



**Diplomski studij**

**Informacijska i  
komunikacijska  
tehnologija**

Telekomunikacije i  
informatika

**Računarstvo**

Programsko inženjerstvo i  
informacijski sustavi

Računarska znanost

## **Raspodijeljeni sustavi**

Pitanja za provjeru znanja  
**1. blok predavanja**

**Ak. g. 2020./2021.**

**Napomena:** Preporučena literatura su bilješke s predavanja.

<b>Zadatak 1.1</b>	Objasnite pojam skalabilnosti raspodijeljenog sustava.
<b>Zadatak 1.2</b>	Objasnite pojam migracijske transparentnosti raspodijeljenog sustava.
<b>Zadatak 1.3</b>	Definirajte Internet stvari.
<b>Zadatak 1.4</b>	Opišite okruženje Interneta stvari. Na jednom primjeru usluge navedite ulogu svakog dionika u ostvarenju usluge.

1.1 Skalabilnost omogućava povećavanje sustava prilikom pristizanja više zahtijeva, to se može raditi automatski ili ručno, replikama imamo problem održavanja konzistentnosti između kopije i originala, višetrakostcu - raspodijeljena baza podataka.

- koliko ih skaliramo
- na kojem prostoru (nije isto lokalno ili mrežno)
- kako komuniciraju

1.2 prikrivanje promjene lokacija, ako promijenimo lokaciju to ne omogućava pristup sredstvu niti mijenja taj način

1.3 povezivanje uređaja na internet, fizičkih i virtualnim

1.4 senzor može opaziti okolinu, aktuator izvršiti određene funkcije, oni se spajaju na internet i imaju svoj id. Njima upravlja ili šalje podatke svome korisniku, primjer je mikrokontroler esp 8266 koji se može spojiti na internet i preko kojeg se mogu dobivati podaci te slati naredbe

**Zadatak 2.1** Objasnite razliku između sinkrone i asinkrone komunikacije.

**Zadatak 2.2** Navedite obilježja komunikacije *socketom* UDP.

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

**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.

2.1

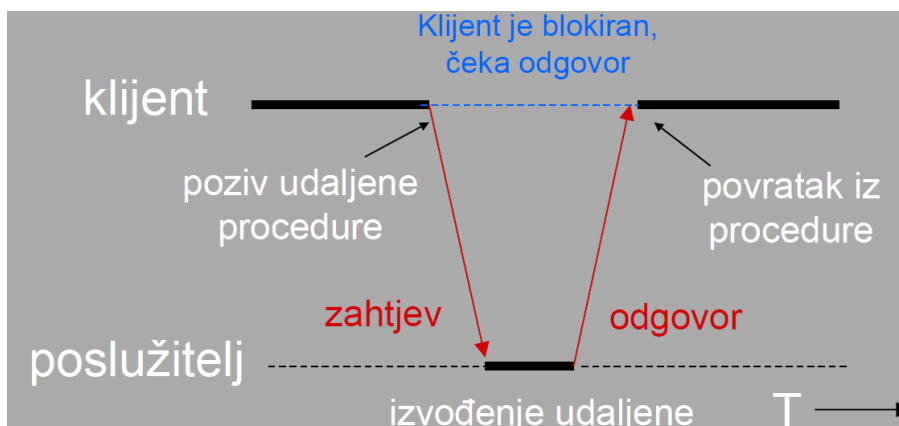
Sinkrona komunikacija - posiljalatelj je blokiran dok primatelj ne odgovori,  
asinkrona - nakon slanja poruke može ju ponoviti jer nije blokiran

2.2

Za razliku od TCP-a, nema provjere paketa, služi za prijenos videa pa gubitci se mogu tolerirati do neke granice  
nespojini  
prenosi datagrame  
asinkrona komunikacija  
tranzijetna stanja

