# Otvoreno računarstvo

# 7. Raspodijeljeni sustavi

7.1 Uvod

7.2 Modeli

7.3 Međuprocesna komunikacija

7.4 Vanjske reprezentacija podataka

7.5 Izravna komunikacija

7.6 Neizravna komunikacija

# Creative Commons



**Attribution-NonCommercial-ShareAlike 4.0 International** (CC BY-NC-SA 4.0)
This license requires that reusers give credit to the creator.
It allows reusers to distribute, remix, adapt, and build upon the material in any medium or format, for noncommercial purposes only.
If others modify or adapt the material, they must license the modified material under identical terms.

**BY:** Credit must be given to you, the creator.
**NC:** Only noncommercial use of your work is permitted.
**SA:** Adaptations must be shared under the same terms.

# Otvoreno računarstvo

# 7. Raspodijeljeni sustavi

7.1 Uvod

7.2 Modeli

7.3 Međuprocesna komunikacija

7.4 Vanjske reprezentacija podataka

7.5 Izravna komunikacija

7.6 Neizravna komunikacija

# Smjerovi razvoja računarstva

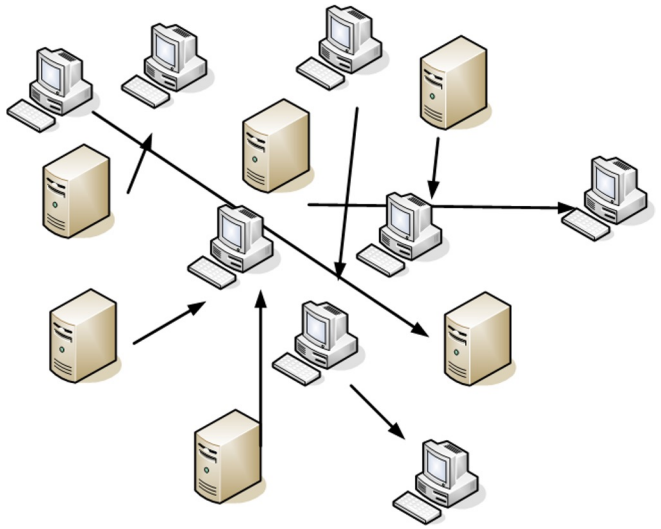
- Sveprisutnost
  - Komunikacijskih mreža (4G, 5G, WiFi, PAN, LPWAN, vozila, …)
  - Fizičko prisustvo (pametni prostori, pametni gradovi …)
  - Izvori energije
- Pokretljivost
  - Temeljena na infrastrukturi (pristupne točke, bazne stanice …)
  - Ad-hoc umrežavanje
  - Izvori energije
- Usluge
  - Lokalne, u oblaku, u magli, na rubu, u rosi …
- Računanje i komunikacija kao *komunalija*
  - struja, voda, grijanje*, garbage collection*

# Raspodijeljeni sustavi

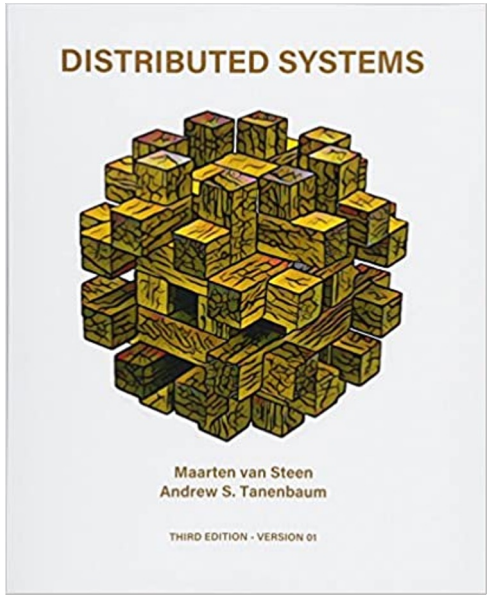
- Svaki iole složeniji računalni sustav/usluga sastoji se od više udaljenih, međusobno komunikacijski povezanih komponenata

