



**Diplomski studij**

**Informacijska i  
komunikacijska tehnologija  
Telekomunikacije i  
informatika**

**Računarstvo  
Računarska znanost  
Programsko inženjerstvo i  
informacijski sustavi**

# **Raspodijeljeni sustavi**

Pitanja za provjeru znanja s odgovorima  
1. blok predavanja

**Ak.g. 2011./2012.**

***Napomena:***

*Preporučena literatura su bilješke s predavanja.*

**Zadatak** Objasnite pojam skalabilnosti raspodijeljenog sustava.

**1.1**

Raspodijeljeni sustav je skalabilan ukoliko posjeduje sposobnost prilagodbe povećanom broju korisnika i sredstava, njihovu rasprostranjenosti te načinu upravljanja sustavom.

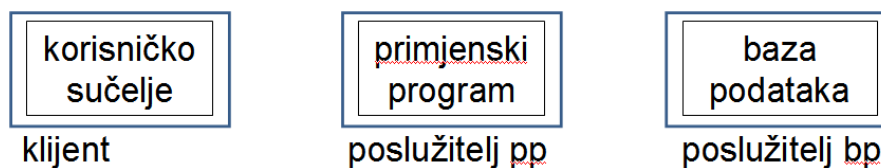
**Zadatak** Objasnite pojam migracijske transparentnosti raspodijeljenog sustava.

**1.2**

Za raspodijeljeni sustav kažemo da posjeduje migracijsku transparentnost ukoliko on prikriva promjenu lokacije nekog sredstva na način da ta promjena ne utječe na način pristupa tom sredstvu.

**Zadatak** Skicirajte trorednu arhitekturu klijent-poslužitelj te na proizvoljnom primjeru aplikacije objasnite ulogu svake razine u cjelokupnoj arhitekturi.

**1.3**



Primjer su aplikacije weba, gdje klijentski program koji se izvodi na klijentskom računalu nikada ne pristupa direktno bazi podataka, već posredno preko aplikacije weba. Klijentski program prikazuje korisničko sučelje i komunicira s aplikacijom weba koja obavlja cjelokupnu logiku usluge i pristupa potrebnim podacima.

**Zadatak** Objasnite razliku između sinkrone i asinkrone komunikacije.

**1.4**

Dok je kod sinkrone komunikacije pošiljalac blokiran nakon slanja poruke sve do primitka potvrde o isporuci, kod asinkrone komunikacije pošiljalac nije blokiran te nastavlja procesiranje odmah nakon slanja.

**Zadatak** Navedite obilježja komunikacije *socketom* UDP.

**2.1**

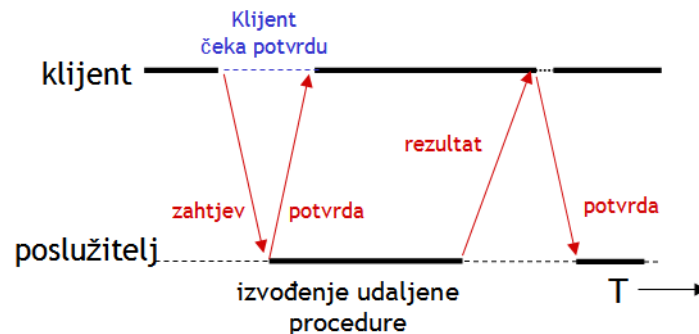
Ova komunikacija se temelji na modelu klijent-poslužitelj, gdje oba moraju istovremeno biti aktivna da bi se komunikacija mogla ostvariti. Komunikacija je tranzijentna i asinkrona, a može se koristiti za implementaciju komunikacije na načelu *pull* ili *push*.

**Zadatak** U tablicama su prikazane metode na klijentskoj i poslužiteljskoj strani *socket* TCP. Upišite ispravan redoslijed izvođenja metoda u tablice.

