. **Cetiri proseca zivrsavaju sledece aktivnosti nad x I y u repliciranom skladistu podataka:**



**Da l je prikazano skladiste sekvencijalno konzistentno? Da li je kauzalno konzistentno? Obrazloziti.**

\*11.9.2017

1. **Cetiri procesa P1, P2, P3, P4 pristupaju zajednickoj promenljivoj x. Svaki process ima sopstvenu kopiju skladista podataka u kojoj se nalazi ova promenljiva.**

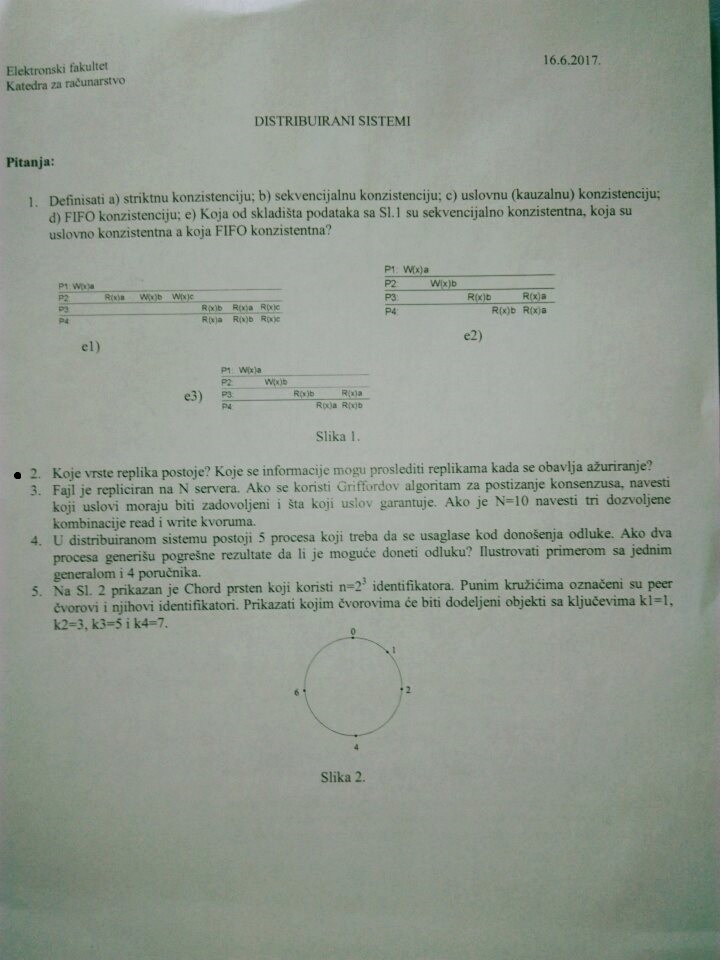


* 1. **Da li je skladiste podataka sa slike kauzalno konzistentno? Obrazloziti.**
  2. **Koji dogadjaj treba dodati/obrisati da bi se odgovor promenio?**

16.6.2017.

1. **Definisati: a.striktnu, b.sekvencijalnu, c.kauzalnu, d.FIFO konzistenciju.**

**e.Koja od skladista podataka sa slike su sekvencijalno, koja kauzalno, a koja FIFO konzistenta?**



**a. Striktna konzistencija:**

-Najjači model konzistencije:

*-Bilo koja operacija nad podatkom X vraća rezultat poslednje write operacije nad X.*

-Definicija je prirodna i očigledna, ali implicitno usvaja postojanje globalnog vremena, tako da određivanje poslednje write operacije bude nedvosmisleno.

-U sistemu u kome se podaci nalaze na više mašina, kojima može pristupati više procesa, situacija je mnogo složenija:

\*Pretpostavimo da proces Pi ažurira vrednost promenljive x sa 4 na 5 u trenutku t1 i šalje novu vrednost svim replikama u grupi (multicast)

\*Proces Pj ‚čita (read) vrednost promenljive x u trenutku t2 (t2 > t1).