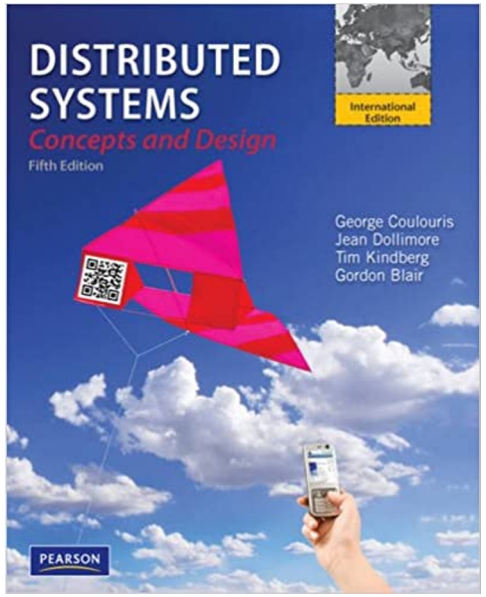
- "*Sustavi u kojem programske i sklopovske komponente umreženih računala komuniciraju i međusobno usklađuju aktivnosti* ***isključivo razmjenom poruka"*** (Coulouris i dr., 2012)

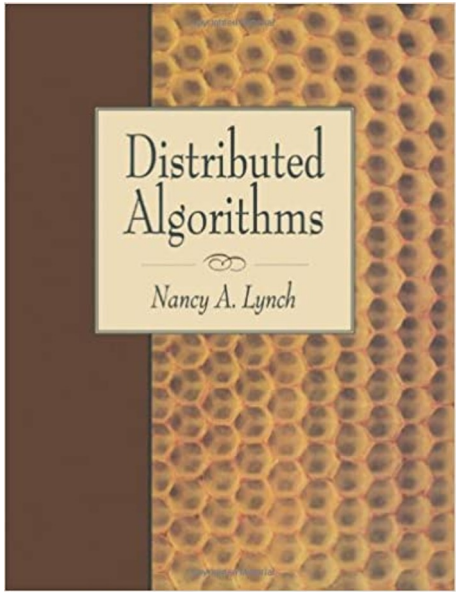
- *Sustav u kojem kvar računala za koje uopće ne znate da postoji može vaše računalo učiniti neupotrebljivim* (L. Lamport)



# Preporučena literatura :)



# Motivacija uvođenja raspodijeljenih sustava

- Zašto je osmišljena toljaga?
- Zašto je osmišljen brod?
  - Olakšati pristup udaljenim (tuđim) resursima, optimalno ih koristiti (za vlastitu korist)

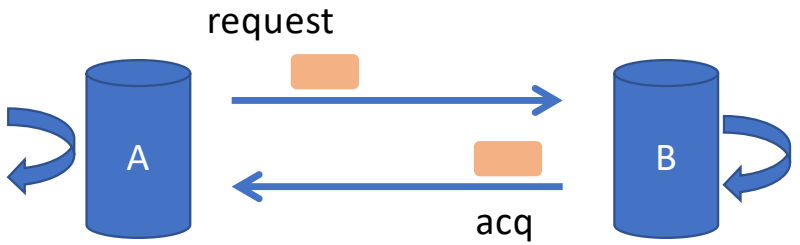
- Što su u našem slučaju resursi?
  - Sklopovski: printer, diskovni prostor …
  - Programski: datotečni sustav, obrada podataka …
  - Podatkovni: dokumenti, dijeljeni prostor, baze podataka, izvori podataka (IoT), video …
  - Usluge: infrastruktura (iaas), platforma (paas), programi (saas), … XaaS

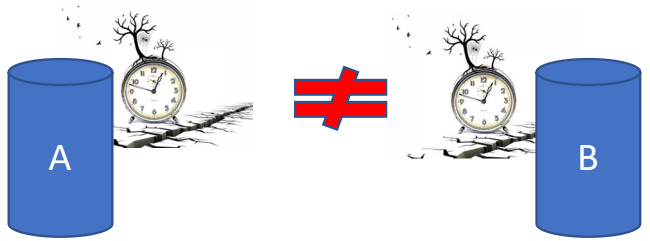
# Nekoliko pitanja vezanih uz udaljene resurse