**2.2**

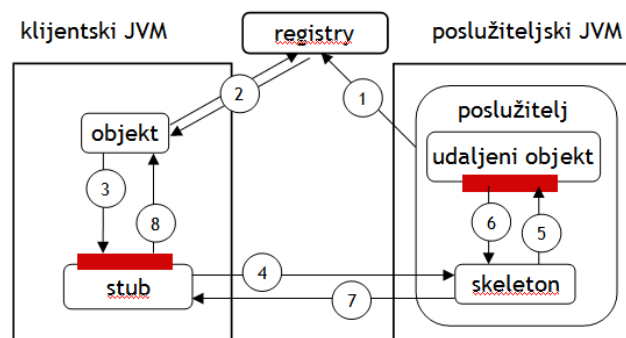
Klijent		Poslužitelj	
1	socket()	3	listen()
3	write()	1	socket()
4	read()	4	accept()
5	close()	6	write()
2	connect()	5	read()
		7	close()
		2	bind()

- Zadatak 2.3** Skicirajte tijek komunikacije između klijenta i poslužitelja te objasnite odgođeni sinkroni poziv udaljene procedure RPC (*Remote Procedure Call*).



Kod odgođenog sinkronog poziva udaljene procedure, klijent nije blokiran dok čeka rezultat izvođenja, već nastavlja s radom nakon uspješnog primitka potvrde. Kasnije mu poslužitelj šalje rezultat koristeći drugi asinkroni poziv udaljene procedure.

- Zadatak 2.4** Skicirajte model pozivanja udaljene metode Java RMI (*Remote Method Invocation*). Navedite korake u komunikaciji potrebne da bi klijent pozvao metodu dostupnu na poslužitelju, uz pretpostavku da je klasa stub već instalirana na klijentskoj strani



Koraci u komunikaciji su sljedeći:

1. Poslužitelj registrira udaljeni objekt pod odabranim imenom.
2. Klijent od *registry*a traži referencu na udaljeni objekt koristeći registrirano ime.
3. Klijent poziva metodu *stuba* dostupnu na klijentskom računalu.
4. *Stub* serijalizira parametre i šalje ih *skeletonu*.
5. *Skeleton* deserijalizira parametre i poziva metodu udaljenog objekta.
6. Udaljeni objekt vraća rezultat izvođenja metode *skeletonu*.
7. *Skeleton* serijalizira rezultat i šalje ga *stubu*.
8. *Stub* deserijalizira rezultat i dostavlja ga klijentu.

- Zadatak 3.1** Skicirajte i objasnite primjer komunikacije porukama između dva procesa/objekta (primatelja i pošiljatelja). Kakva je komunikacija porukama s obzirom na vremensku ovisnost primatelja i pošiljatelja?



U komunikaciji između pošiljatelja i primatelja rep sudjeluje kao posrednik. Pošiljatelju se u načelu

garantira isporuka poruke u primateljev rep, ali ne i isporuka poruke primatelju. Primatelj može pročitati poruku iz repa u bilo kojem budućem trenutku. Stoga su pošiljalatelj i primatelj poruke vremenski neovisni.

**Zadatak 3.2** Objasnite sličnost i razlike u obilježjima komunikacije između dva komunikacijska modela podržana s JMS (*Java Messaging Service*)?

JMS podržava komunikaciju porukama i model objavi-pretplati. Obje vrste komunikacije su vremenski neovisne zato što pošiljalatelj i primatelj ne moraju istovremeno biti dostupni. Kod komunikacije porukama pošiljalatelj mora znati identifikator odredišta, dok je kod modela objavi-pretplati komunikacija anonimna. Komunikacija je perzistentna i asinkrona u oba slučaja. Komunikacija se pokreće na načelu *pull* kod komunikacije porukama, a na načelu *push* kod modela objavi-pretplati.

**Zadatak 3.3** Navedite i objasnite operacije koje implementira programska infrastruktura dijeljenog podatkovnog prostora.