\*Proces Pj treba da pročita vrednost 5, bez obzira kolika je razlika (t2-t1).

\*šta ako je t2-t1 = 1 nsec i koriste se optička vlakna izmedju dve host mašine (na kojima se izvršavaju ova dva procesa) koje se nalaze na rastojanju od 3 metra?.

\*Poruka kojom se zahteva ažuriranje morala bi da putuje 10 puta brže od brzine svetlosti

\*Po Ajnštajnovoj specijalnoj teoriji relativiteta to nije moguće!

-Zaključak: striktnu konzistenciju u distribuiranom sistemu je nemoguće postići!

b. -**Sekvencijalna** konzistencija je slabiji model od striktne:

-Rezultat bilo kog izvršenja je isti kao da su (read i write) operacije svih procesa na skladištu podataka izvršene u nekom sekvencijalnom redosledu i operacije svakog pojedinačnog procesa pojavljuju se u ovoj skvenci u redosledu koji je određen njihovim programom.

-Def. kaže da kada se procesi izvršavaju konkurentno na (moguće) različitim mašinama, bilo koje validno preplitanje read i write operacija je prihvatljivo, ali svi procesi vide isto preplitanje operacija.

c. -**Uslovna (kauzalna)** konzistencija- Upisi (write) koji su potencijalno uslovljeni moraju se videti u svim procesima u istom redosledu

-Konkurentni upisi se mogu videti u različitom redosledu u različitim procesima

d. **FIFO**- Upisi koje obavi jedan proces se vide od strane drugih procesa po redosledu u kome su izdati.

… ali upisi različitih procesa mogu se videti u različitom redosledu u različitim procesima.

-Drugim rečima, ne postoje nikakve garancije o tome kako različiti procesi vide upise drugih procesa, osim što dva ili više upisa iz istog izvora moraju stići po redosledu.

-Jednostavna implementacija:

Svaki proces dodaje poruci ažuriranja : (proces id, redni broj)

Svi ostali procesi primenjuju ažuriranja po redosledu u kome su ona obavljena u izvornom procesu.

1. e1 – FIFO

e2 - Uslovno, Sekvencijalno, FIFO

e2 – Uslovno, FIFO

?16.6.2017.

1. **Fajl je repliciran sa N servera. Ako se koristi Giffordov algoritam za postizanje konsenzusa , navesti koji uslovi moraju biti zadovoljeni I sta koji uslov garantuje. Ako je N=10, navesti tri dozvoljene kombinacije read I write kvoruma.**

Opštija šema (Giffordova) kaže

* 1. ako ima N replika, klijent mora da postigne read kvorum na skupu od najamanje NR ili više servera.
  2. Da bi modifikovao fajl, write kvorum od najmanje NW servera se zahteva.
  3. Moraju da važe sledeći uslovi za NR i NW:
  4. Prvi uslov sprečava read-write konflikte, a drugi write-write konflikte (tj. da postoji preklapanje između dva write skupa). Samo ako se odgovarajući broj servera složi, može se obaviti read ili write.

Nr + Nw > N

Nw > N/2

Nr = 4 , Nw = 7, Nr = 1 , Nw = 10, Nr = 3 , Nw = 9 ??????

**DFS**

\*2.7.2021.

1. **Kad smo govorili o DFS, rekli smo da server moze biti projektovan kao statefull Ili kao stateless. Pored svakog od sledecih tvrdjenja staviti oznaku T ili F:**
   1. Zakljucavanje fajla je tesko implementirati kod stateless servera.  **T**
   2. Lakse je izboriti se sa greskama kod statefull nego kod stateless servera. **F**
   3. Implementacija klijentske strane moze biti komplikovanija sa statefull serverom. ?
   4. Kod stateless servera svaki klijentski zahtev mora da sadrzi kompletnu informaciju o zahtevu (npr. Ime fajla, offset, itd).  **T**

2.7.2018.