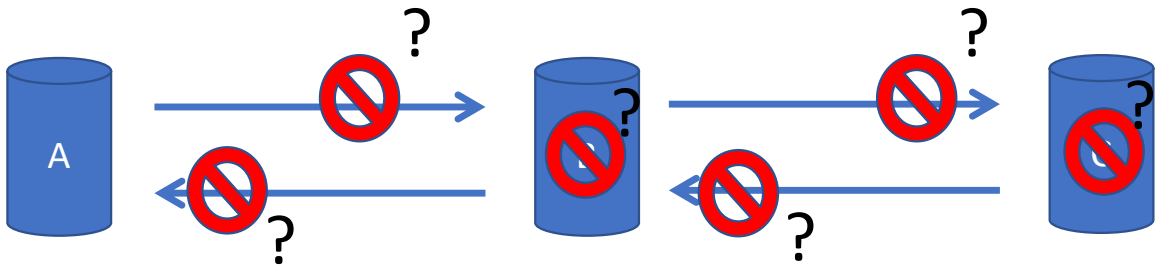
- Kako pristupiti udaljenom resursu?
  - Identitet, lokacija, metoda, jezik komunikacije …
  - URI, sučelje, komunikacijski protokoli, aplikacijski protokoli, *normiranost*…
- Što ako je moguć istodobni pristup više korisnika nekom resursu?
  - Štićenje integriteta resursa
  - Zaključavanje datoteke, raspodijeljene transakcije, semafori, procesni modeli
- Kako upravljati dozvolama za pristup/promjeni resursa?
  - Moguće akcije, dozvoljene akcije, propagiranje prava, (privremeno) prenošenje prava …
  - autentikacija, autorizacija, sigurnosni mehanizmi (šifriranje, potpisivanje), *norme* …

# Osnovne karakteristike raspodijeljenih rač. sustava

- Osnovne karakteristike raspodijeljenih računalnih sustava:
  - **paralelizam** izvođenja elemenata sustava
    - nužna koordinacija
  - **nepostojanje globalnog sata**
    - koordinacija isključivo porukama
  - **neovisnost grešaka**
    - mjesto greške, detekcija, oporavak



# Izazovi raspodijeljenih sustava

1. heterogenost
2. otvorenost
3. sigurnost
4. skalabilnost
5. pogreške u radu
6. paralelizam
7. transparentnost
8. kvaliteta usluga

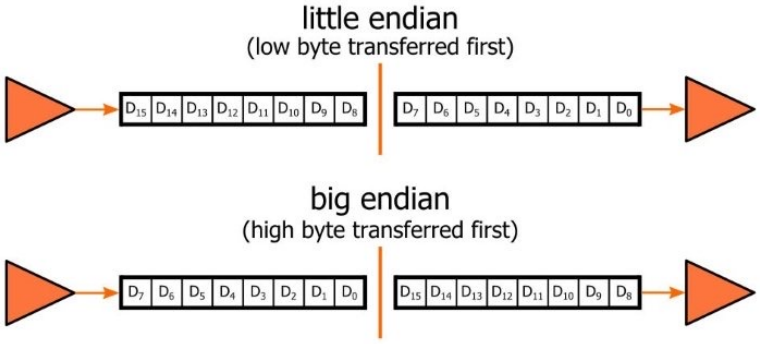
# Heterogenost sustava (I)

- **Heterogenost** na razinama:
  1. Komunikacijskih mreža
     - Različite vrste mreža kojima se prenose podaci (bakar, optika, bežično …)
  2. Računalnog sklopovlja
     - arhitektura, raspoloživi procesni, memorijski, energetski i komunikacijski resursi
  3. Operacijskih sustava
     - Različiti operacijski sustavi, postojanje/nepostojanje operacijskog sustava
  4. Programskih jezika korištenih u izgradnji aplikacija
     - Reprezentacija/postojanje tipova podataka, različite semantika
  5. Implementacija programskih komponenti sustava
     - Pridržavanje normi, različita semantika