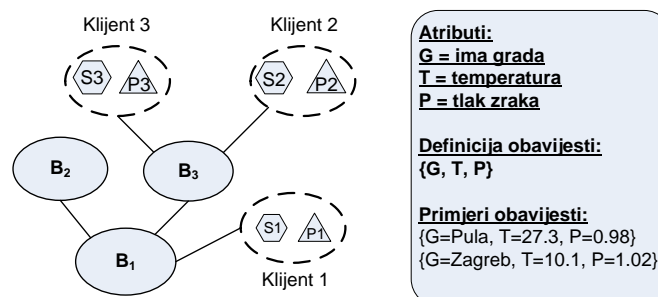
write(t) – dodaj tuple t u raspodijeljeni podatkovni prostor  
 read(s) -> t – vraća tuple t koji odgovara predlošku s  
 take(s) -> t – vraća tuple t koji odgovara predlošku s i briše ga iz podatkovnog prostora

**Zadatak 3.3** Objasnite opći format poruka protokola HTTP. Navedite kako glasi potpun i apsolutan URI koji identificira resurs zatražen u zahtjevu, ako prva 2 retka HTTP zahtjeva sadrže sljedeće podatke:

GET /predmet/rassus HTTP/1.1  
 Host: [www.fer.hr](http://www.fer.hr)

Opći format poruka protokola HTTP sastoji se od početnog retka, polja zaglavlja te tijela poruke. Potpun i apsolutan URI je <http://www.fer.hr/predmet/rassus>.

**Zadatak 3.4** Raspodijeljeni sustav objavi-pretplati, u kojem se koristi **algoritam preplavlivanja obavijestima**, sastoji se od 3 posrednika i 3 klijenta kako je prikazano slikom. Svaki klijent u sustavu ima ulogu pretplatnika i objavljiivača. Odgovorite na sljedeća pitanja:



- U trenutku  $t_1$  **klijent 1** generira pretplatu  $s_1=\{G=Zagreb, T<15.5, P>0.98\}$ . Napišite oznake svih posrednika na kojima se pohranjuje ova pretplata. Pretplata se pohranjuje na posredniku B1.
- U trenutku  $t_2>t_1$  **klijent 2** generira pretplatu  $s_2=s_1$ . Napišite oznake svih posrednika na kojima se pohranjuje ova pretplata. Pretplata se pohranjuje na posredniku B3.
- U trenutku  $t_3>t_2$  **klijent 3** generira obavijest  $p_1=\{G=Zagreb, T=-2.2, P=1.01\}$ . Objasnite točan redoslijed kojim će se ova obavijest proširiti sustavom i biti isporučena zainteresiranim klijentima.  
 $P_3 \rightarrow B_3 \rightarrow S_2 \rightarrow B_1 \rightarrow S_1 \rightarrow B_2$

**Zadatak 4.1** Korisnik nakon ispunjavanja obrasca na Web-u odabire opciju Submit, čime pošalje podatke Web-poslužitelju na adresu `www.tel.fer.hr/obrazac/accept` korištenjem protokola HTTP verzije 1.1. Kojim se HTTP zahtjevom šalju podaci poslužitelju i kako je definiran prvi redak zahtjeva?

Podaci se šalju zahtjevom POST. Prvi redak je definiran na sljedeći način:  
POST /obrazac/accept HTTP 1.1 .

**Zadatak 4.2** Objasnite opći format poruka protokola HTTP. Navedite kako glasi potpun i apsolutan URI koji identificira resurs zatražen u zahtjevu, ako prva 2 retka HTTP zahtjeva sadrže sljedeće podatke:

GET /predmet/rassus HTTP/1.1  
Host: www.fer.hr

Opći format poruka protokola HTTP sastoji se od početnog retka, polja zaglavlja te tijela poruke. Potpun i apsolutan URI je `http://www.fer.hr/predmet/rassus`.

**Zadatak 4.3** Pretpostavite da se sjedište weba sastoji od 2 poslužitelja priključena na Internet preko posrednika (*proxy*). Navedite i objasnite svojstva ovog raspodijeljenog sustava.

Ovaj raspodijeljeni sustav karakteriziraju:

- 1) replikacijska transparentnost – zato što korisnik nije svjestan koji poslužitelj ga je zapravo poslužio,
- 2) otpornost na kvarove – jer se kvar jednog poslužitelja može prikriti od korisnika i
- 3) skalabilnost – zato što ovaj sustav posjeduje sposobnost prilagodbe povećanom broju korisnika.

