1 Uvod u raspodijeljene sustave

1.1. Objasnite ulogu programskog posredničkog sloja za raspodijeljene sustave.

- Programski posrednički sloj omogućava povezivanje, komunikaciju i suradnju aplikacija, sustava i uređaja te omogućuje interakciju programa na aplikacijskoj razini. Nalazi se između operacijskog sustava i aplikacijskih programa u programskoj arhitekturi sustava. Cilj je olakšati korisnicima i aplikacijama pristup i korištenje udaljenih sredstava na kontroliran i učinkovit način.

1.2. Usporedite migracijsku i relokacijsku transparentnost raspodijeljenog sustava. Koje svojstvo (ne)zadovoljava web poslužitelj i zašto?

- Migracijska transparentnost: prikrivanje promjene lokacije sredstva na način da ta promjena ne utječe na način pristupa sredstvu
- Relokacijska transparentnost: prikrivanje premještanja / kretanja sredstva
- Web poslužitelj zadovaljava migracijsku transparentnost. Mijenjanjem lokacije i/ili mrežnog sučelja mijenja se mrežna adresa, ali se ne mijenja simbolička adresa. Web poslužitelj ne zadovoljava relokacijsku transparentnost jer je poslužitelj stacionaran i ne kreće se tijekom pružanja usluge.

1.3. Objasnite pojam skalabilnosti raspodijeljenog sustava.

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

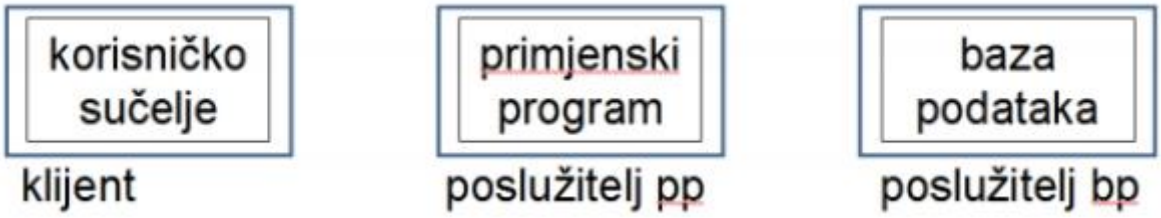
1.4. Kojim aspektima transparentnosti pridonosi sustav imenovanja domena (DNS)?

- Migracijska transparentnost, konkurencijska transparentnost i lokacijska transparentnost.

1.5. Napravite analizu transparentnosti raspodijeljenog sustava elektroničke pošte.

- Primjer gmaila:
  Zadovoljava lokacijsku transparentnost jer korisnik pristupa usluzi preko simboličke adrese. Zadovoljava konkurencijsku transparentnost jer uslugu koristi puno korisnika, a individualan korisnik ima dojam kao da je on jedini. Transparentnost pristupa, prikrivaju se razlike u pristupu sredstvu korisnicima s različitim OS-ima (npr. Mac, MS Windows itd.). Transparentnost na kvar i replikacijska transparentnost, kopije podataka se čuvaju na drugim poslužiteljima za slučaj kada se jedan poslužitelj pokvari, da drugi može preuzeti posao bez da korisnik primjeti da je došlo do kvara.

1.6. Prikažite i objasnite primjer troredne arhitekture weba.



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

1.7. Kakvi bi se problemi pojavili kad bi se više repliciranih poslužitelja weba priključilo na mrežu izravno, a ne putem zastupnika (proxy)?

- Korisnik ne bi znao kojem poslužitelju pristupa i sa svakim sljedećim pristupanjem ne bi bilo zajamčeno pristupanje sadržaju kojem je prethodno pristupio.

1.8. Kakvi su tržišni udjeli web-poslužitelja i web-preglednika?

- Wtf?

1.9. Što je vrijeme odziva za poslužitelja weba?

- Vrijeme odziva je vrijeme koje je potrebno da poslužitelj procesira zahtjev i da odgovor stigne do korisnika.

2 Procesi i komunikacija

2.1. Objasnite na primjeru razliku između perzistentne i tranzijentne komunikacije

-Primjer perzistentne komunikacije bi bio komunikacija porukama. Korisnik 1 šalje poruku korisniku 2 i sustav jamči da će korisnik 2 dobiti tu poruku, bez obzira je li mu upaljen ili ugašen uređaj. Primjer tranzijentne komunikacije je chat sustav baziran na UDP-u gdje korisnik 1 šalje poruku korisniku 2, ali se ne jamči isporuka poruke ako korisnik 2 nije aktivan.

2.2. Objasnite razliku između sinkrone i asinkrone komunikacije.

-Kod sinkrone komunikacije pošiljatelj je blokiran nakon slanja poruke sve do primitka potvrde o isporuci ili sve do primitka odgovora od poslužitelja, ovisno o implementaciji. Kod asinkrone komunikacije pošiljatelj nije blokiran te nastavlja procesiranje odmah nakon slanja.

2.3. Objasnite zašto tranzijentna sinkrona komunikacija potencijalno pati od problema vezanih uz skalabilnost.

-Uzmimo na primjer da imamo jednog korisnika. On pošalje poruku poslužitelju i budući da je komunikacija sinkrona korisnik je blokiran, ne radi ništa dok ne primi odgovor. Ali, budući da je komunikacija tranzijentna moguće je da poslužitelj nije dobio poruku i korisnik tada ulazi u beskonačnu petlju čekanja. Pri povećanjem skalabilnosti se povećava broj korisnika kojima sustav ne radi ispravno.

2.4. Navedite prednosti koje ima operacija slanja zahtjeva koja je neblokirajuća u odnosu na blokirajuću operaciju.

-Korisnik koji šalje zahtjev nije blokiran i može nastaviti s radom dok čeka zahtjev. Za to vrijeme može poslati nove zahtjeve drugim poslužiteljima ili može napraviti neki unutarnji posao.

2.5. Ima li smisla ograničiti broj dretvi za obradu korisničkih zahtjeva na višedretvenom poslužitelju? Zašto?

-Valjda ima smisla jer se ostatak dretvi koristi za npr. obradu unutarnjih poslova.

2.6. Je li poslužitelj koji održava TCP/IP konekciju prema klijentu stateful ili stateless?

-Poslužitelj je stateful jer postoje zahtjevi unutar TCP/IP protokola koji mijenjaju stanje konekcije, npr. ako se korisnik želi odspojiti, poslati će zahtjev za raskid konekcije.

2.7. Navedite prednosti konkurentnog poslužitelja u odnosu na iterativni poslužitelj.

-Prednost je što konkurentni poslužitelj može posluživati više korisnika odjednom i samim time se povećava efikasnost sustava.

3 Sloj raspodijeljenog sustava za komunikaciju među procesima

3.3 Poslužitelj je implementiran pomoću socketa TCP s ograničenjem veličine repa za dolazne zahtjeve na instanci klase ServerSocket (tzv. backlog) na 10. Što se događa s novim zahtjevima ako je prilikom dolaska novog zahtjeva u tom repu već na čekanju 10 zahtjeva?

-Zahtjevi se odbacuju.

3.4. Na koji je način moguće ograničiti broj paralelnih konekcija prema višedretvenom poslužitelju kao što je npr. UpperCaseServer?

- Wtf?

3.5. U tablicama su prikazane metode na klijentskoj i poslužiteljskoj strani socketa TCP. Upišite ispravan redoslijed izvođenja metoda u tablice.

Klijent	**Klijent**
Socket()	Socket()
Write()	Connect()
Read()	Write()
Close()	Read()
Connect()	Close()

Mnemonička metoda ili šalabahter: SCWRC

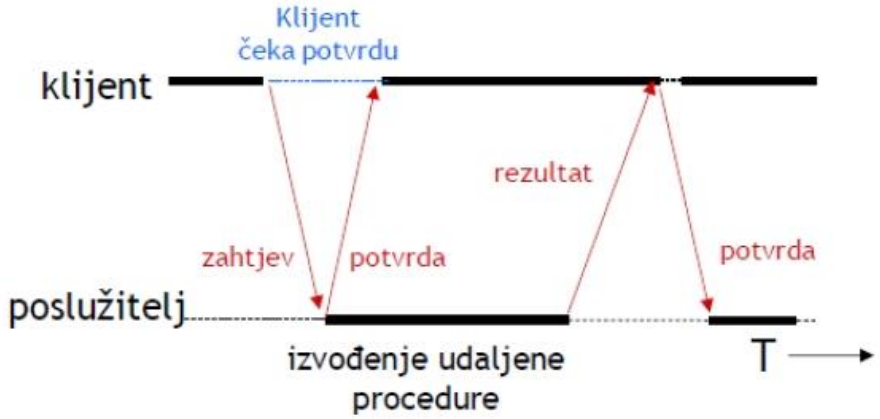
Poslužitelj	**Poslužitelj**
Listen()	Socket()
Socket()	Bind()
Accept()	Listen()
Write()	Accept()
Read()	Socket()
Close()	Read()
Bind()	Write()
	Close()

Mnemonička metoda ili šalabahter: SBLASRWC

3.6. Može li se pomoću UDP-a implementirati protokol za pouzdanu komunikaciju između klijenta i poslužitelja? Ako može, na koji način?

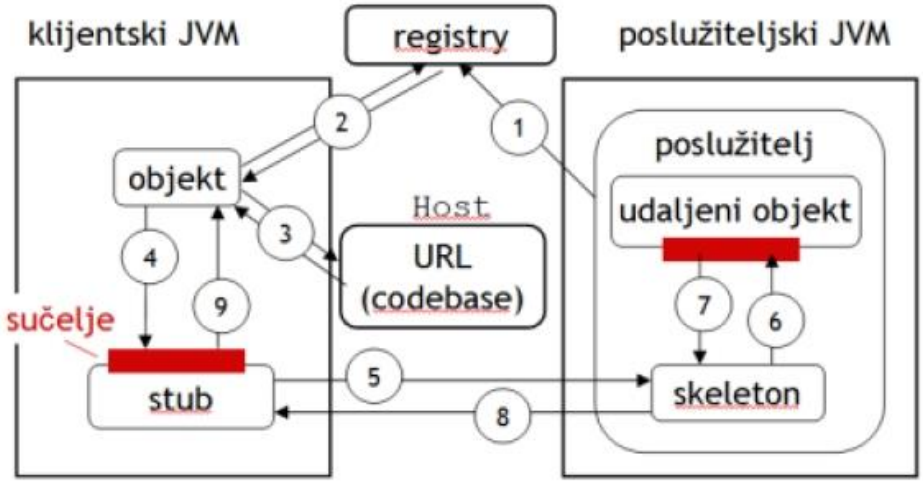
-Može se implementirati pomoću slanja potvrda za svaki UDP paket. Ako korisnik 1 nije dobio potvrdu od korisnika 2 za primitak poslanog UDP paketa nakon određenog vremena, korisnik 1 isponova šalje taj određeni UDP paket.

3.7. 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.

3.8. 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 klasa stub već postoji i dostupna je na klijentskoj strani.



1) Poslužitelj definira codebase udaljenog objekta i registrira ga pod odabranim imenom.
2) Klijent od regstrya traži referencu na udaljeni objekt, i dobiva je.
3) Klijent traži i dobiva klasu stub koristeći codebase.
4) Klijent poziva metodu stuba dostupnu na klijentskom računalu.
5) Stub serijalizira parametre i šalje ih skeletonu.
6) Skeleton deserijalizira parametre i poziva metodu udaljenog objekta.
7) Udaljeni objekt vraća rezultat izvođenja metode skeletonu.
8) Skeleton serijalizira rezultat i šalje ga stubu.