2.3

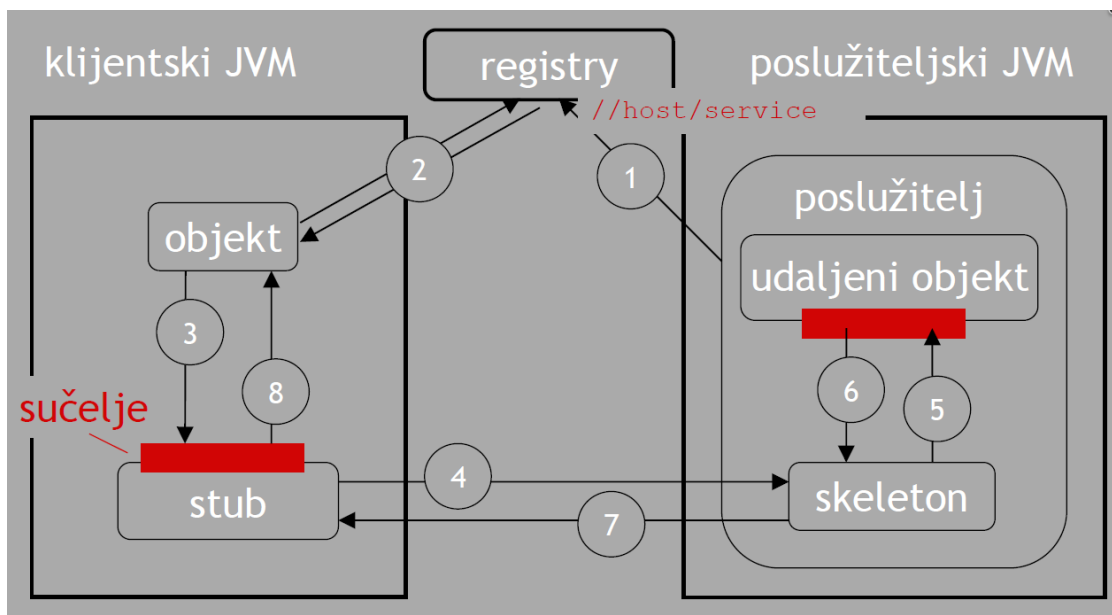
- omogućuje procesima pozivanje i izvođenje procedure na udaljenom računalu



2.4

sucelje udaljenog implementira stup u adresnom prostoru klijentskog računala

klase stub i skeleton generiraju se iz implementacije a ne udaljenog objekta



- Zadatak 3.1** 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?
- Zadatak 3.2** 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
- Zadatak 3.3** Objasnite razliku između web-aplikacija temeljenih na CGI (Common Gateway Interface) i poslužiteljskim skriptama.
- Zadatak 3.4** Navedite dva osnovna načina rada protokola SOAP i objasnite kako se poruka SOAP šalje pomoću protokola HTTP.
- Zadatak 3.5** Objasnite sadržaj apstraktnog i konkretnog opisa u strukturi dokumenta WSDL.
- Zadatak 3.6** Objasnite svojstvo slabe povezanosti usluga kod uslužno orijentirane arhitekture.

3.1 POST *www.tel.fer ... HTTP/1.1*

3.2 apsolutni uri --> *www.fer.hr/predmet/rassus*

Format

Request GET /path HTTP/1.0 --> GET - metoda, /path - putanja, HTTP/x.x - verzija

Response HTTP/1.0 200 OK , HTTP/1.0 -verzija, 200 OK - status kod

jos sadrže header i body

3.3 CGI kod svakog zahtjeva pokreće proces, podaci se razmjenjuju preko varijabli i tokova (Bash i Perl)  
Server script - generiranje htmla iz skripte (PHP, ASP, Ruby on Rails)

3.4 remote procedure call (RPC) i razmjena dokumenata/poruka

3.5 Apstraktan -

types (vrste podataka neovisne o platformi i jeziku)

message (ulazne i izlazne poruke kao parametri),

operation (operacija na uzluzi, sastoji se od ulaznih, izlaznih i iznimnih poruka)

portType (koristi poruke za opisivanje operacija)

konkretni -

binding (konkretna implementacija povezana s operacijama u apstraktnom opisu)

service (uri na kojem je usluga)

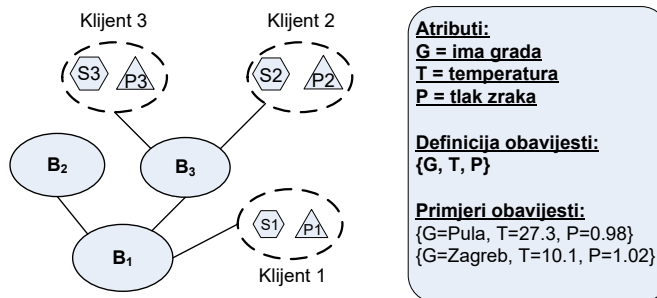
3.6 jedna usluga ne ovisi o tehnologiji implementacije druge

**Zadatak 4.1** 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?

**Zadatak 4.2** Objasnite sličnosti i razlike u obilježjima komunikacije između dva komunikacijska modela podržana s JMS (*Java Messaging Service*)?