Platforma = sklopovlje + operacijski sustav

# Heterogenost sustava (II)

- **Primjeri izazova uzrokovanih heterogenošću**
  - Poredak zapisa okteta (*big/little endian*, *network byte order*)
  - Zapis struktura podataka (JavaScript, serijalizacija...)
  - Kodiranje znakova (*ASCII*, *Unicode*...)
  - Razlike u jezicima implementacije
    - kako se terminira niz znakova?
    - dijeljene složene strukture podataka
    - …




https://www.allaboutcircuits.com/technical-articles/big-endian-little-endian-endianness-byte-arrangement-digital-systems/

  - Raspoloživa sučelja za mrežnu komunikaciju (*Berkeley socket API*, *Windows, MPI* ...)

# Heterogenost sustava (III)

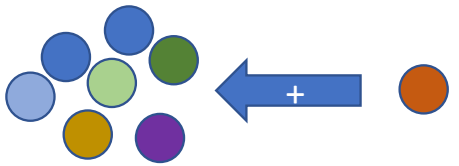
- Metode skrivanja heterogenosti
  - Komunikacijski protokoli
    - Skrivanje različitosti prijenosnog medija
  - *Middleware (posrednički sloj)*
    - *Skrivanje različitosti*
      - *jezika implementacije komponenti aplikacijskog sloja,*
      - platformi,
      - komunikacijskih protokola

**Postelov zakon** (princip robusnosti):

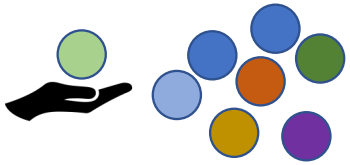
***"Budi konzervativan u onome što činiš** (šalješ),*
***budi liberalan u onome što prihvaćaš od drugih."***

# Otvorenost raspodijeljenih sustava

- **Otvorenost** raspodijeljenih sustava određena
  - mogućnošću **dodavanja novih** usluga
  - njihovim **jednostavnim korištenjem**



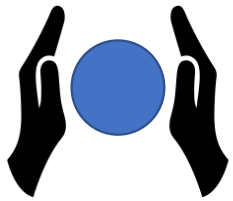
- Temelji:
  - **normirani** komunikacijski **protokoli**
  - javno objavljena ili **normirana sučelja**
    za pristup dijeljenim resursima (API)



- Otvoreni raspodijeljeni sustavi su građeni od **heterogenih** elemenata, ali svaki od tih elemenata mora biti sukladan korištenim normama

# Sigurnost

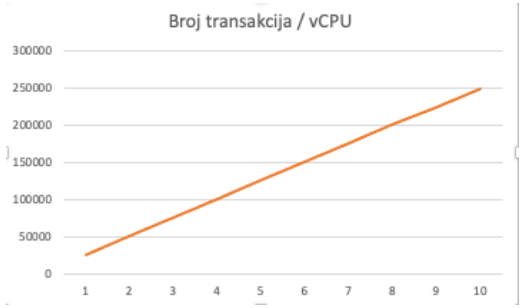
- Povjerljivost
  - čuvanje resursa od neovlaštenog pristupa
- Integritet
  - čuvanje od promjena ili uništavanja
- Raspoloživost
  - čuvanje od zapreka pristupa resursima



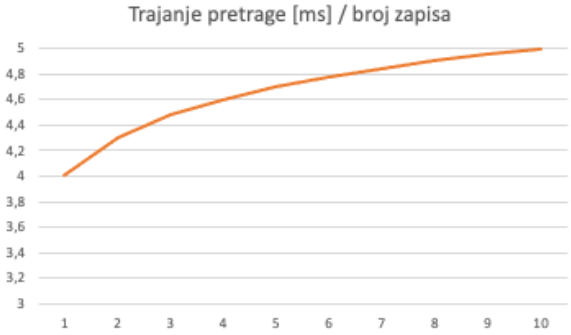
# Skalabilnost

**Učinkovitost** sustava bez obzira na **značajno povećanje posluživanih resursa i/ili broja posluživanja**

- Skalabilnost s obzirom na **cijenu posluživanja**
  - nadogradnja opreme, zbog povećanja zahtjeva za resursima
  - povećanje cijene posluživanja resursa, *prihvatljivo* - ***O(n)*** [n - broj zahtjeva]