Ovaj sustav podržava i lokacijsku i migracijsku transparentnost (putem sustava DNS).

**Zadatak 4.4** Objasnite razliku između web-aplikacija temeljenih na CGI (Common Gateway Interface) i poslužiteljskim skriptama.

CGI (*Common Gateway Interface*) je jednostavno sučelje za pokretanje eksternih programa iz web-poslužitelja na platformski i programski neovisan način. Kod svakog zahtjeva se pokreće novi proces, a podaci između poslužitelja i procesa šalju se preko varijabli okoline i tokova podataka. Nakon svake obrade proces se gasi. Nedostatak CGI-a je što se kod svakog zahtjeva pokreće novi proces i nakon obrade gasi što je zahtjevno za resurse (procesorsko vrijeme i memorija) pa kod velikog broja zahtjeva na poslužitelju to znatno utječe na performanse.

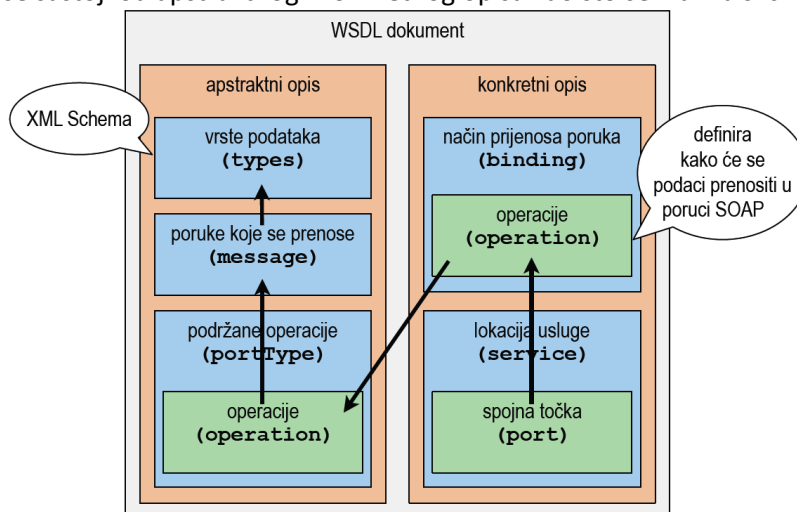
Poslužiteljske skripte (engl. *server side scripts*), dinamički generiraju HTML-dokumente poput CGI-ja, ali je razlika u tome što se za svaki zahtjev ne pokreće novi proces i na taj način se štede resursi.

**Zadatak 5.1** Navedite dva osnovna načina rada protokola SOAP i objasnite kako se poruka SOAP šalje pomoću protokola HTTP.

Dva osnovna načina rada koje podržava protokol SOAP su poziv udaljene procedure te razmjena dokumenata i poruka. Poruka SOAP, koja je pisana jezikom XML, se sastoji od zaglavlja i tijela. Prilikom slanja poruke SOAP protokolom HTTP, i zaglavlje i tijelo poruke SOAP se nalaze u tijelu poruke HTTP.

**Zadatak 5.2** Objasnite i skicirajte sadržaj apstraktnog i konkretnog opisa u strukturi dokumenta WSDL.

Dokument WSDL se sastoji od apstraktnog i konkretnog opisa kao što se vidi na slici.



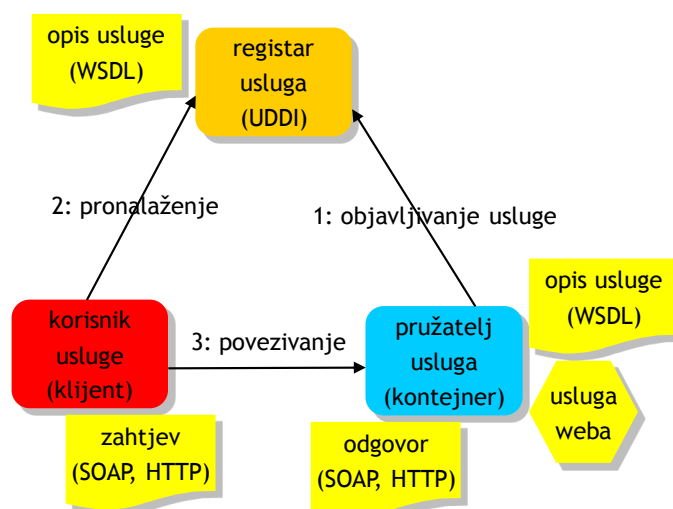
Apstraktan opis u WSDL-u se sastoji od 4 elementa:

1. *types*: definira vrste podataka neovisne o platformi i jeziku (koristi se XML Schema),
2. *message*: definiraju ulazne i izlazne poruke koje se mogu koristiti kao parametri usluge,
3. *operation*: predstavlja jednu operaciju/metodu/proceduru koja je definirana u usluzi, a sastoji se od definicija ulaznih, izlaznih i iznimnih poruka koje se mogu razmjenjivati korištenjem ove operacije i
4. *portType*: koristi poruke (pod 2) da bi opisao sve operacije koje pruža usluga.

Konkretni opis se sastoji od 2 dijela:

1. *binding*: definira kako je konkretna implementacija povezana s operacijama u apstraktnom opisu i definira format u kojem će se poruke prenositi (protokol i elemente) i
2. *service*: definira URI gdje je usluga isporučena tj. na kojoj adresi se može pozvati usluga (taj URI je definiran u spojnoj točki).

**Zadatak 5.3** Prikažite arhitekturu i objasnite korištenje usluge Web.



U arhitekturi usluge Weba postoje 3 uloge: pružatelj usluge, registar usluga i korisnik usluge. Pružatelj usluge je vlasnik usluge Weba i zadužen je za njen smještaj. Klijent je stranka zainteresirana za uslugu weba, a registar usluga omogućava pretraživanje registriranih usluga weba po njihovim opisima.

Tipično korištenje usluge Weba sastoji se od tri operacije: objavljivanje, pronalaženje i povezivanje. Pružatelj usluge registrira uslugu i njen opis (WSDL) objavljivanjem usluge u registru usluga. Korisnik usluge putem registra usluga pronalazi traženu uslugu Weba. U zadnjem koraku korisnik usluge poziva metodu usluge Weba i dobiva rezultat njenog izvođenja putem poruka SOAP i protokola HTTP.

**Zadatak** Objasnite svojstvo slabe povezanosti usluga kod uslužno orijentirane arhitekture.

#### 5.4

Svojstvo slabe povezanosti usluga kod uslužno orijentirane arhitekture (SOA – Service Oriented Architecture) odnosi se na dizajn programske izvedbe usluga. U sustavu SOA, usluge trebaju biti izvedene tako da promjena u jednoj usluzi ne zahtijeva promjenu neke druge usluge. Pri tome, svaka usluga može i dalje nesmetano koristiti neku drugu uslugu. Npr. algoritam koji se koristi u usluzi se može promijeniti bez znanja drugih usluga i druge usluge ju mogu nesmetano koristiti. Bitno je da se sučelja opisana WSDL-om ne promijene.

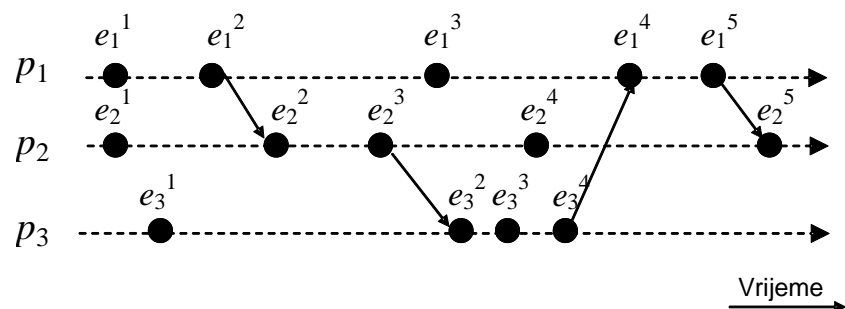
**Zadatak** Objasnite za koje je od sljedeća tri svojstva raspodijeljenih sustava značajna komunikacijska složenost algoritama: a) replikacijska transparentnost b) skalabilnost c) otvorenost.