**Zadatak 4.3** Navedite i objasnite operacije koje implementira programska infrastruktura dijeljenog podatkovnog prostora.

**Zadatak 4.4** 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:



4.3 write(A) - dodaj tuple A (slijed podataka)

read(a) -> A vraća tuple koji odgovara predlošku a

take(b) -> B vraća tuple koji odgovara predlošku b i briše ga

a) 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.

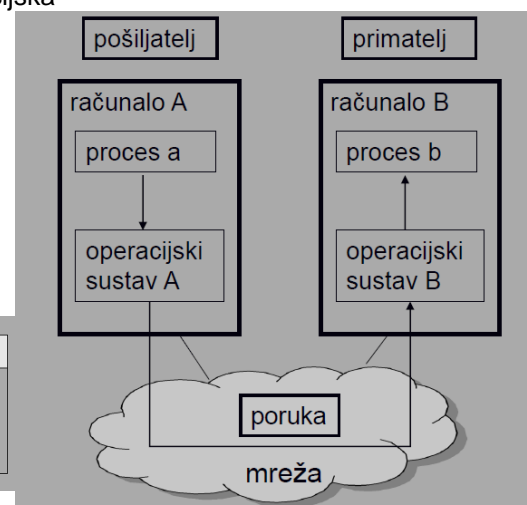
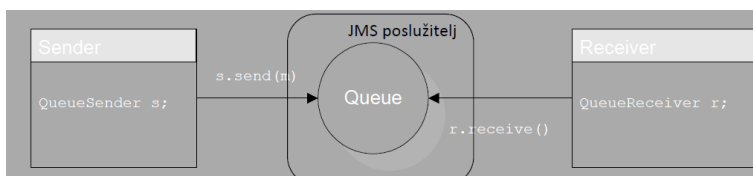
b) U trenutku  $t_2 > t_1$  **klijent 2** generira pretplatu  $s_2 = s_1$ . Napišite oznake svih posrednika na kojima se pohranjuje ova pretplata.

c) 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.

4.1 akcija je ili sinkrona ili asinkrona, pull push, konekcijska bezkonekcijska, perzistentna, tranzijentna

4.2

Point-to-Point - komunikacija porukama, jedna poruka za jedno odrediste  
klijent s šalje prouku m sa `s.send(m)`, sprema se u rep, ako postoji poruka briše se iz repa i šalje sa `r.receive()`



Publish/subscribe - objavi- pretplati - poruka za skup pretplatnika  
izvor objavljuje prouku sa `p.publish(m)`

isporučuje se poruka preko TopicSubscribera s u MessageListerner l sa naredbom `l.onMessage(m)`



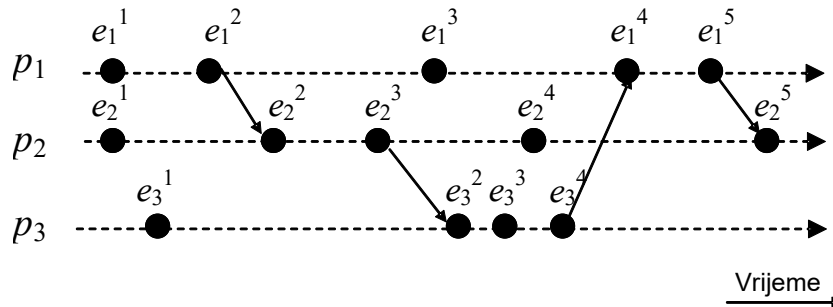
**Zadatak 5.1**

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

**Zadatak 5.2**

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  $e_1^5$ .

autonomni proces pi  
događaj  $e_i^{1,2,3 \dots}$

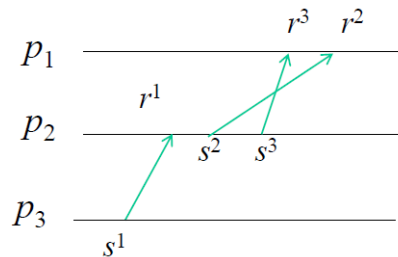


**Zadatak 5.3**

Objasnite model komunikacijskog kanala koji se temelji na uzročnoj slijednosti.