9) Stub deserijalizira rezultat i dostavlja ga klijentu.

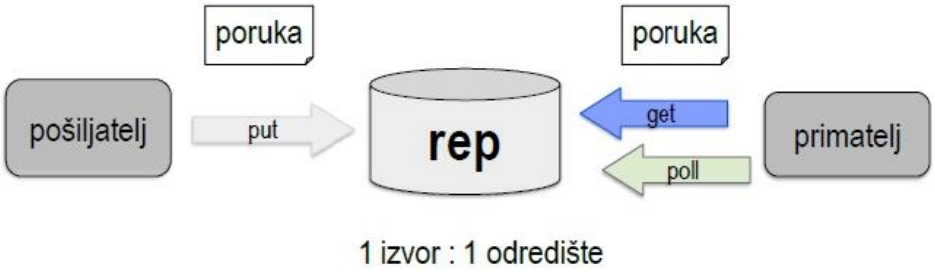
3.9. Ima li smisla implementirati perzistentnu asinkronu komunikaciju pomoću RMI-ja? Zašto?

- wtf?

3.10 Objasnite razliku između transportne adrese i adrese repa u sustavima za komunikaciju porukama?

- Svaki rep ima jedinstveno ime (tzv. adresa repa) neovisno o samoj transportnoj adresi te je potrebna usluga koja povezuje ime repa s transportnom adresom (analogno DNS-u). Adresiranje se izvodi najčešće na nivou sustava, a svaki rep ima jedinstveni identifikator i stoga komunikacija porukama nije anonimna. Pošiljatelj i primatelj ne moraju znati ništa jedan o drugome, no moraju znati adresu repa i format poruke kako bi mogli komunicirati.

3.11 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?



1 izvor : 1 odredište

- 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 trenutnku.

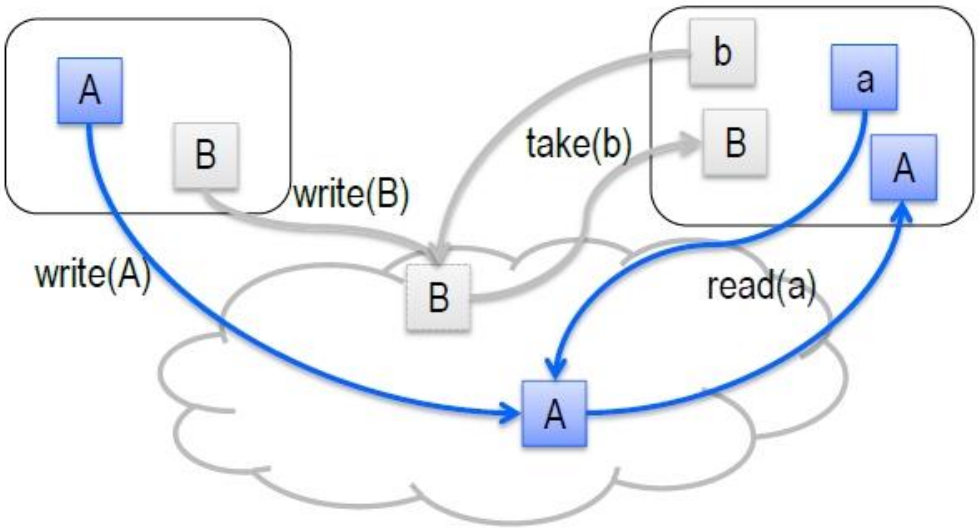
- Primatelj ne mora biti aktivan za vrijeme kada pošiljatelj želi poslati poruku primatelju.

3.12. 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



3.13. Navedite sličnosti i razlike komunikacije na načelu objavi-pretplati i dijeljenog podatkovnog prostora.

- Obje vrste komunikacije su vremenski neovisne zato što pošiljatelj i primatelj ne moraju istovremeno biti dostupni da bi se komunikacija mogla ostvariti. Kod komunikacije porukama pošiljatelj 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.

3.14. Usporedite pretplatu u sustavima objavi-pretplati i predložak u sustavima s dijeljenim podatkovnim prostorom. Može li se koristeći navedenu međuopremu realizirati tzv. anonimnu komunikaciju (pošiljatelj ne zna adresu primatelja) i zašto?

-Može se kombinirati posrednik iz sustava objavi-pretplati s dijeljenim podatkovnim prostorom. Komunikacija u sustavu dijeljenog podatkovnog prostora je anonimna jer se temelji na sadržaju podataka. Dakle, posrednik iz sustava objavi-pretplati komunicira s krajnjim korisnicima na temelju dijeljenog podatkovnog prostora.

3.15. Gdje se filtriraju obavijesti u raspodijeljenom sustavu objavi-pretplati koji koristi preplavljivanje obavijestima?

-Vrši se u krajnjem posredniku prije isporuke obavijesti lokalnim pretplatnicima.

4 Arhitekture web-aplikacija

4.1. Navedite i objasnite najčešće korištene metode protokola HTTP/1.1.

- OPTIONS – služi za informiranje i dohvaćanja mogućnosti resursa i poslužitelja,
- GET – koristi se za dohvaćanje resursa (najčešće u uporabi),
- HEAD – koristi se za dohvaćanje podataka o resursu (npr. veličina, provjera postojanja),
- POST – služi za aktiviranje resursa (npr. slanje podataka obrazaca),
- PUT – postavljanje resursa (npr. promjena podataka u web-uslugama),

- DELETE – brisanje resursa (npr. kod web-usluga),
- TRACE – koristi se za dijagnostiku i
- CONNECT – predviđeno za buduću uporabu.

4.2. Objasnite namjenu jezika CSS. Zašto se pojavila potreba za definiranjem takvog jezika?

- Jezik CSS (*Cascading Style Sheets*) je jezik za definiranje pravila prikaza sadržaja HTML-ovog dokumenta, a nastao je iz potrebe odvajanja sadržaja i prikaza dokumenta.