1. **Definisati UNIX semantiku i semantiku sesije kod DFS. Da li semantika sesije poboljsava performance DFS? Obrazloziti.**

**\*Sekvencijalna semantika** (unix semantika)

* Apsolutno vremensko uređenje svih operacija.
* Svaka read operacija nad fajlom vidi efekte prethodnih write operacija.
* Sekvencijalnu semantiku u DFS je lako ostvariti ako svakim fajlom upravlja jedan server i klijenti ne keširaju fajlove.

Tada redosled read i write pristupa određuje server.

* Ako se ne koristi keširanje na strani klijenta, javlja se problem sa performansama – klijenti će se obraćati serveru za svaku operaciju na fajlu (čak i čitanje jednog bajta)
* Problem performansi se može ublažiti ako se dozvoli keširanje na strani klijenta.

-Javlja se problem konzistencije podataka: Ako klijent modifikuje podatak u kešu, a drugi klijent pročita podatke sa servera, dobiće ustajale podatke (sekvencijalna semantika više ne važi!)

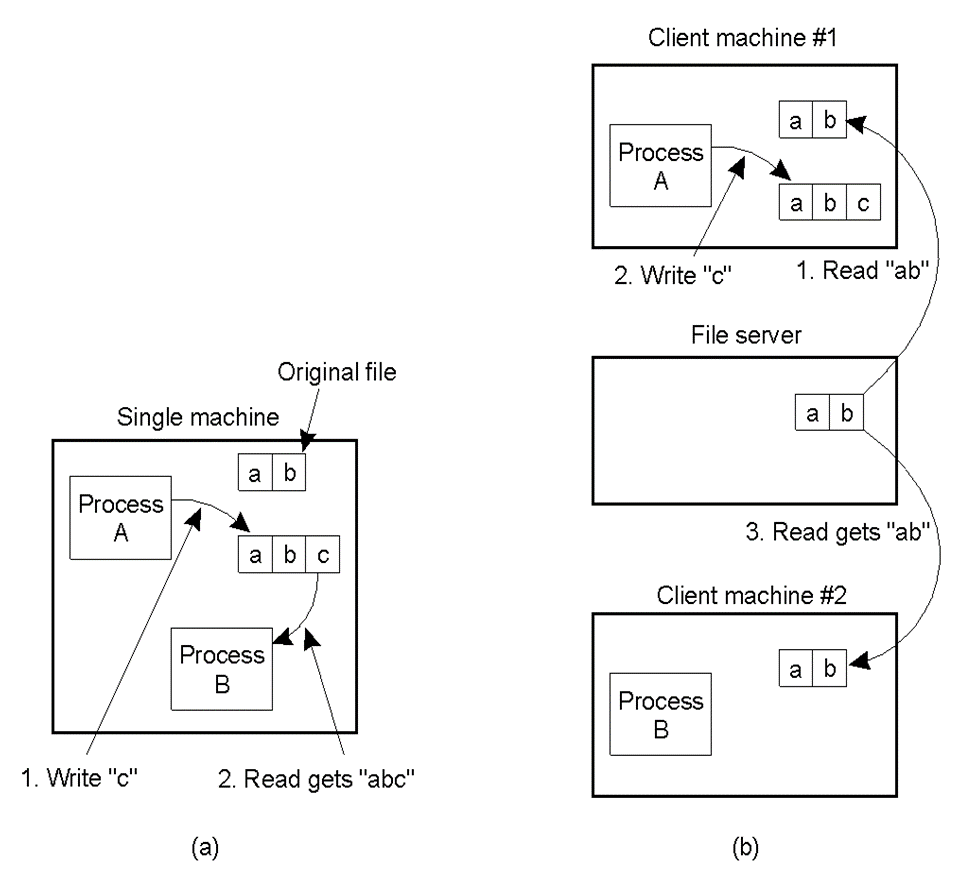
-Rešenje: koristiti write-through politiku upisa.