**Zadatak 5.4**

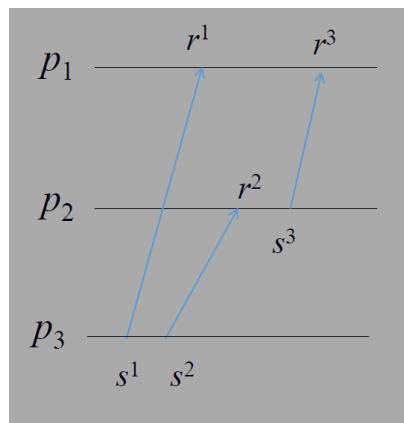
Objasnite zašto za sljedeći primjer vrijedi CO ili vrijedi non-CO?



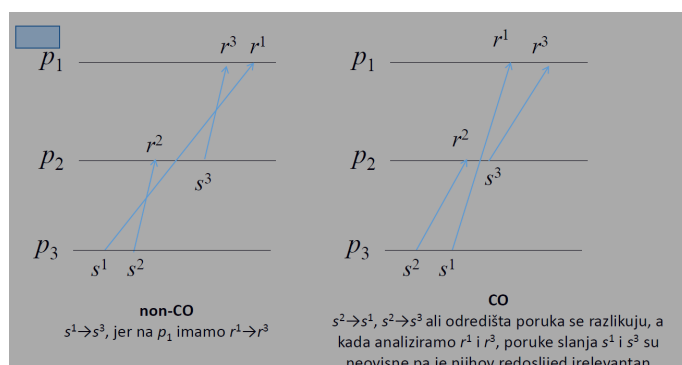
5.1 C je valjda, replikacijska transparentnost znaci da ne znas kojoj replici pristupas, skalabilnost povecanje resursa, jedino otvorenost izgleda kao mjesto koje treba

5.2 ?

5.3 osigurava da su slanje 2 poruka istom primatelju dodu u onom redoslijedu u kojem su poslone

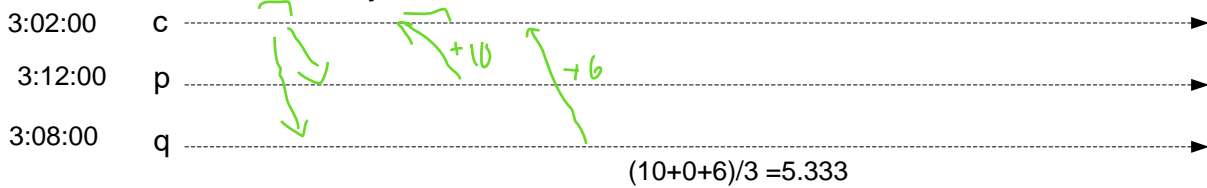


5.4 Prema slici desno rekao bih da je to non-Co jer imamo krizanje linija i  $s^2 \rightarrow s^3$  jer na  $p_1$  imamo  $r^2 \rightarrow r^3$  sto god to znacilo ili funkcioniralo



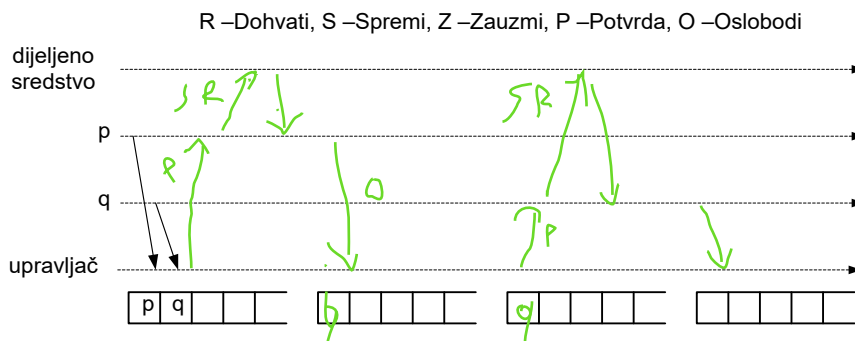
**Zadatak 6.1**

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.