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

4.4. 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.unizg.hr
```

- Poruka se sastoji od sljedećih elemenata: Početnog retka koji je obvezan, zaglavlja koja se sastoje od polja (neka polja su obvezna), prazan redak i tijela poruke. Obvezni dijelovi su početni redak, dio zaglavlja s poljem Host i prazan redak.
- http://www.fer.unizg.hr/predmet/rassus

4.5. Objasnite ulogu posredničkog poslužitelja s priručnim spemištem koji se nalazi u mreži između web-preglednika i web-poslužitelja. Gdje se sve može nalaziti u mreži?

- Posrednički poslužitelj s priručnim spremištem služi kao privremeno spremište. S njim se ubrzava posluživanje i manje je se tereti mreža između posredničkog poslužitelja i poslužitelja klijenta. Može se nalaziti u lokalnoj mreži klijenta, javnoj mreži ili u lokalnoj mreži poslužitelja.

4.6. Navedite i objasnite prednosti korištenja posredničkog poslužitelja s priručnim spremištem za korisnika, izvorni poslužitelj i komunikacijsku mrežu.

- Prednosti s korisničke strane su brže učitavanje sadržaja koji je spremljen u priručno spremište i smanjuje se korištenje veze između lokalne mreže i ostatka mreže.
- Prednosti s izvornog poslužitelja je privremeno spremanje izračunatih podataka: posrednički poslužitelj može svježe resurse spremiti u svoje priručno spremište i njih slati kao odgovor te na taj način rasteretiti poslužitelja usluge.
- Prednost za komunikacijsku mrežu je smanjenje korištenih mrežnih resursa.

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

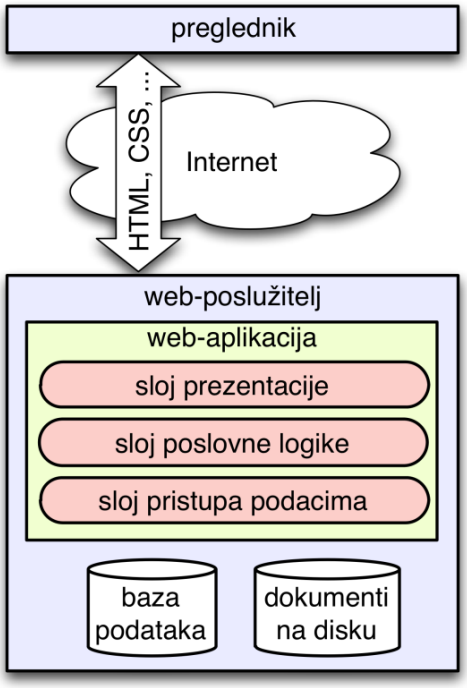
- Zastupnik poslužitelja (*proxy*) posreduje između klijenata i poslužitelja tako da od klijenata prikriva broj i lokaciju poslužitelja te način na koji su povezani (tj. omogućava replikacijsku transparentnost). Također, proxy pruža mogućnost rasterećenja rada poslužitelja tako da nadolazeće zahtjeve preusmjeri na onog poslužitelja koji ja najmanje zaposlen, kriptira ili komprimira podatke i privremeno sprema izračunate podatke i njih šalje kao odgovor, što naravno rasterećuje poslužitelja usluge. Ujedno, može dozvoliti pristup samo nekim uslugama i na taj način poslužitelj usluge ne mora biti toliko siguran jer ga štiti posrednički poslužitelj koji može osiguravati SSL komunikaciju između klijenta i posredničkog poslužitelja, a komunikacija između posredničkog poslužitelja i poslužitelja usluga ne treba biti zaštićena.

4.8. 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.
- Poslužiteljske skripte isto dinamički generiraju HTML-dokumente, ali je razlika u tome što se za svaki zahtjev ne pokreće novi proces i na taj način se štede resursi.

4.9. Skicirajte i objasnite slojevitu arhitekturu web-aplikacija.

- Web-aplikacije su obično organizirane u 3 sloja:
  - sloj prezentacije – služi za prikaz informacija (GUI, HTML, klikovi mišem, ...) i za obradu HTTP-zahtjeva;
  - sloj poslovne (domenske) logike – obrađuje podatke koje je dobio od sloja prezentacije. To je glavni dio sustava koji radi ono za što je sustav namijenjen;
  - sloj pristupa podacima – komunicira s bazom podataka i drugim komunikacijskim sustavima te se brine o transakcijama i o pohrani podataka.

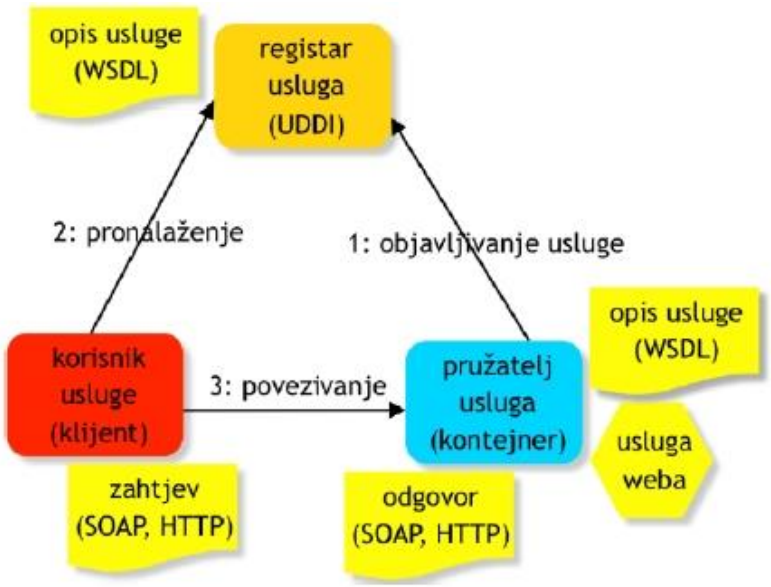


4.10. Koja je prednost korištenja tehnike dugog prozivanja poslužitelja za web-aplikacije?

Prednost je što se u slučajevima kada nema puno poruka ne opterećuje mreža s klijentovim zahtjevima i poslužiteljevim praznim odgovorima. Nego klijent pošalje jedan zahtjev i poslužitelj mu odgovori s porukom tek kada ta poruka stigne od drugog klijenta.

5 Web usluge

5.1. Skicirajte i objasnite arhitekturu web-usluge te interakcije između komponenti web-usluge.



Kada se na pružatelju usluge instalira web-usluga, pružatelj ju prvotno registrira u registru usluga. Koristeći protokol UDDI pružatelj objavi uslugu i registru šalje opis usluge u WSDL-u. Kada klijent želi pronaći uslugu, prvo registru pošalje upit za pronalaženje, a registar mu odgovara opisom usluge koja odgovara zadanim kriterijima. Nakon toga se klijent može povezati s pružateljem usluge jer ima opis usluge. Klijent pomoću protokola HTTP i SOAP šalje zahtjev za pozivom usluge, pružatelj usluge pokreće izvođenje usluge i šalje natrag rezultat u odgovoru HTTP i SOAP protokolima.