Ne rešava problem ustajalih kopija u keševima drugih klijenata – server mora da obavesti ostale klijente da imaju ustajale podatke.

-Drugo rešenje – ubalažiti semantiku

\***Semantika sesije**

* + Sesija – serija pristupa fajlu između otvaranja i zatvaranja fajla.
  + Promene učinjene na fajlu su vidljve samo klijent procesu (moguće i drugim procesima na istoj mašini)
  + Drugim procesima su promene vidljive nakon zatvaranja i ponovnog otvaranja fajla.



**P2p**

11.9.2017.

1. **Neki p2p sistemi koriste centralni server, a neki potpuno eliminisu koriscenje istih. Koji su prednosti, a koji nedostaci ovih arhitektura?**

Centralni:

-Kada aktivni peer dobije novi objekat, ili ukloni objekat, on informiše centrali server koji zatim ažurira svoju bazu podataka

-Da bi održao bazu podataka validnom, server mora biti u stanju da odredi kada se neki peer čvor diskonektuje.

* Prednosti:

-Peer čvor se može diskonektovati tako što će zatvoriti P2P aplikaciju ili se diskonektovati sa Interneta.

-Jedan način da server ima validne informacije je da se periodično vrši prozivka peer čvorova.

-Ako server ustanovi da peer ne odgovara na prozivku, uklanja njegovu IP adresu iz baze podataka

* Nedostaci:

-Slaba otpornost na greške (otkaz centralnog direktorijuma je fatalan za ceo sistem)

-u P2P sistemu može biti stotine hiljada korisnika, pa server mora održavati veliku bazu podtaka i odgovarati na hiljade zahteva u sec.

-produkcijske kuće su prigovarale da pomoću P2P file sharing aplikacije korisnici lako mogu doći do sadržaja koji su bili zaštićeni.

Decentralizovani:

* + - decentralizovani nestruktuirani (druga generacija)
      * svaki peer čvor sadrži tabelu sa sadržajem koji je spreman da deli sa drugima (npr. Gnutela).
      * Overlay mreža nema pravilnu strukturu.
      * Pronalaženje željenog sadržaja se ostvaruje korišćenjem ograničene bujice.
    - decentralizovani struktuirani (treća generacija)
      * Overlay mreža ima pravilnu strukturu (npr. prsten, stablo hiperkub,..)
      * informacije o podacima koji se pretražuju su rasejane po čvorovima u P2P mreži.
      * Za pronalaženje informacija koriste se tabele slične ruting tabelama, ali se pretraživanje tabela ne obavlja na osnovu IP adresa već na osnovu ključa (tvz. DHT-distributed hash table - sistemi, npr. Chord)

8.9.2020.

1. **Objasniti kako se ostvaruje pridruzivanje cvora u Gnutella mrezi.**
   1. Peer čvor X mora prvo pronaći neki drugi peer čvor koji se već nalazi u Gnutella mreži:

koristi listu potencijalnih kandidata za koje se zna da su često prisutni ili kontaktira Gnutela sajt koji sadrži takvu listu.

* 1. X sekvencijalno pokušava da uspostavi TCP konekciju sa peer čvorovima iz liste, dok sa nekim ne uspostavi konekciju, npr. Y
  2. Nakon uspostavljanja TCP konekcije X šalje Ping poruku Y;

- ping poruka sadrži IP adresu i broj porta izvora

-Y prosleđuje Ping poruku drugim peer čvorovima u overlay mreži

-Poruka sadrži i brojač koji se dekrementira pri prolasku kroz svaki peer čvor da bi se bujica držala pod kontrolom.

* 1. Svi peer čvorovi koji prime Ping poruku odgovaraju sa Pong porukom.