- Skalabilnost s obzirom na **performanse**
  - smanjenje performansi posluživanja uz istu raspoloživu opremu i povećanje resursa, *prihvatljivo* - ***O(log n)*** [n – veličina resursa]




- Ograničenost resursa (konačna)
  - primjer IPv4 adrese

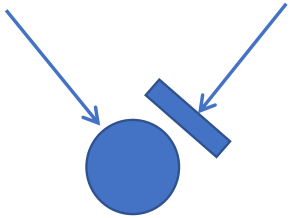
- Izbjegavanje uvođenja uskih grla sustava
  - raspodijeljena umjesto centraliziranih rješenja

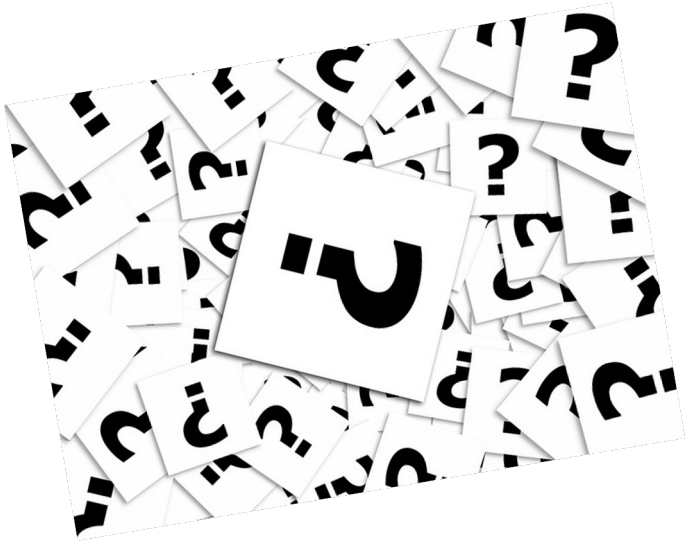
# Pogreške u radu

- Pogreške u raspodijeljenom sustavu su **djelomične**
  - dio komponenata nastavlja obavljati svoju funkciju
  - Komponente uglavnom nisu svjesne grešaka drugih dijelova sustava
- Najčešće vrste pogrešaka
  - u **procesima** (komponentama) sustava
  - u mrežnoj **komunikaciji**
- Problematika pogrešaka u raspodijeljenom sustavu
  - detekcija pogreške (mjesto, uzrok)
  - posljedice pogreške, poznavanje ukupnog stanja sustava?
  - toleriranje pogrešaka, oporavak od pogreške
  - semantika izvođenja operacija uz mehanizme oporavka od pogrešaka
    - Operacija se smije izvesti samo jednom ili se može izvesti više puta uz isti učinak?
    - Operacija je izvedena *možda jednom*, *točno jednom*, *barem jednom* … ?

# Paralelizam

- Komponente raspodijeljenog sustava izvršavaju se **paralelno**
  - zahtjevi za resursom mogu prispjeti od **više klijenata** istovremeno
    - Rezultat izvršnog modela primijenjenog na poslužitelju
  - problem usklađivanja pristupa resursu – **sinkronizacija** akcija





# Transparentnost (I)

- Prikaz sustava kao cjeline umjesto kao skupa raznih samostalnih komponenata
- Transparentnost **pristupa**
  - **jednak skup operacija** za pristup lokalnim i udaljenim resursima
- Transparentnost **lokacije**
  - skrivena prava lokacija resursa (simboličko ime naprema IP adresi poslužitelja)