5.2. Navedite i ukratko objasnite tri vrste web-usluga.

1. Poziv udaljene procedure – usko povezuje klijenta i poslužitelja jer je usluga usko vezana za programski jezik jer se za svaki programski jezik koristi alat koji automatski iz programskog jezika na svoj način definira elementa WSDL-a.
2. Usluge temeljene na dokumentima / porukama – ne povezuje jako klijenta i poslužitelja odnosno njihove implementacijske jezike jer je fokus na ugovorima koji su propisani WSDL-om.
3. Usluge temeljene na stanju resursa - koristi protokol HTTP pa je sučelje dobro definirano, fokus je na interakciji s resursima koji imaju stanje, a ne na porukama ili operacijama.

5.3. Prikažite arhitekturu i objasnite korištenje usluge Weba.

Valjda isto kao 5.1.?

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

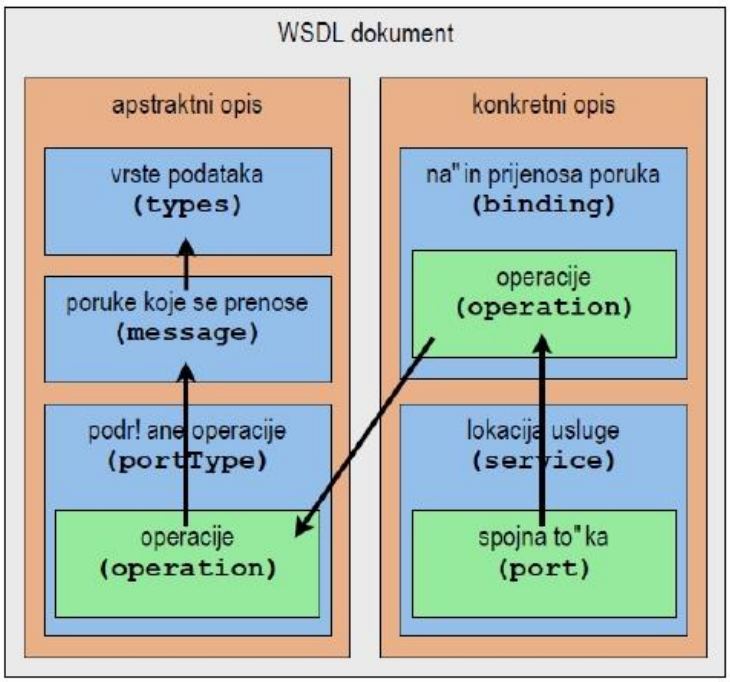
Dva osnovna načina rada protokola SOAP su:

1) Poziv udaljene procedure – služi za prijenos serijaliziranih parametara i rezultata. Korist ovog načina rada je dobro definirano sučelje i tipovi podataka, te prilagodni kod može biti generiran automatski.

2) Razmjena dokumenata / poruka – koriste se XML dokumenti za razmjenu poruka.

5.5. Navedite i objasnite primjenu dokumenta u WSDL-u. Od kojih se dijelova sastoji takav dokument?

Dokument u WSDL-u se koristi za opis web-usluge. Dokument se sastoji od apstraktnog i konkretnog dijela.



Types – definira vrste podataka neovisne o platformi i jeziku

2) Message – definiraju ulazne i izlazne poruke koje se mogu koristiti kao parametri usluga

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

4) Binding – definira kako je konkretna implementacija povezana s operacijama u apstraktnom opisu i definira formt u kojem de se poruke prenositi

5) Service – definirani URI preko kojeg se usluga može pozvati

5.6. Objasnite svojstvo idempotentnosti za web-usluge temeljene na REST-u.

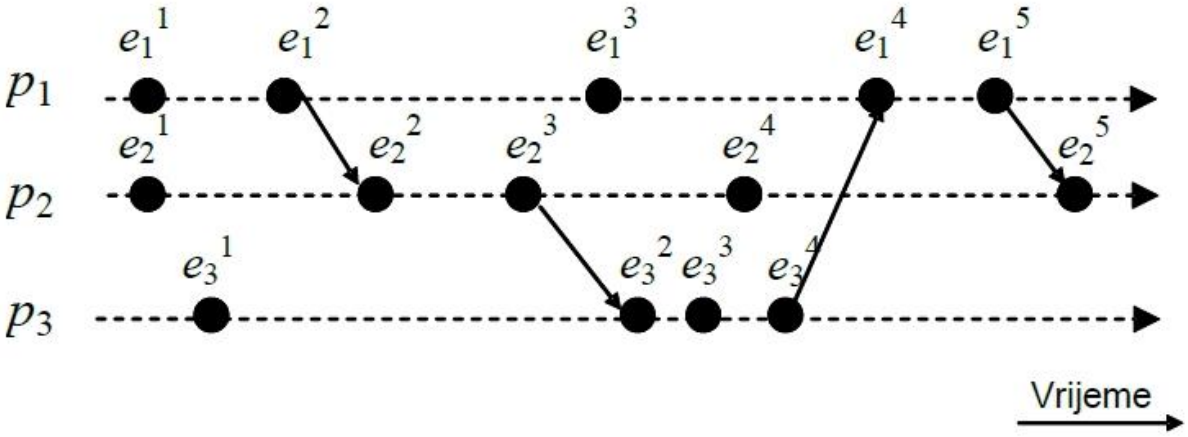
Metoda koja ima svojstvo idempotentnosti se može pozvati više puta, ali rezultat će uvijek biti isti.

6. Model raspodijeljenog sustava

6.1. Za koje je svojstvo raspodijeljenih sustava značajna komunikacijska složenost algoritama? Zašto?

- Za skalabilnost sustava. Na temelju komunikacijske složenosti možemo zaključiti kako raste generirani promet raspodijeljenog sustava s rastom tog sustava.

6.2. 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.

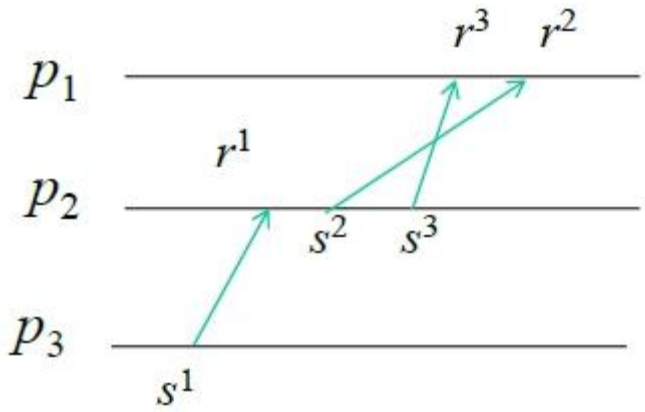


a) e_1^3 -/> e_2^2, zato što:
e_1^2 -> e_2^2, e_1^2 se dogodio prije e_2^2, ali e_1^3 se dogodio nakon e_1^2 stoga ne postoji tranzitivna ovisnost i ne e_2^2 i e_1^3 ne razmjenjuju poruke.

b) e_2^2 -> e_1^5 ?
e_2^2-> e_2^3, e_2^3 -> e_3^2, e_3^2-> e_3^3, e_3^3-> e_3^4, e_3^4-> e_1^4, e_1^4-> e_1^5. Preko tranzitivnosti e_2^2 -> e_1^5

6.3. Objasnite model komunikacijskog kanala koji se temelji na uzročnoj slijednosti.
Vrijedi li za slijedeći primjer CO ili non-CO i zašto?

- Kanal koji osigurava uzročnu slijednost (causal ordering, CO) osigurava da uzročno povezani događaji slanja dviju poruka istom primatelju rezultiraju primanjem u slijedu kojim su poslani.
-

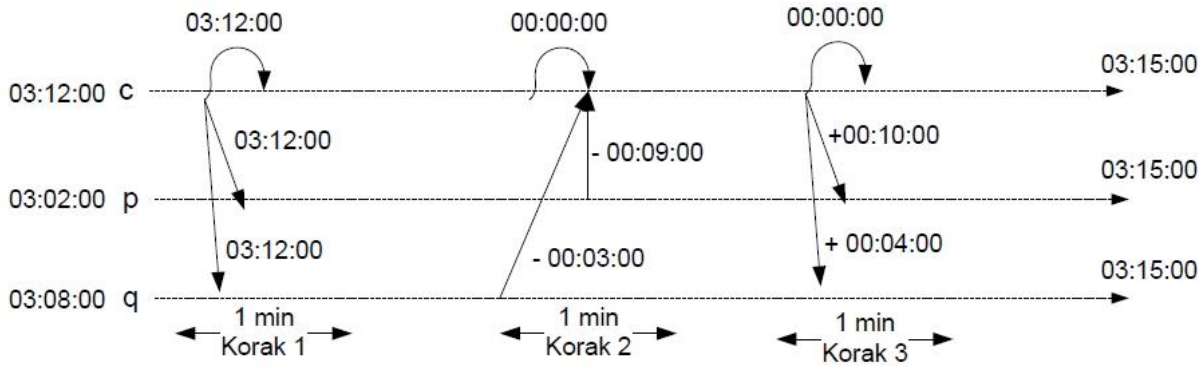


- Non-CO. Proces 2 šalje poruke procesu 1 slijedom s2, s3. Proces 1 umjesto da te poruke dobije istim slijedom kojim su poslane, on ih dobije obrnutim redoslijedom.

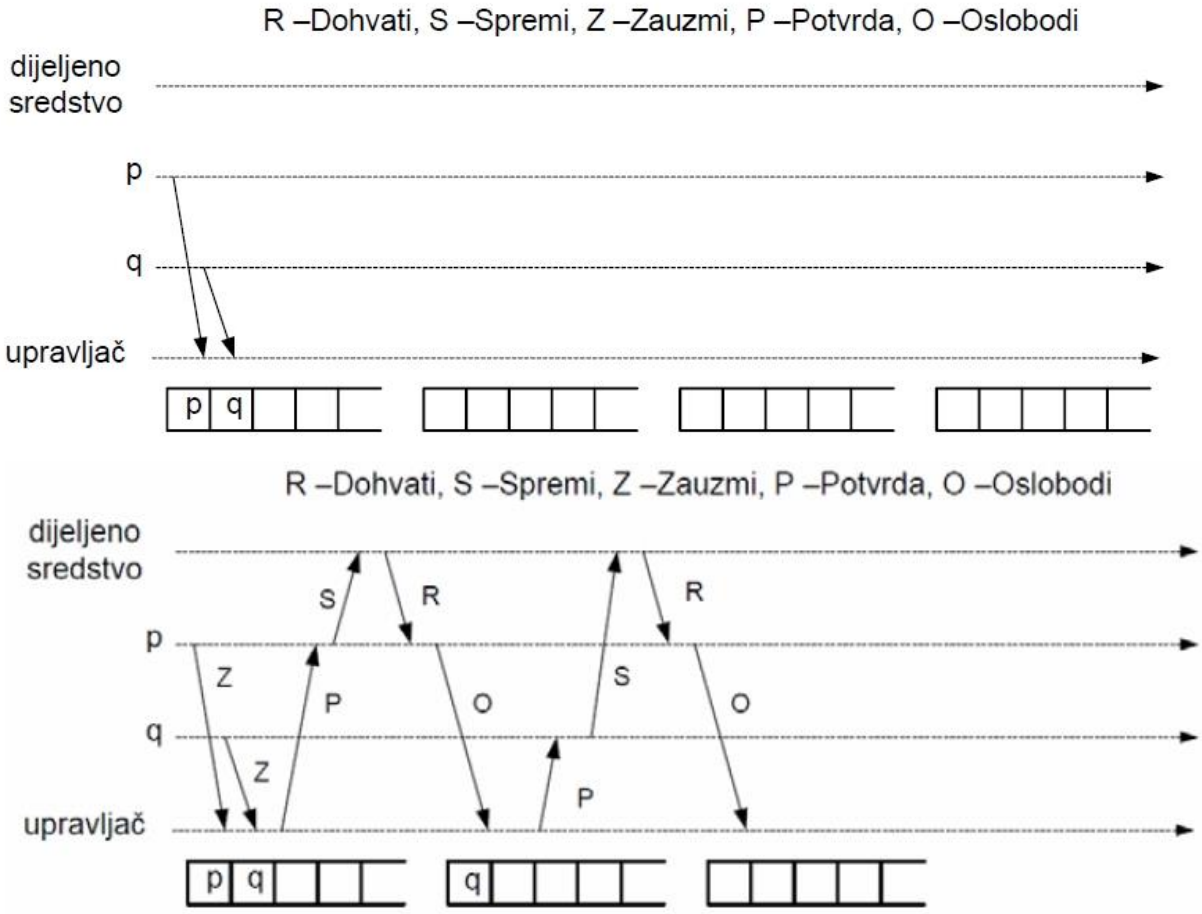
7 Sinkronizacija procesa u vremenu

7.1. 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.



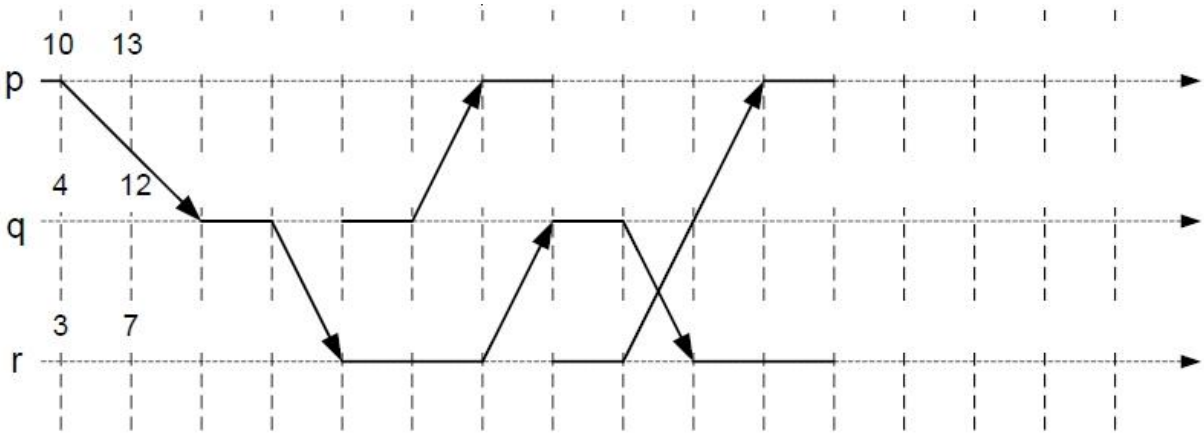
7.2. Opišite postupak međusobnog isključivanja dvaju procesa (p i q) primjenom središnjeg upravljača s repom čekanja tako da nacrtate redoslijed operacija i objasnite ih. Nakon zauzimanja dijeljenog spremnika, proces provodi jednu operaciju čitanja ili pisanja nad dijeljenim spremnikom.

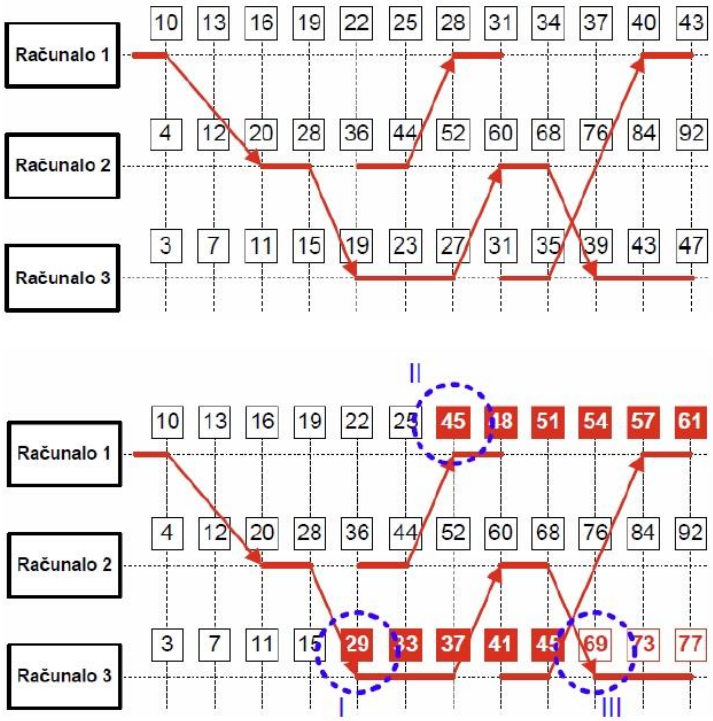


• Proces p šalje zahtjev za zauzimanje sredstva, zahtjev se sprema u rep
• Proces q šalje zahtjev za zauzimanje sredstva, zahtjev se stavlja u rep
• Kako je zahtjev od R0 stigao prije, upravljač šalje potvrdu R0 i uklanja njegov zahtjev iz repa
• Proces p provodi operaciju pisanja

• 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

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.





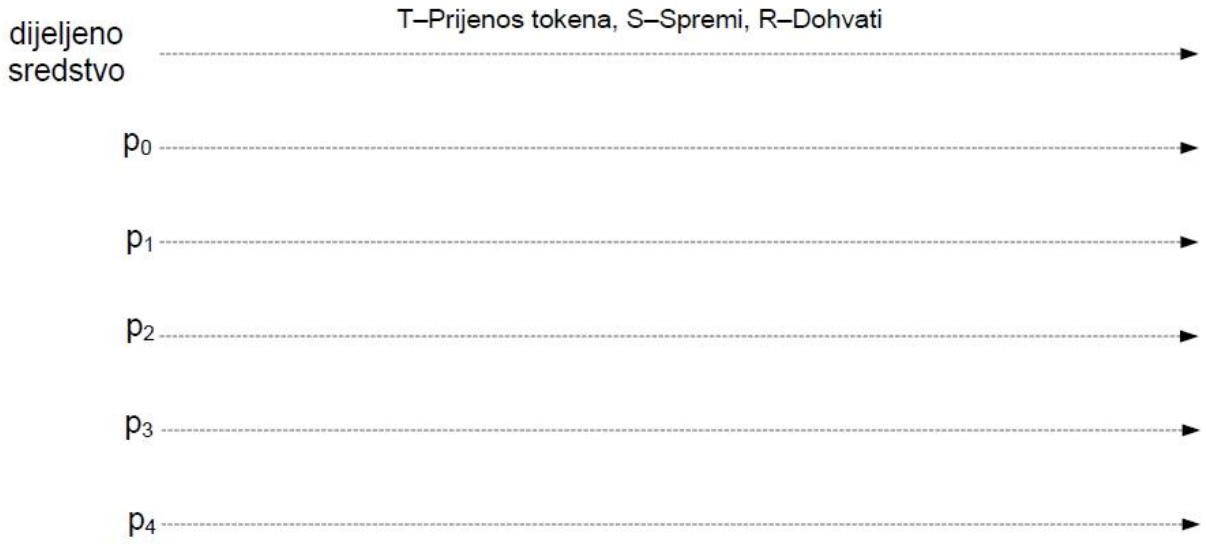