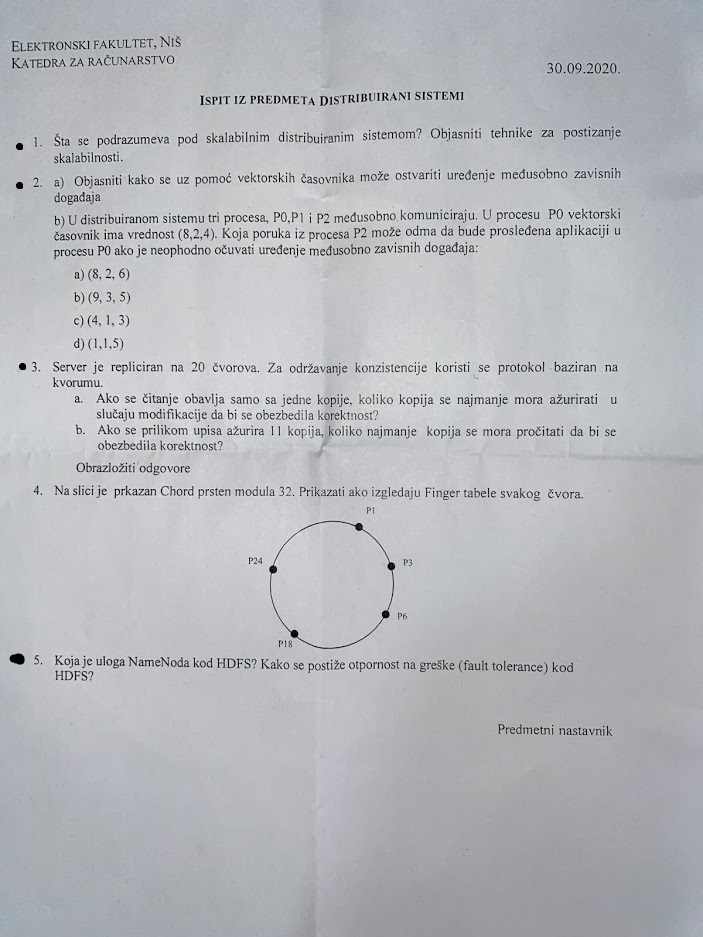
-Pong poruka sadrži IP adresu peer čvora, broj fajlova koje čvor ima na raspolaganju i ukupnu veličinu fajlova u Kbyte

* 1. X može primiti mnogo Pong poruka.

-X može nakon toga uspostaviti nove TCP konekcije i dodati nove potege u overlay mrežu.

30.9.2020., 26.6.2017

1. **Na slici je prikazan Chord prsten modula 32.** 
   1. **Prikazati kako izgledaju Finger tabele svakog cvora.**
   2. **U kom redosledu ce se posecivati cvorovi pocev od cvora P3 da bi se pronasao sadrzaj sa kljucem 27?**