**Transparentnost mreže**

- Transparentnost **paralelizma**
  - očuvanje konzistencije dijeljenog resursa
- Transparentnost **replikacije** resursa

# Transparentnost (II)

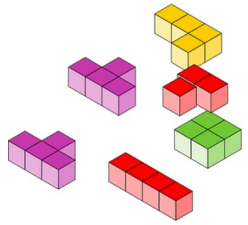
- Transparentnost **pogrešaka**
  - skrivanje postupka oporavka
- Transparentnost **mobilnosti**
  - promjena lokacije resursa ne utječe na način pristupa resursu
- Transparentnost **performansi**
  - prilagodba novom stanju (npr. broju korisnika) zbog očuvanja performansi sustava
- Transparentnost **skalabilnosti**
  - proširenje sustava moguće bez utjecaja na strukturu sustava, postojeće aplikacije i korištene algoritme

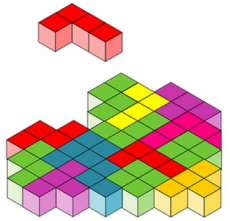
# Otvoreno računarstvo

# 7. Raspodijeljeni sustavi

# Modeli raspodijeljenih sustava

- **Model raspodijeljenog sustava** – apstraktan, pojednostavljen i konzistentan opis nekog aspekta dizajna raspodijeljenog sustava
  - **Fizički model** – fizički uređaji i komunikacijski kanali kojima su uređaji spojeni
  - **Arhitekturni model** – struktura sustava; gradivne komponente sustava (entiteti), njihove uloge u sustavu i suodnosi između komponenata

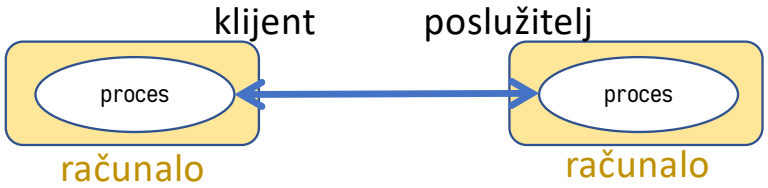


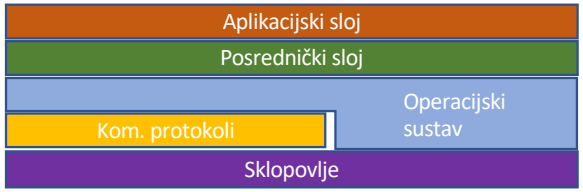


- Ključna pitanja na koje arhitekturni model mora odgovoriti:
  **1)** Koji **entiteti** čine konkretni raspodijeljeni sustav?
  **2)** Koju **komunikacijsku paradigmu** koriste entiteti da bi međusobno komunicirali?
  **3)** Koje su **uloge** i odgovornost entiteta u konkretnom raspodijeljenom sustavu?
  **4)** Kako se entiteti arhitekturnog modela **mapiraju** na komponente fizičkog modela sustava ?

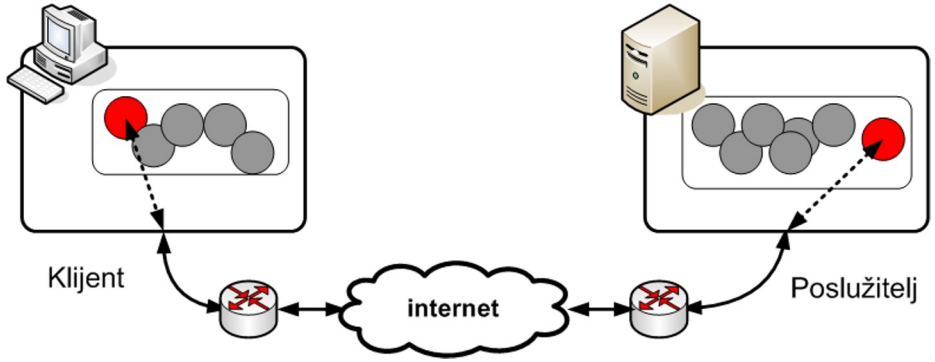
# Entiteti raspodijeljenog sustava

- Perspektiva (operacijskog) sustava:
  - procesi
  - dretve
  - *firmware* (ugradbeni uređaji, IoT)