Trenutak I: Računalo 3 prima poruku od računala 2 s oznakom vremena T_P=28 koja je veća od lokalne oznake vremena T_L= 19. Lokalni sat se pomiče na vrijednost T_P+1=29.

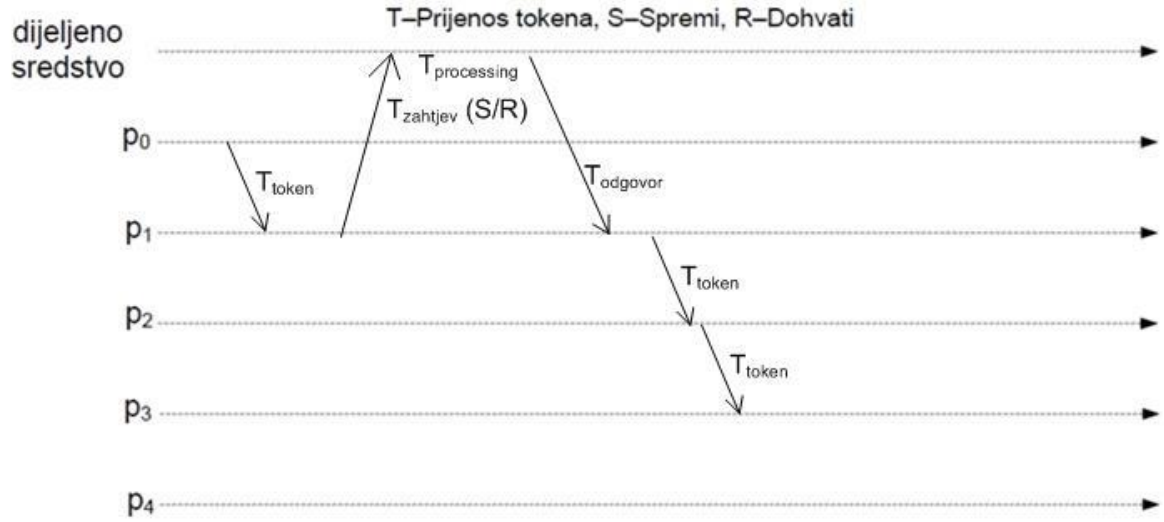
Trenutak II: Računalo 1 prima poruku od računala 2 s oznakom vremena T_P=44 koja je veća od lokalne oznake vremena T_L= 28. Lokalni sat se pomiče na vrijednost T_P+1=45.

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.

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. 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.

8 Konzistentnost i replikacija podataka

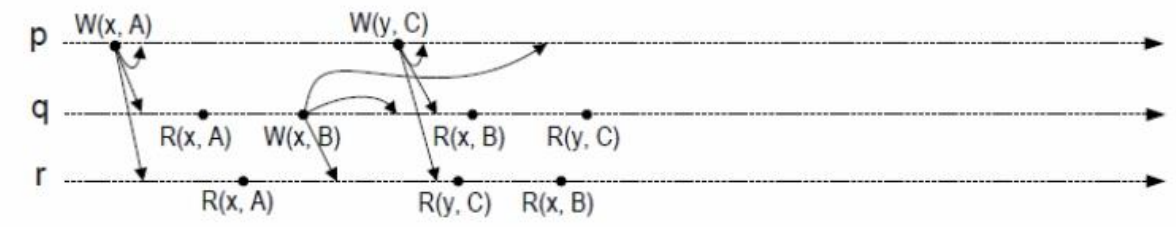
8.1. Objasnite što je replika podatka, a što je nekonzistentnost replike podatka.

Replika podatka je jedna kopija podatkovnog objekta u raspodijeljenoj okolini. Nekonzistentnost replika podataka se javlja kada dvije ili više replika u raspodijeljenoj okolini u nekom trenutku u vremenu se nalaze u različitim stanjima.

8.2. Objasnite što je povezana konzistentnost operacija u raspodijeljenim sustavima? Na primjeru procesa p, q i r prikažite slijed operacija čitanja i pisanja koji je a) u skladu i b) nije u skladu s načelima povezane konzistentnosti.

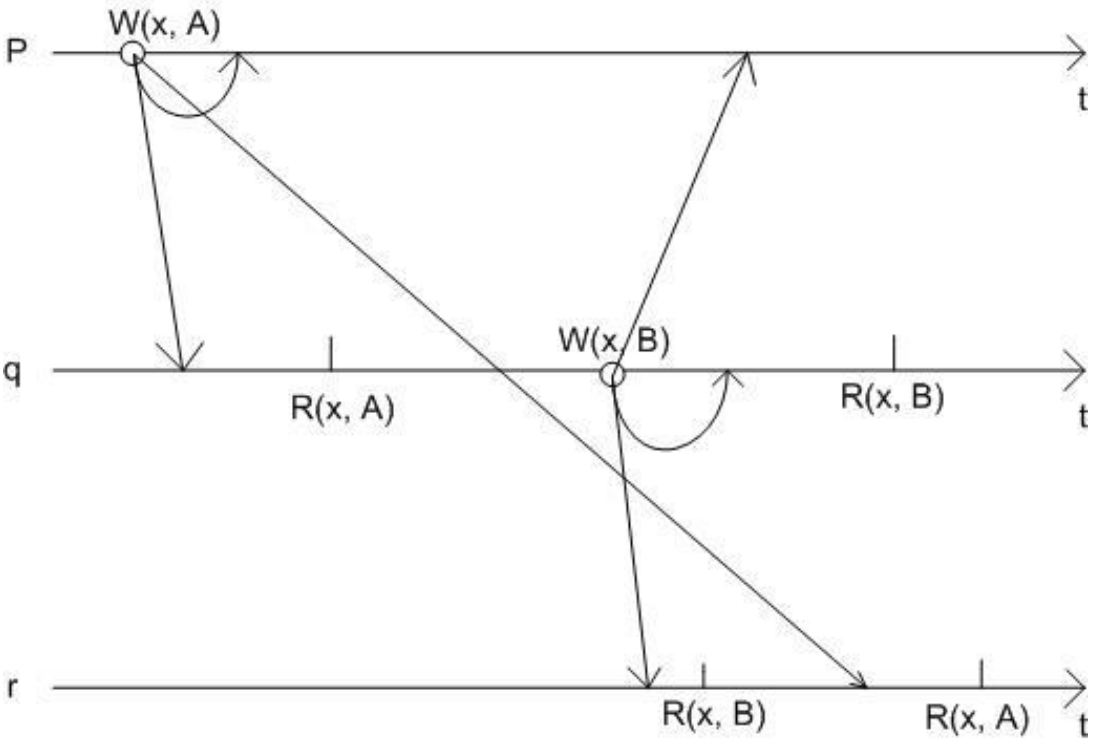
Model povezane konzistentnosti zahtijeva da povezane operacije svi procesi moraju vidjeti u istom redoslijedu. Redoslijed izvođenja povezanih operacija upisivanja vidljiv je svim procesima na jednak način, dok redoslijed izvođenja operacija upisivanja koje nisu povezane, svakom procesu može biti prikazan na drukčiji način.

A)



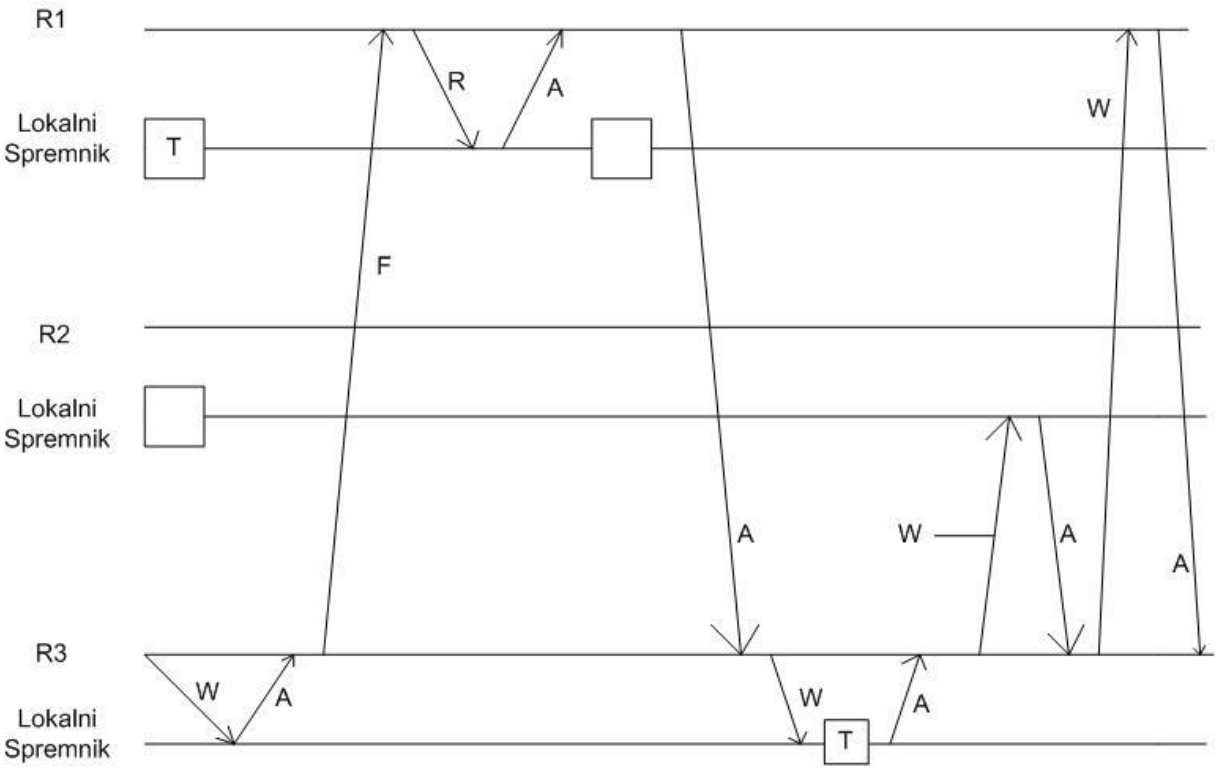
Proces p zapisuje A na mjesto x, zatim proces q čita A s lokacije x i nakon toga na lokaciju x zapisuje B. Vidi se da su operacije R(x, A) i W(x, B) povezane jer očitana vrijednost A može utjecati na izračun vrijednosti B. Operacije W(x, B) i W(y, C) nisu povezane jer upisuju dva različite podatka na dvije različite lokacije. Dakle, svi procesi moraju vidjeti slijed povezanih operacija u istom redoslijedu tj. mora se prvo izvršiti R(x, A) pa onda W(x, B). Vidimo da se na procesu q tako i izvršava, ali vidimo da je taj redoslijed i na procesu r.

B)



Proces p zapisuje na lokaciju x, podatak A. Zatim proces q čita A s lokacije x i nakon toga zapisuje na lokaciju x podatak B. Vidimo da su operacije R(x, A) i W(x, B) povezane i svaki proces bi trebao vidjeti taj slijed operacija. No, vidimo da proces r prvo zapisuje B na lokaciju x, zatim čita A s lokacije x i zato ovaj primjer ne zadovoljava povezanu konzistentnost.

8.3. Raspodijeljeni sustav uključuje tri računala (R0, R1, R2) s lokalnim spremnicima. U lokalnom spremniku računala R1 nalazi se trajna replika dokumenta, dok se u lokalnom spremniku računala R2 nalazi obična replika dokumenta. Korisnik putem računala R3 provodi operaciju pisanja nad dokumentom primjenom postupka lokalnog obnavljanja stanja replike. Skicirajte i objasnite korake postupka.



Korisnik na računalu R3 želi pristupiti promijeniti saržaj podatka (replike). Prvo pristupa svom lokalnom spremniku da provjeri je li se tamo nalazi podatak. Nakon što utvrdi da se ne nalazi od glavnog poslužitelja zatraži repliku. Glavni poslužitelj (R1) dohvati repliku iz svog lokalnog spremnika i proslijedi ju korisniku. Korisnik lokalno obavlja izmjenu replike i on sam postaje glavni poslužitelj. Nakon izmjene obavještava ostale poslužitelje s novom verzijom replike.

8.4. U sustavu replika koji se sastoji od glavnog poslužitelja i n=4 podjednako opterećena pomoćna poslužitelja, odredite metodu održavanja konzistentnosti replika za koju će prosječno mrežno (prometno) opterećenje poslužitelja L biti najmanje. Pri tome pretpostavite da korisnike isključivo poslužuju pomoćni poslužitelji, da je prosječna frekvencija upita fu=5 upita/s, prosječna frekvencija promjena fp=1 promjena/min te da su prosječne veličine upita/odgovora, operacija za promjenu sadržaja i replika lp=1kB, lo=50 kB i lr=100 kB. Usporedite dobivena opterećenja s centraliziranim slučajem kada korisnike poslužuje glavni poslužitelj.

PUSH metoda s prosljeđivanjem obavijesti o promjenama: Nakon svake promjene, glavni poslužitelj šalje svakom pomoćnom poslužitelju obavijest o promjenama. Za prvi sljedeći korisnički zahtjev, svaki od pomoćnih poslužitelja šalje glavnom poslužitelju upit za novom replikom na koji mu glavni poslužitelj odgovara novom verzijom replike.