#### 6.1

Komunikacijska složenost algoritama je važna za skalabilnost raspodijeljenog sustava jer na temelju komunikacijske složenosti možemo zaključiti kako raste generirani promet raspodijeljenog sustava s rastom tog sustava. Primjer: komunikacija grupe procesa.

**Zadatak** Na temelju primjera procesa sa slike **objasnite** jesu li sljedeći parovi događaja uzročno povezani ili nisu? a)  $e_1^3$  i  $e_2^2$  i b)  $e_2^2$  i  $e_1^5$ .

#### 6.2



- Događaji  $e_1^3$  i  $e_2^2$  su neovisni zato što nisu slijedni događaji na istom procesu, između njih ne postoji slanje i primanje poruke te između njih ne postoji tranzitivna uzročnost.
- Između događaja  $e_2^2$  i  $e_1^5$  postoji uzročna povezanost preko tranzitivne uzročnosti  $e_2^2 \rightarrow e_2^3 \wedge e_2^3 \rightarrow e_3^2 \wedge e_3^2 \rightarrow e_3^3 \wedge e_3^3 \rightarrow e_3^4 \wedge e_3^4 \rightarrow e_1^4 \wedge e_1^4 \rightarrow e_1^5$ .

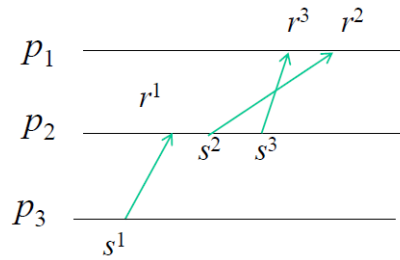
**Zadatak** Objasnite model komunikacijskog kanala koji se temelji na uzročnoj slijednosti.

#### 6.3

Uzročna slijednost (causal ordering) osigurava da uzročno povezani događaji slanja dviju poruka istom primatelju rezultiraju primanjem u slijedu kojim su poslani.

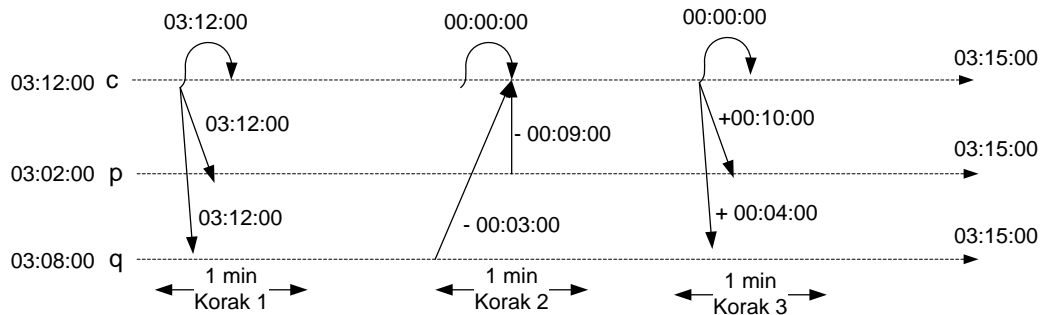
**Zadatak** Objasnite zašto za sljedeći primjer vrijedi CO ili vrijedi non-CO?

**6.4**



Za primjer vrijedi non-CO zato što proces  $p_1$  prima poruke od  $p_2$  drugačijim redoslijedom od redoslijeda slanja, a pri tome je slanje poruke 3 uvjetovano slanjem poruke 2.

**Zadatak** Prikažite i objasnite korake algoritma Berkeley za usklađivanje satnih mehanizama tri računala u raspodijeljenoj okolini. Računala imaju sljedeće vrijednosti satova  $T(p)=03:02:00$ ,  $T(q)=03:08:00$  i  $T(c)=03:12:00$ . Upravitelj je treće računalo. Pretpostavite da prijenos poruke između 2 računala traje 1 minutu i da upravitelj koristi svoje lokalno vrijeme kao zajedničko pri usklađivanju satnih mehanizama.