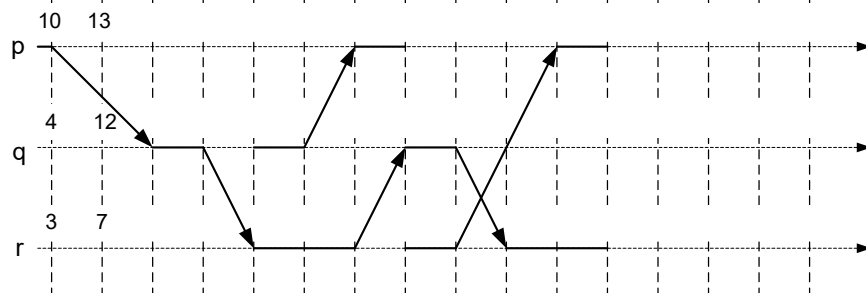
**Zadatak 6.2**

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.



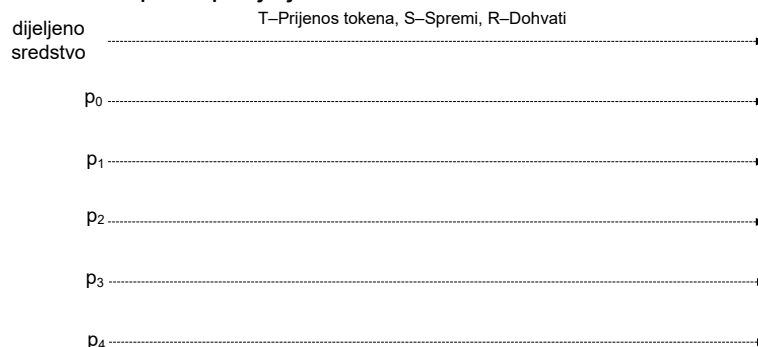
**Zadatak 6.3**

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.



**Zadatak 6.4**

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. Naznačite navedena vremena na dijagramu. Koje je minimalno, a koje maksimalno vrijeme čekanja bilo kojeg procesa u prstenu za pristup dijeljenom sredstvu.



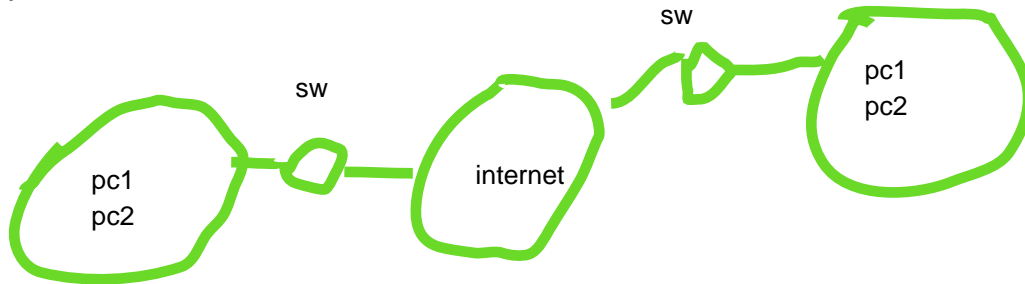


**Zadatak 7.1** Skicirajte i ukratko objasnite slojevitu arhitekturu spleta računala.

**Zadatak 7.2** Navedite i objasnite modele usluga u računalnom oblaku.

**Zadatak 7.3** Što su mikrousluge? Navedite njihova obilježja.

7.1 Skup udaljenih računala koji su slabo povezani internetom i rade na zajedničkom zadatku, različite domene, primjer je BTC network



7.2 klasični (korisnik upravlja svime), IaaS (infrastruktura) (AWS, Azure --> korisnik upravlja aplikacijama, podacima, middlewareom) PaaS (platforma) (Google --> korisnik upravlja aplikacijama, podacima), SaaS (softver) (Gmail --> korisnik upravlja softverom)

7.3 Mikrousluge daju mogućnost skaliranja i razvoja pojedinačnih komponenti, automatski se mogu povećavati, smanjivati, nadomjestiti, u Dockerfilu možemo definirati kakve resurse trebamo, napravimo image i napravimo kontejner koji je samostalna neovisna cijelina, a sustav za upravljanje (skaliranje) Docker kontejnerima je k8s (Kubernetes)