$\mathbf{L_1 = n \cdot f_p \cdot l_p + n \cdot f_p \cdot l_p + n \cdot f_p \cdot l_r} = n \cdot f_p \cdot (2 \cdot l_p + l_r) = 6{,}8$ kB/s

PUSH metoda s prosljeđivanjem operacija za promjenu sadržaja: Nakon svake promjene, glavni poslužitelj šalje svakom pomoćnom poslužitelju obavijest o operacijama koje je potrebno izvršiti nad prethodnom verzijom replike da bi se ona dovela u konzistentno stanje.
$\mathbf{L_2 = n \cdot f_p \cdot l_o} = 3{,}33$ kB/s

PUSH metoda s prosljeđivanjem cjelokupnog sadržaja replika: Nakon svake promjene, glavni poslužitelj šalje novu verziju replike svakom pomoćnom poslužitelju.
$L_3 = n \cdot f_p \cdot l_r = 6{,}67$ kB/s

PULL metoda: Nakon svakog upita, pomoćni poslužitelj provjerava kod glavnog poslužitelja je li došlo do promjene stanja replike koju pohranjuje lokalno. U 1/(n·fp) od 1/fu slučajeva će poslužitelj odgovoriti novom verzijom replike, a u 1/(fu - n·fp) od 1/fu slučajeva će odgovoriti porukom da nije došlo do promjene.
$L_4 = f_u \cdot l_p + n \cdot f_p \cdot l_r + (f_u - n \cdot f_p) \cdot l_p = 16{,}6$ kB/s

Centralizirani slučaj: Glavni poslužitelj odgovara replikom na svaki korisnički upit.
$L_5 = f_u \cdot (l_p + l_r) = 505$ kB/s

Min {L1, L2, L3, L4, L5} = L2.

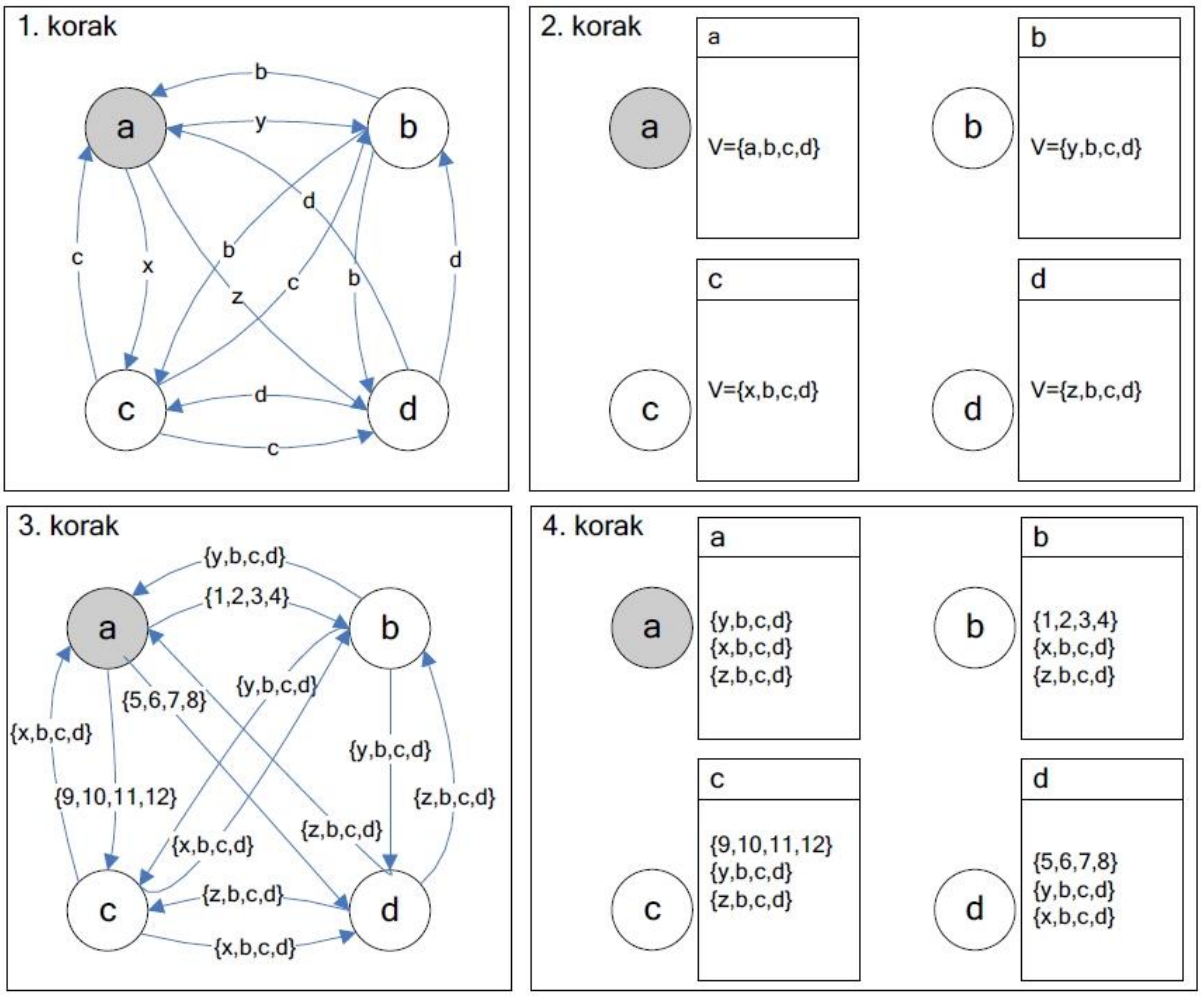
9 Otpornost na neispravnosti

9.1. Objasnite razliku između ispada sustava i neispravnosti u sustavu.
-Ispad sustava je stanje sustava koje se detektira kroz nemogudnost korištenja jedne ili više njegovih usluga. Posljedica je neispravnosti te ukazuje na nju. Neispravnost je nedostatak u programskom kodu, oblikovanju sustava ili komunikacijskom kanalu koji može uzrokovati ispad sustava.

9.2. Pretpostavite da grupa procesa treba postići sporazum. U slučaju da su dva procesa grupe u stanju bizantskog ispada, koji je minimalni ukupni broj procesa u grupi za postizanje sporazuma?
-N = 3k+1 gdje je k broj procesa u bizantskom ispadu. N = 3*2+1 = 7

9.3. Objasnite razliku protokola three-phase commit u odnosu na two-phase commit.
-*Three-phase commit* protokol ima jedno stanje više od *two-phase commit* protokola. To stanje je PRECOMMIT. U *two-phase commit* protokolu postoji problem blokirajućeg stanja. Stanje PRECOMMIT riješava taj problem. Koordinator nakon odluke za izvođenje operacije šalje poruku PREPARE_COMMIT na koju procesi odgovaraju s READY_COMMIT. Nakon što primi poruku READY_COMMIT od svih procesa, koordinator šalje GLOBAL_COMMIT. Ako koordinator ispadne, procesi se međusobno mogu dogovoriti koja im je sljedeća akcija i time je riješen problem blokirajućeg stanja.

9.4. U grupi od 4 procesa (p_1, p_2, p_3 i p_4) proces p_1 je neispravan (pretpostavite bizantski ispad). Grupa procesa želi postići sporazum o identifikatorima ostalih procesa grupe. U koracima 1 i 3 procesi međusobno razmjenjuju podatke, a u koracima 2 i 4 prikupljaju i analiziraju primljene podatke. Nacrtajte na slici podatke koje procesi razmjenjuju u koracima 1 i 3, a za korake 2 i 4 navedite podatke koje pojedini proces ima na raspolaganju radi donošenja odluke o sporazumu.



10 Vrednovanje performansi raspodijeljenih sustava

10.1 Disk za trajno spremanje podataka ispunjava 50 zahtjeva u sekundi. Srednje vrijeme obrade zahtjeva operacija pisanja i čitanja je 10 ms. Disk ima prosječno 1 zahtjev u repu. Koliko je prosječno vrijeme čekanja na obradu zahtjeva?

Propusnost sustava je X = 50 z/s.

Srednje vrijeme obrade zahtjeva je S = 10 ms/z.

Broj zahtjeva u repu je Q = 1 z.

Vrijeme zadržavanja zahtjeva u sustavu je R = Q / X = (1 z) / (50 z/s) = 20 ms.

Vrijeme zadržavanja R uključuje vrijeme čekanja u repu (W) i vrijeme obrade zahtjeva (S): R = W+S.

Vrijeme čekanja na obradu je W = R – S = 20 ms – 10 ms = 10 ms.

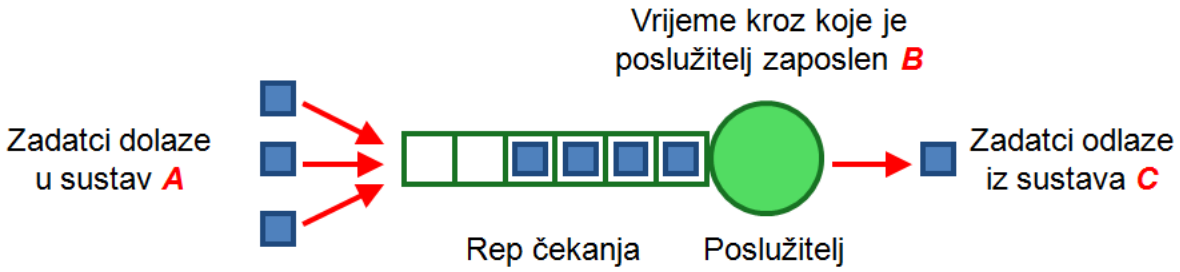
10.2 Web aplikacija uključuje podršku korisnicima putem chat usluge. Kupci sami odabiru jedan od 10 repova čekanja. Mjerenja pokazuju da zahtjevi prosječno dolaze 3 upita u minuti te da svaki kupac prosječno čeka 3 minute u repu i prosječno provodi 2 minute u konverzaciji. Koliko je srednje vrijeme zadržavanja kupaca za zadani sustav?

Tw = 3 min

Ts = 2 min

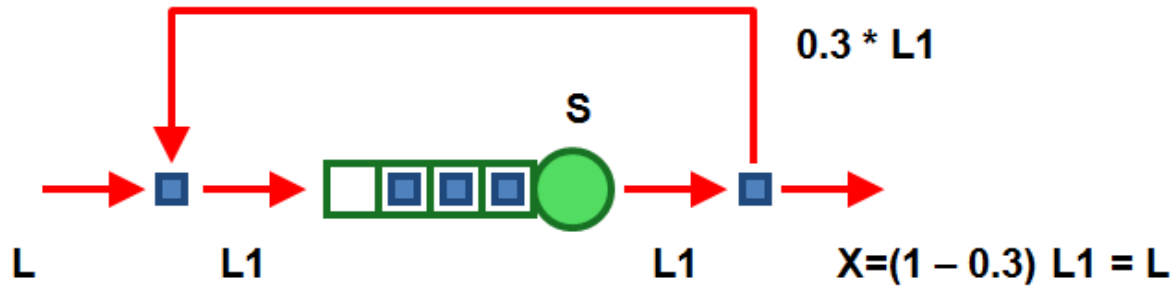
T = Ts + Tw = 5 min

10.3 Prikažite elemente osnovnog modela repa čekanja. Koje su osnovne veličine, a koje izvedene u modelu repa čekanja ? Kako je definirano stacionarno stanje sustava?