** m=5,** b.P3, P24, P1

P1: P1+1 | P3 P6: P6+1 | P18

P1+2 | P3 P6+2 | P18

P1+4 | P6 P6+4 | P18

P1+8 | P18 P6+8 | P18

P1+16|P18 P6+16|P24

P3: P3+1 | P3 P18: P18+1 | P24 P24: P24+1 | P1

P3+2 | P6 P18+2 | P24 P24+2 | P1

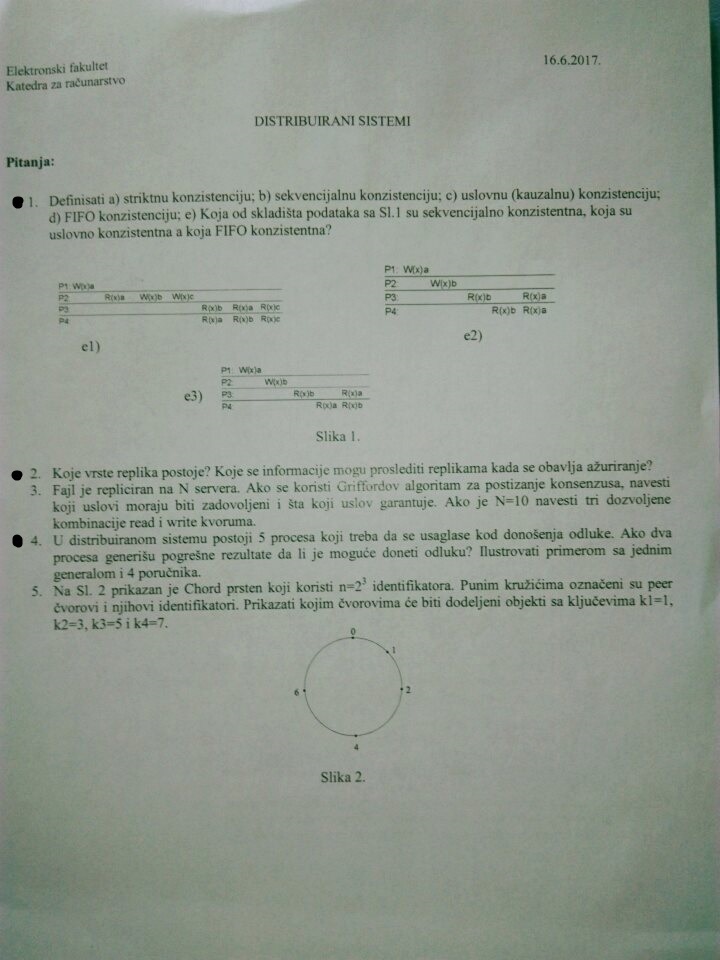
P3+4 | P18 P18+4 | P24 P24+4 | P1

P3+8 | P18 P18+8 | P1 P24+8 | P1

P3+16|P24 P18+16|P3 P24+16 | P18

16.6.2017.

1. **Na slici je prikazan Chord prsten koji koristi n=23 identifikatora. Punim kruzicima oznaceni su peer cvorovi I njihovi identifikatori. Prikazati kojim cvorovima ce biti dodeljeni objekti sa kljucevima k1=1, k2=3, k3=5 I k4=7.**

****

m=3

k1 -> 1

k2 -> 4

k3 -> 6

k4 -> 0

**HDFS**

10.9.2018.

1. **Koja je uloga NameNoda kod HDFS? Kako se postize otpornost na greske kod HDFS?**

U HDFS demone spadaju jedan NameNode (NN) demon i više DataNode (DN) demona.

Namenode deamon je demon koji se izvršava na glavnom ili master čvoru

• NameNode održava fajl sistem namespace (prostor imena) i svaka promena njegovih karakteristika je zapamćena od strane NameNode.

• NameNode čuva informacije o tome kako su fajlovi podeljeni na blokove, koji slave čvorovi čuvaju koje blokove i sam učestvuje u preslikavanju blokova u čvorove

• NameNode čuva informaciju i o vrednosti faktora replikacije

• NameNode je centralni kontroler HDFS-a

• NN čuva metapodatke fajl sistema, nadgleda ponašanje DN-a i koordiniše pristup podacima.

• NN ne čuva podatke nad kojima se vrši obrada , on samo ima informaciju o blokovima koji čine fajl i gde su ti blokovi locirani u klasteru

• Takođe vodi računa da svaki blok zadovolji definisani faktor replikacije

• Namenode je u direktnoj vezi sa ostalim čvorovima, i za komunikaciju sa DataNode demonima koristi paket određene strukture koji se naziva heartbeat

• Ovaj paket nosi takav simboličan naziv jer preko njega Namenode proverava da li je Datanode aktivan, i pored ovoga, ovaj paket prenosi identifikatore blokova podataka koji se nalaze na hostu Datanode demona.