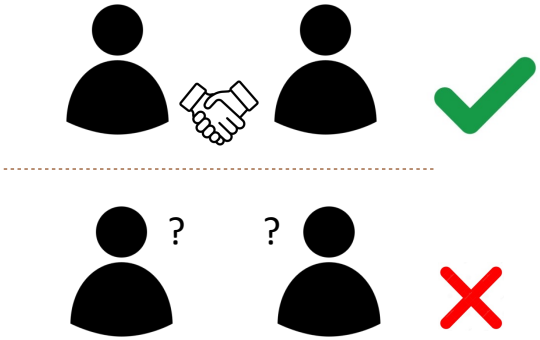




- Perspektiva aplikacijskog sloja:
  - objekti
  - komponente
  - web usluge

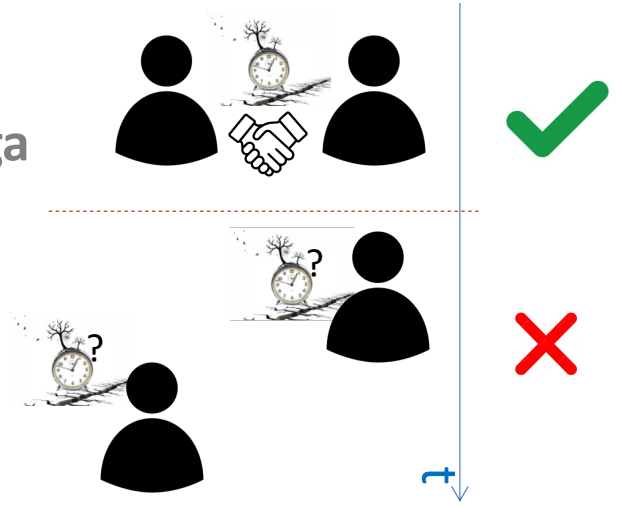


# Prostorna i vremenska sprega

- Prostorna sprega entiteta
  - Entitet korisnik svjestan entiteta pružatelja usluge, najčešće i entitet pružatelj usluge svjestan trenutnog entiteta korisnika – ***prostorna sprega*** *(engl. space coupling*)



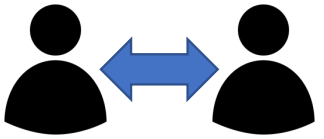

- Vremenska sprega entiteta
  - Oba entiteta aktivna tijekom komunikacije – **vremenska sprega** (engl. *time coupling*)



# Komunikacijske paradigme (I)

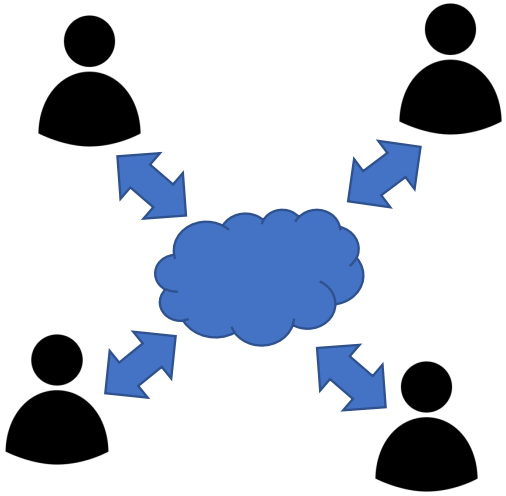
**1) Izravna** komunikacija

- jedan entitet korisnik i jedan entitet pružatelj usluge
  - odnos 1:1
- Nužno postojanje i ***prostorne i vremenske sprege*** komunicirajućih entiteta