Osnovne veličine modela su vrijeme promatranja (T), broj dolazaka (A), broj odlazaka (C) i vrijeme zaposlenosti poslužitelja (B).
Izvedene veličine modela su ulazni ritam (L=A/T), izlazni ritam (X=C/T), srednje vrijeme posluživanja (S=B/C) i zaposlenost poslužitelja (U=B/T).
U stacionarnom stanju sustava je X = L.

10.4 Upiti dolaze na poslužitelj s učestalošću od 12 upita u sekundi te zahtijevaju 0,75 sekundi za obradu. Za 30 % paketa dogodi se pogreška pri obradi te se oni moraju ponovno obraditi. Izračunajte koliko vremena paket prosječno provede u sustavu?



Broj pristiglih upita u sekundi L = 0.5 p/s
Prosječno vrijeme obrade upita S = 0.75 s/p
Vjerojatnost pogreške pri obradi p = 0.3
L1 = p L1 + L => L1 = L / (1 – p)
L1 = L / (1 – p) = 0.5 / 0.7 = 0.714 p/s
Prosječna zaposlenost poslužitelja
U = L1 * S = 0.714 p/s * 0.75 s/p = 0.536 (53.6 %)
Srednje vrijeme čekanja u repu
W = S * U / (1 – U) = 0.866 s/p
Srednje vrijeme zadržavanja na poslužitelju

R1 = W + S = 0.866 s/p + 0.75 s/p = 1.616 s/p
Prosječno vrijeme zadržavanja u sustavu R = R1 / (1 – p) = 2.31

11 Sustavi s ravnopravnim sudionicima

11.1. Usporedite svojstva centraliziranih i decentraliziranih raspodijeljenih sustava na primjeru.
-Centralizirani raspodijeljeni sustavi temelje se na modelu klijent-poslužitelj. U sustavu postoji koordinator koji prihvaća zahtjeve i raspodjeljuje ih među ostalim procesima u sustavu. Osnovna prednost centraliziranih sustava u smislu performansi sustava je vrlo kratko vrijeme odziva sustava. No, njegova mana je jedinstvena točka ispada sustava (koordinator) i velik broj računala potreban za izvršavanje posla. Primjer takvog sustava je web tražilica. S obzirom da su kolekcije dokumenata prilično velike, potreban je i veliki broj računala za održavanje njihovog indeksa. Npr. za 100 TB tekstualnih dokumenata generira se indeks veličine 25 TB za čije održavanje treba oko 3.000 računala. Stoga je potrebna infrastruktura izrazito složena i skupa, a generira i izrazito visoke troškove održavanja.
Gnutella je primjer potpuno decentraliziranog sustava koji pretražuje susjedne čvorove s ciljem pronalaska željene datoteke. Sustav se sastoji od peerova koji su međusobno ravnopravni tj. korisnici su ujedno i poslužitelji i klijenti. Svi *peer*ovi sudjeluju u procesu pretraživanja (ne postoji centralizirani indeks), a posebno je pogodno rješenje za pronalaženje datoteke koje su replicirane na velikom broju *peer*ova. Ne postoji posebna infrastruktura niti potreba za održavanjem sustava, a time niti jedinstvena točka ispada. Glavni nedostatak ovog rješenja je velika količina generiranog mrežnog prometa pri pretraživanju, a sustav ne može garantirati pronalazak tražene datoteke.

11.2. Kako se izvodi pretraživanje kod strukturiranih, a kako kod nestrukturiranih sustava sustava P2P (*peer-to-peer*)? Koji od ovih sustava su skalabilni i zašto?
-U nestrukturiranim sustavima P2P, podatak je pohranjen na *peer*u koje ga kreira, a njegova kopija može biti pohranjena i na nekim drugim *peer*ovima u mreži. Zbog toga se u ovim sustavima pretraživanje izvodi preplavljivanjem mreže i slučajnim izborom (*random walk*). Kod strukturiranih sustava P2P, podatak je pohranjen na peeru koji je zadužen za ključ tog podatka. Pretraživanje se provodi traženjem podatka po njegovom ključu. Strukturirani sustavi su skalabilni zato što se kod njih pretraživanje odvija u *log(n)* koraka, gdje je n broj *peerova* u mreži.

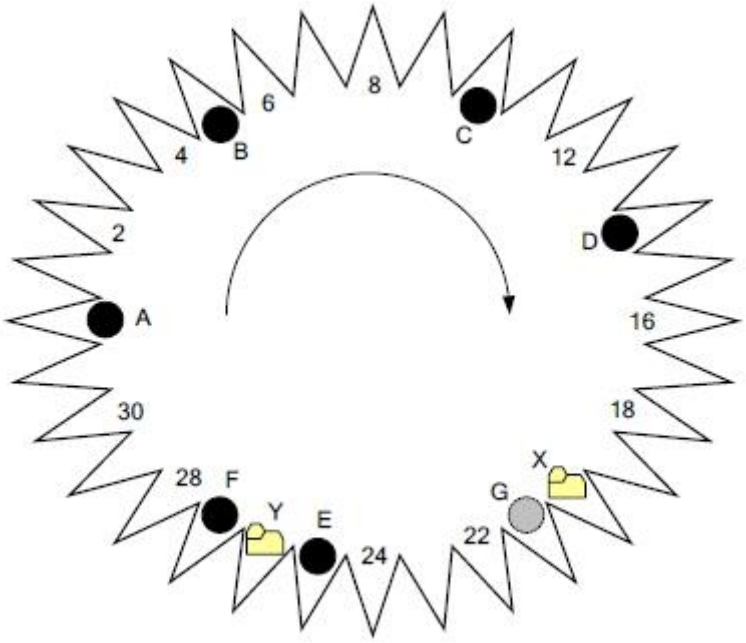
11.3. Na slici je prikazana mreža Chord koja se sastoji od 6 čvorova (A, B, C, D, E i F) i koristi prostor identifikatora duljine N=32 (dovoljno je m=5 bita za kodiranje). Ukoliko je H1(A)=0, H1(B)=5, H1(C)=10, H1(D)=14, H1(E)=25 i H1(F)=27, odgovorite na sljedeća pitanja:

a) Definirajte tablice usmjeravanja na čvorovima *A* i *F*.

b) Na kojem će se čvoru pohraniti podatak *X* s ključem $H_2(X)$=20?

c) Odredite slijed čvorova preko kojih se usmjerava upit od čvora *A* s ciljem pronalaska podatka *Y* s ključem $H_2(Y)$=26.

d) Dodan je novi čvor *G* ($H_1(G)$=21) u mrežu. Što će se promijeniti u tablici usmjeravanja čvora *A*?



a)

Čvor A:

i = 0,1,2,3,4 (i ide od 0 do m-1)

(k+2^i)

k = H(A) = 0

i = 1

(0+1) = 1, nema čvora na jedan. Prvi sljedeći je 5 tj. čvor B.

(0+2) = 2, nema čvora na 2. Prvi sljedeći je 5 tj. čvor B.