• DataNodovi šalju heartbeatove svake tri sekunde i svaki deseti heartbeat je izveštaj o blokovima, gde DN obaveštava NN koje blokove čuva

• Ovi izveštaji služe NN da osiguraju da odgovarajući broj replika blokova (najčešče 3) postoji na različitim čvorovima, u različitim rekovima

– Greške u prisustvu velikog broja računara su neminovnost na koju se mora računati u DS i koja značajno utiče na ispravnost izvršenja aplikacije. Umesto jednog velikog i moćnog računara probati korišćenje sistema koji se sastoji iz više računara (Distribuirani sistemi).

Otpornost – Pošto je Hadoop predviđen da radi na velikom broju računara, on je dizajniran sa pretpostavkama da će često dolaziti do otkaza hardvera i veoma uspešno prevazilazi te probleme.

\*2.7.2021.

1. **Sta omogucava uspesan restart servera na kom se izvrsava NameNode demon? Koji se neophodni podaci za funkcionisanje klastera a koji poseduje NameNode I gde I kako se oni skladiste pre I nakon restarta servera?**

7.9.2021.

1. **Koje vrste HDFS demona postoje I koja je uloga svakog od njih? Opisati postupak citanja I upisa u HDFS.**

Prilikom startovanja Hadoopa na čvorovima klastera se startuje skup demona (eng. deamon) koji ima specifične uloge.

• Postoje dve vrste demona Hadoop-a:

– HDFS demoni

– MapReduce demoni

• U HDFS demone spadaju jedan NameNode (NN) demon i više DataNode (DN) demona

• Postoji samo jedan NameNode i više DataNode-ova u klasteru

• DataNode odgovoran za čuvanje podataka u blokovima, primanje naredbi od NN i davanje informacija NN-u.

(NN opisan iznad)

DataNodovi šalju heartbeatove svake tri sekunde i svaki deseti heartbeat je izveštaj o blokovima, gde DN obaveštava NN koje blokove čuva

• Ovi izveštaji služe NN da osiguraju da odgovarajući broj replika blokova (najčešče 3) postoji na različitim čvorovima, u različitim rekovima

**HDFS upis:** Klijent je spreman da loaduje fajl File.txt u klaster (razbija ga u blokove A,B i C) počevši od bloka A.

• Klijent kontaktira Name Node, konsultuje ga, dobija dozvolu od NN i dobija listu od (3) DataNoda za svaki blok koji treba da bude upisan (jedinsvenu listu za svaki blok).

• NN koristi svoje Rack Awereness podatke da bi uticao na odluku koji DataNodovi će se naći na listi za pojedinačni blok • Ključno pravilo je da za svaki blok podataka, jedna kopija se nalazi u jednom reku, a druge dve kopije u reku različitom od ovog s tim što su te dve kopije u istom reku.

• Sve liste koje dobija Klijent moraju da poštuju ovo pravilo

• Za blok A na listi su DN1, DN5 i DN6.

• Pre nego što Klijent upiše Blok A fajla File.txt u klaster, on želi da zna da su svi DataNodovi u koje treba da upiše spremni da prime blok.

• Nakon stizanja potvrde, od DN6 do DN5, od DN5 do DN1, i od DN1 do klijenta, Klijent je spreman da upiše blok A.

• Prilikom upisa DN-ovi formiraju pipeline, u redosledu koji minimizira rastojanje od klijenta do poslednjeg DN-a

**Čitanje podataka iz HDFS-a:** • Kad Klijent želi da pročita rezultujući fajl iz HDFS, konsultuje NN i pita za lokacije blokova u rezultujućem fajlu

• NN za svaki blok vraća listu DN-ova koji ga sadrže

• Klijent bira za svaki blok prvi DN iz liste (lokacije blokova su sortirane prema udaljenosti od klijenta) tj. najbližu repliku • Klijent čita blokove sa odgovarajućih DN-a sekvencijalno, jedan po jedan