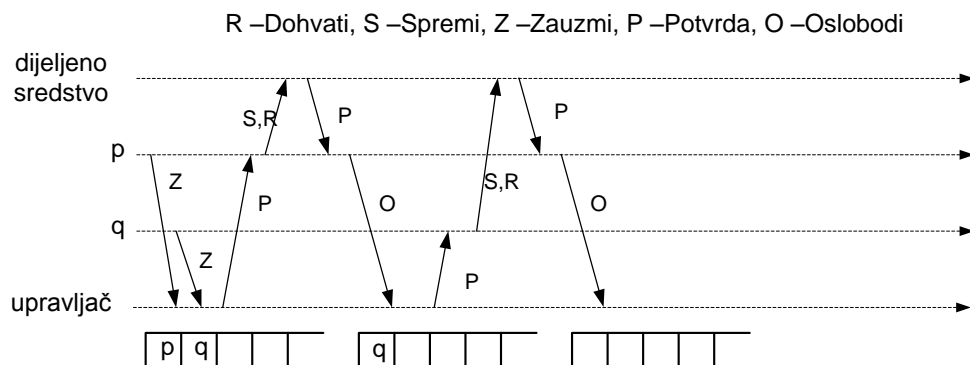


**Korak 1:**  
Upravitelj šalje poruku s trenutnim vremenom svim računalima.

**Korak 2:**  
Poslane poruke putuju 1 minutu i nakon primitka poruka, računala odgovaraju s porukom koja sadrži razliku lokalnog vremena u odnosu na primljeno vrijeme.

**Korak 3:**  
Nakon primitka poruka odgovora, upravitelj šalje poruke zahtjeva koje sadrže vremenski pomak za svako računalo. Poruke zahtjeva putuju 1 minutu te nakon primitka poruke zahtjeva, svako računalo usklađuje lokalni satni mehanizam.

**Zadatak** Opišite postupak međusobnog isključivanja dvaju procesa (p i q) primjenom središnjeg upravljača s repom čekanja tako da nacrtate redoslijed operacija i objasnite ih. Nakon zauzimanja dijeljenog spremnika, proces provodi jednu operaciju čitanja ili pisanja nad dijeljenim spremnikom.





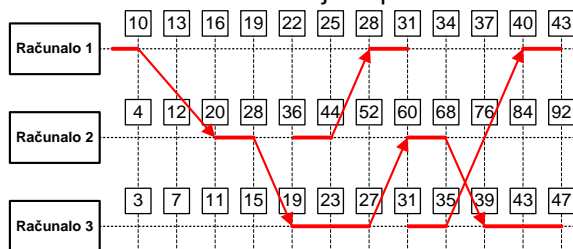
Proces p šalje zahtjev za zauzimanje sredstva,  
zahtjev se sprema u rep  
Proces q šalje zahtjev za zauzimanje sredstva,  
zahtjev se stavlja u rep  
Kako je zahtjev od R0 stigao prije, upravljač šalje  
potvrdu R0 i uklanja njegov zahtjev iz repa

Proces p provodi operaciju pisanja  
Proces p prima potvrdu  
Proces p šalje poruku Upravljaču i otpušta pristup  
Upravljač šalje poruku dojave procesu q te mu  
dodjeljuje pristup dijeljenom spremniku. Iz  
repa zahtjeva uklanja se zahtjev od procesa p  
Proces q provodi operaciju pisanja  
Proces q prima potvrdu  
Proces q šalje poruku Upravljaču i otpušta pristup

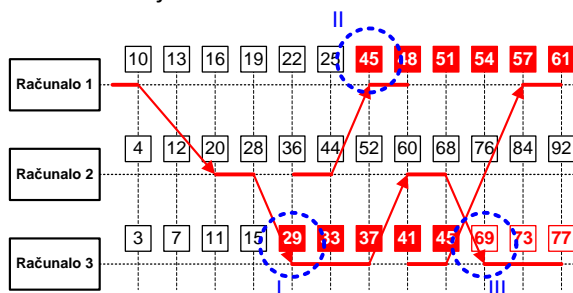
### Zadatak 7.3

Za slijed razmjene poruka između tri računala prikazan na slici uspostavite globalni  
tijek vremena primjenom skalarnih oznaka logičkog vremena. Navedite i opišite  
trenutke u kojima se ostvaruje korekcija lokalnih satnih mehanizama.