(0+4) = 4, nema čvora na 4. Prvi sljedeći je 5 tj. čvor B.

(0+8) = 8, nema čvora na 8. Prvi sljedeći je 10 tj. čvor C.

(0+16) = 16, nema čvora na 16. Prvi sljedeći je 25 tj. čvor E.

Tablica usmjeravanja čvora A:

0	5
1	5
2	5
3	10
4	25

Čvor F

i = 0,1,2,3,4 (i ide od 0 do m-1)

(k+2^i)

k = H(F) = 27

i = 1

(27+1) = 28, nema čvora na 28. Prvi sljedeći je 0 tj. čvor A.

(27+2) = 29, nema čvora na 28. Prvi sljedeći je 0 tj. čvor A.

(27+4) = 31, nema čvora na 31. Prvi sljedeći je 0 tj. čvor A.

(27+8) = 35 = (aritmetika modulo 31) = 32 + 3 = 0+3, nema čvora na 3. Prvi sljedeći je 5 tj. čvor B.

(27+16) = 43 = 31 + 11, nema čvora na 11. Prvi sljedeći je 14 tj. čvor D.

Tablica usmjeravanja čvora F:

0	0
1	0
2	0
3	5
4	14

b)

H2(X) = 20

Nema čvora na 20, prvi sljedeći je 25. Podatak X će se pohraniti na čvoru E.

c)

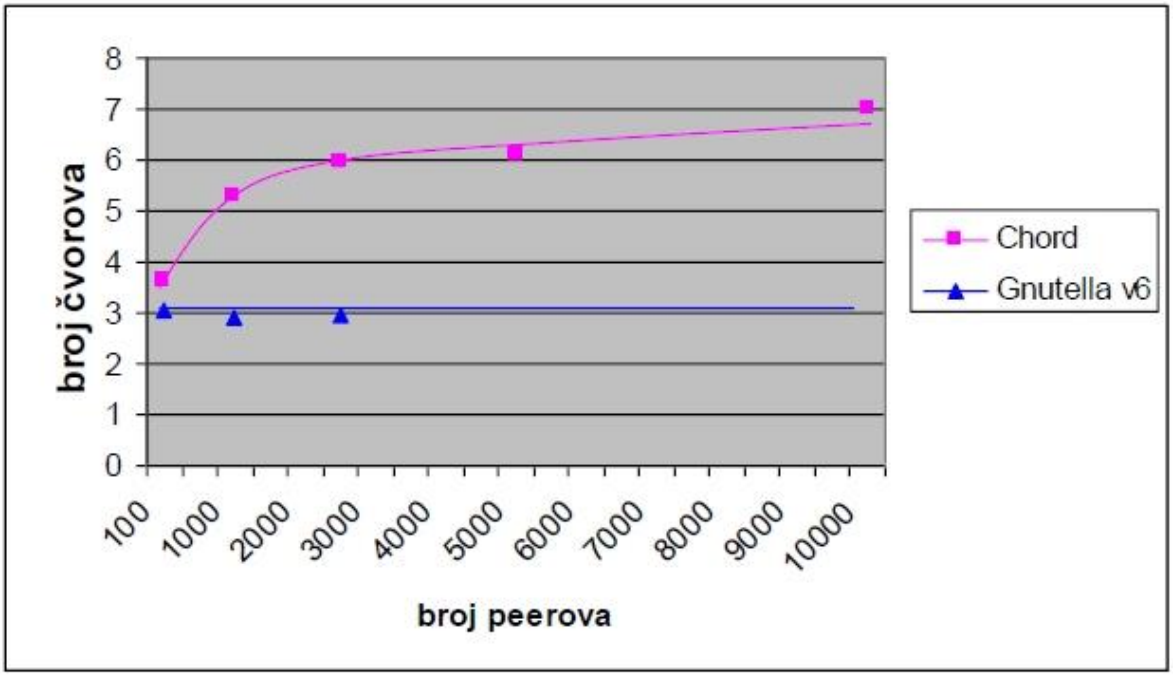
A čvor: lookup(26) -> A gleda u tablicu usmjeravanja i najveći broj kojeg ima je 25. Proslijeđuje upit njemu. Čvor 25 proslijeđuje upit čvoru zaduženom za 26, a to je njegov sljedbenik 27.

d)

Zadnji redak tablice usmjeravanja čvora A će se promijeniti iz 25 u 21.

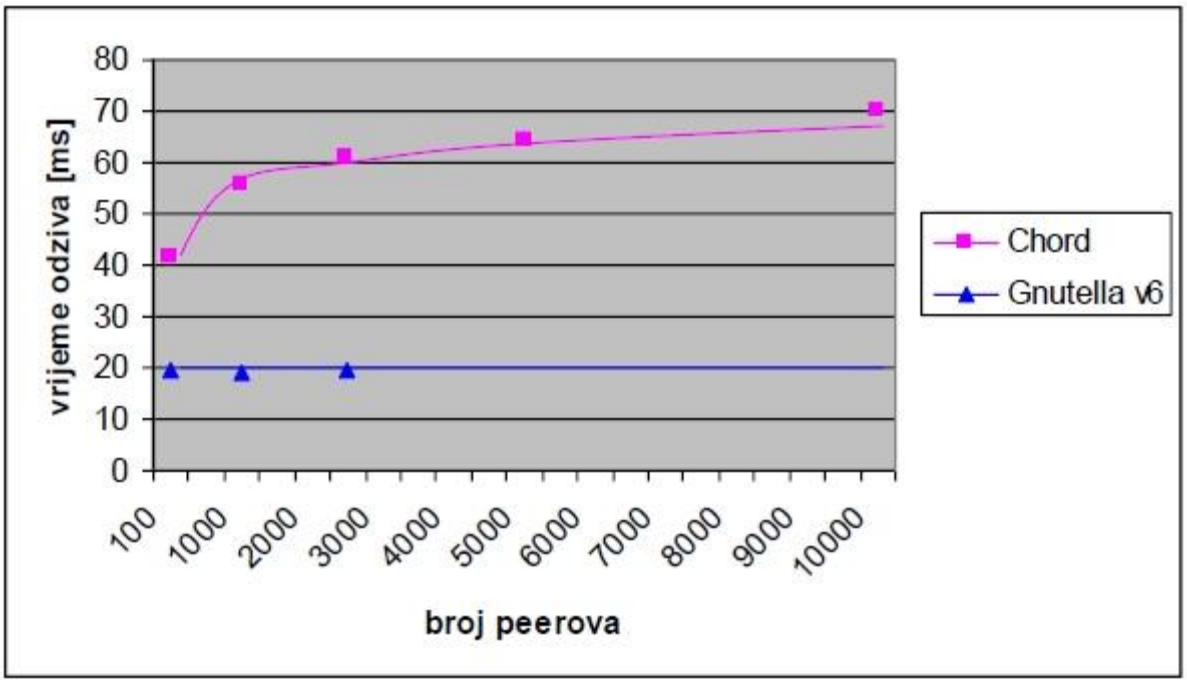
11.4. Komentirajte rezultate eksperimenta kojim se ispituje svojstvo skalabilnosti protokola Gnutella i Chord u statičnom scenariju. Kako objašnjavate krivulje kojima se prikazuje prosječan broj čvorova po upitu i prosječno vrijeme odziva?

-



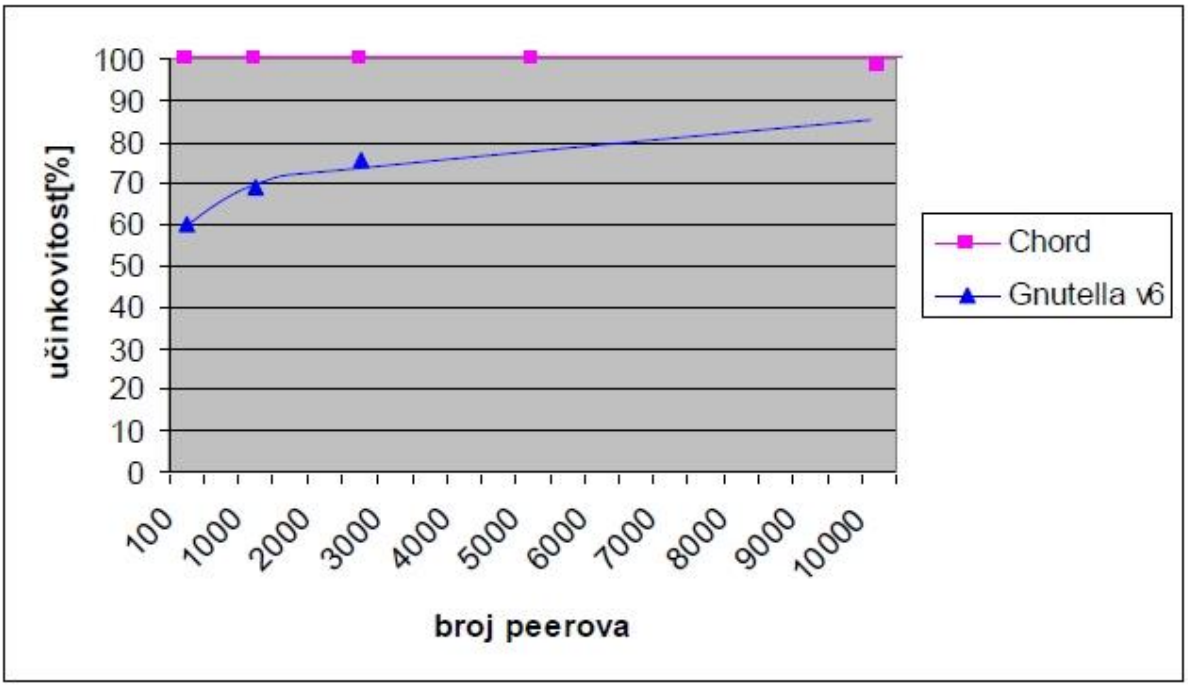
Slika 11.15. Prosječan broj kontaktiranih čvorova po upitu

Vidljivo je da s povećanjem broja peerova u mreži Chord raste broj skokova i vrijeme odziva. Taj porast nije linearan, već se uočava logaritamski porast, što dokazuje da je složenost pretraživanja kod Chorda O(log *n*) pri čemu je *n* broj peerova u mreži. Učinkovitost pretraživanja, tj. postotak uspješno odgovorenih upita je velika što pokazuje slika 11.17.



Slika 11.16. Prosječno vrijeme odziva

Iz slika 11.15. i 11.16. vidljivo je da je kod Gnutelle v6 gotovo jednako vrijeme odziva i broj skokova bez obzira na broj peerova u sustavu, međutim učinkovitost pronalaska podataka raste s povećanjem broja peerova zbog veće povezanosti peerova u mreži što je uočljivo iz slike 11.17.



Slika 11.17. Postotak uspješno odgovorenih upita

12 Grozdovi i spletovi računala

12.1. Skicirajte i ukratko objasnite slojevitu arhitekturu spleta računala.

-**Sloj osnovnih sredstava** upravlja osnovnim funkcionalnostima spleta računala. On implementira lokalne operacije koje su specifične svakom sredstvu po vrsti i implementaciji te omogućuje korištenje tih operacija na višim slojevima.

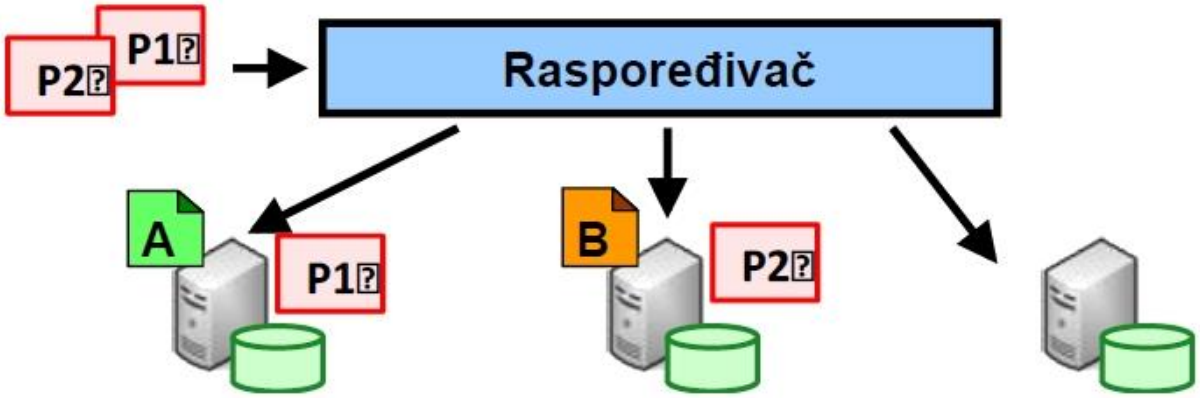
Sloj komunikacijskih protokola definira protokole komunikacije i atutentikacije koji su potrebni za transakcije u spletu.

Sloj protokola za sredstva omogućuje korisniku interakciju s udaljenim resursima i uslugama. On definira protokole za sigurno pregovaranje, pokretanje, praćenje, kontrolu, obračun (engl. accounting) i naplatu dijeljenih operacija i individualnih resursa.

Sloj zajedničkih usluga definira protokole i usluge koji su zaduženi za upravljanje grupom sredstava, a ne za pojedino sredstvo.

Na vrhu se nalazi **sloj korisničkih aplikacija**.

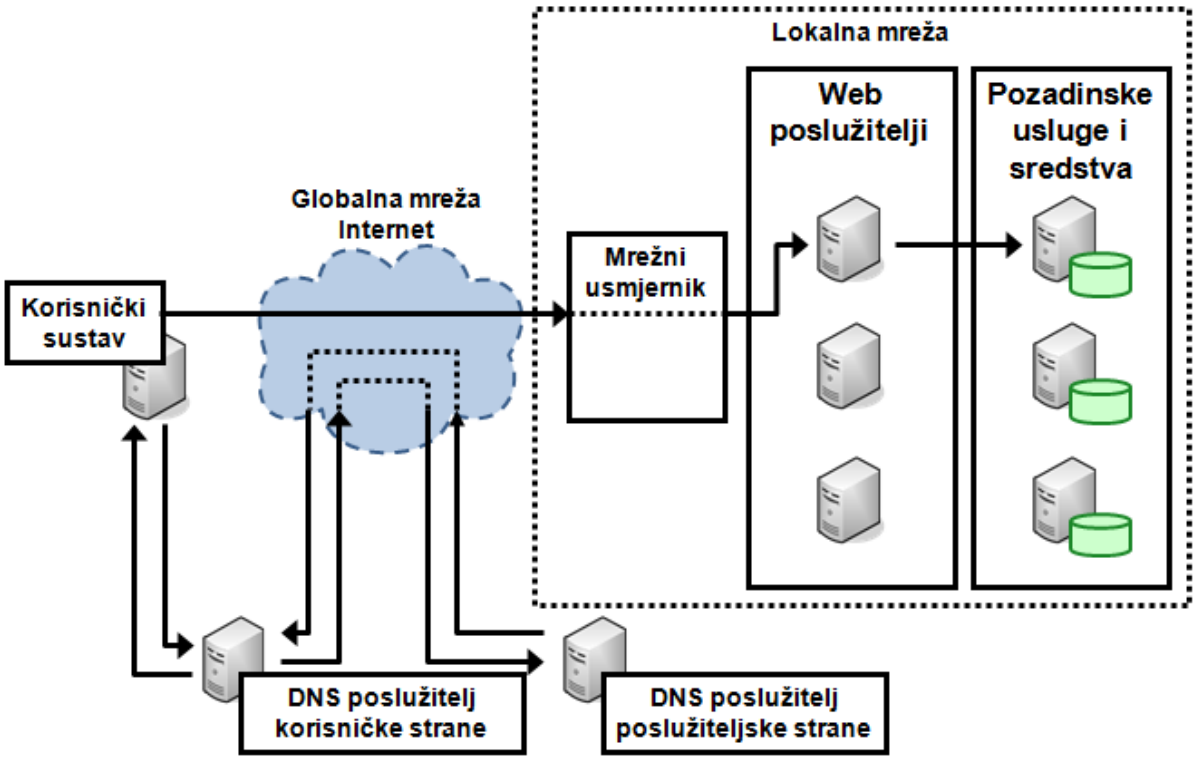
12.2. Na primjeru opišite značajke raspoređivanja zasnovanog na korištenju prostorne lokalnosti.



Slika 12.7. Primjer prostorne lokalnosti

-Kod prostorne lokalnosti, poslovi se raspoređuju na čvorove koji sadrže podatke potrebne za izvođenje posla. Drugim riječima, poslovi se približavaju podacima. Na primjeru sa slike, proces P1 koristi podatke A. Proces P1 se šalje na obradu na čvor na kojem su smješteni podaci A, a proces P2 se šalje na čvor na kojem su smješteni podaci B.

12.3. Prikažite i opišite elemente modela grozda računala.



Korisnički sustav je aplikacija kojom korisnik ostvaruje pristup i koristi sredstva i usluge na grozdu računala.

DNS poslužitelj korisničke strane je poslužitelj pomoću kojeg korisnički sustav razlučuje adrese udaljenih računala na Internetu.

DNS poslužitelj poslužiteljske strane je poslužitelj koji razlučuje adrese poslužitelja u lokalnoj mreži.

Mrežni usmjernik je uređaj koji prihvaća, analizira i usmjerava pristigle zahtjeve.

Web poslužitelji i pozadinska sredstva i usluge su osnovni elementi grozda računala.