1. Određivanje vrijednosti satnih mehanizama  
u trenutcima razmjene poruka



2. Primjena skalarnih oznaka vremena



**Trenutak I:** Računalo 3 prima poruku od računala 2 s oznakom vremena  $T_p=28$  koja je veća od lokalne oznake vremena  $T_L=19$ . Lokalni sat se pomiče na vrijednost  $T_p+1=29$ .

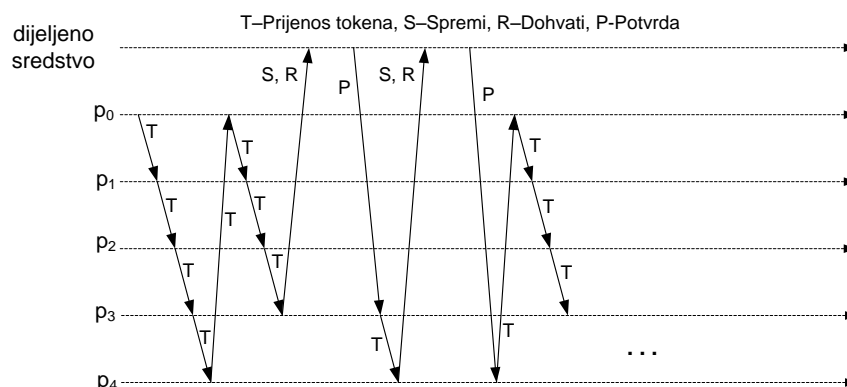
**Trenutak II:** Računalo 1 prima poruku od računala 2 s oznakom vremena  $T_p=44$  koja je veća od lokalne oznake vremena  $T_L=28$ . Lokalni sat se pomiče na vrijednost  $T_p+1=45$ .

**Trenutak III:** Računalo 3 prima poruku od računala 2 s oznakom vremena  $T_p=68$  koja je veća od lokalne oznake vremena  $T_L=49$ . Lokalni sat se pomiče na vrijednost  $T_p+1=69$ .

$$\max(28+1, 19) = 29$$

### Zadatak 7.4

Pet procesa postavljenih na različita računala u raspodijeljenoj okolini ostvaruje međusobno isključivanje primjenom prstena. Vrijeme prijenosa poruke zahtjeva i odgovora pri pristupu dijeljenom sredstvu jednako je 3 ms, vrijeme obrade poruke zahtjeva na sredstvu je 5 ms, vrijeme prijenosa *tokena* između dva susjedna procesa u prstenu je 2 ms. Kada primi *token*, proces može maksimalno jednom ostvariti pristup dijeljenom sredstvu prije nego što proslijedi *token* idućem susjedu. Prikažite naznačite navedena vremena na dijagramu. Koje je minimalno, a koje maksimalno vrijeme čekanja bilo kojeg procesa u prstenu za pristup dijeljenom sredstvu.



**Min. vrijeme** - U najboljem slučaju, proces koji želi ostvariti pristup čeka  $T=0$  sekundi. Naime, taj slučaj nastupa kada proces uđe u stanje u kojem želi ostvariti pristup sredstvu netom prije nego što je primio token. **Max. vrijeme** - U najgorem slučaju, proces ulazi u stanje u kojem želi ostvariti pristup sredstvu netom nakon što je proslijedio token svojem susjedu. U tom slučaju, proces mora čekati da svi ostali procesi prime token i ostvare pristup dijeljenom sredstvu. Maksimalno vrijeme čekanja u tom slučaju iznosi  $T = 5 * T_T + 4 * (T_Z + T_O + T_P) = 10 + 44 = 54$  ms.