**2) Neizravna** komunikacija

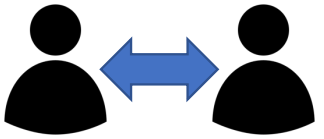
- **nije potrebna sprega** entiteta pružatelja usluge i (*jednog ili **više***) entiteta korisnika
  - ***ne mora postojati ni prostorna ni vremenska sprega***
- Najčešće korištenje trećeg - **posredničkog** - entiteta za razdvajanje entiteta korisnika i entiteta pružatelja usluge
  - odnosi 1:n, n:1, n:m



# Komunikacijske paradigme (II)

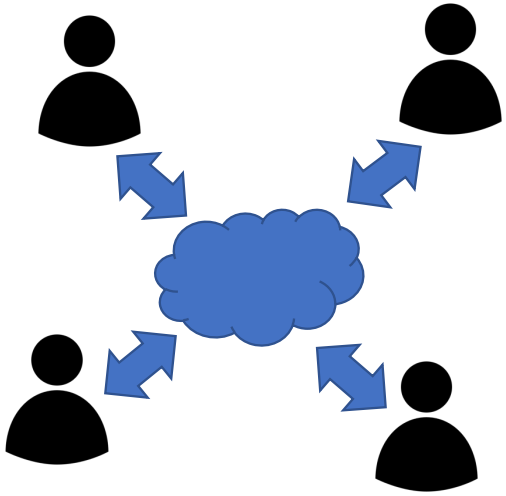
**1) Izravna** komunikacija

- Međuprocesna komunikacija
- Udaljeni pozivi



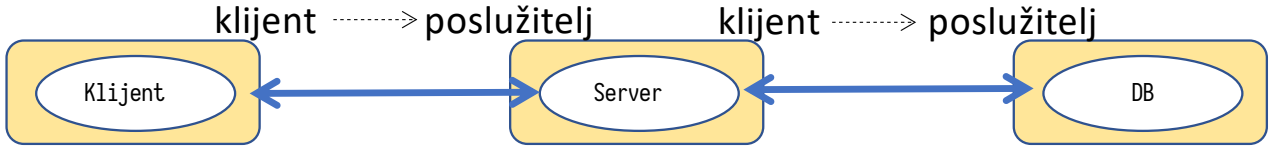
**2) Neizravna** komunikacija

- Grupna komunikacija
- Objavi – pretplati
- Redovi poruka
- Raspodijeljena dijeljena memorija
- Prostori podataka



# Uloge i odgovornosti entiteta (I)

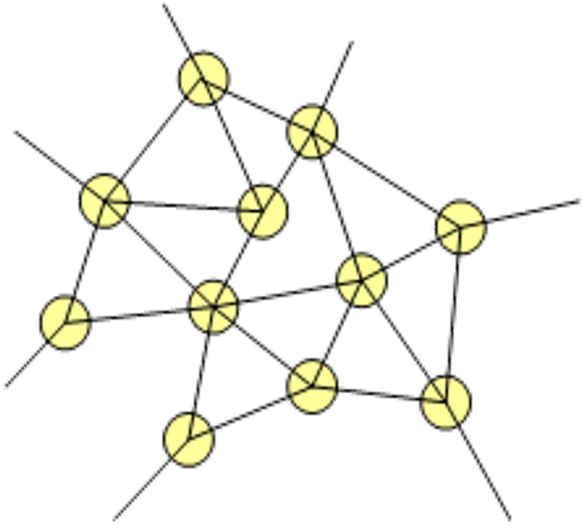
- **Klijent – poslužitelj** arhitekturni stil (engl. client – server)
  - klijent zahtijeva uslugu
  - poslužitelj mora isporučiti uslugu




- Podjela uloga klijenata i poslužitelja u složenijem sustavu ne mora biti jasna
  - jedan entitet može imati **obje uloge**
  - npr. program poslužitelj weba je:
    - poslužitelj u odnosu na preglednik weba od kojeg prihvaća zahtjev za web stranicom
    - klijent u odnosu na bazu podataka iz koje dohvaća podatke za dinamički generiranu html stranicu

# Uloge i odgovornosti entiteta (II)

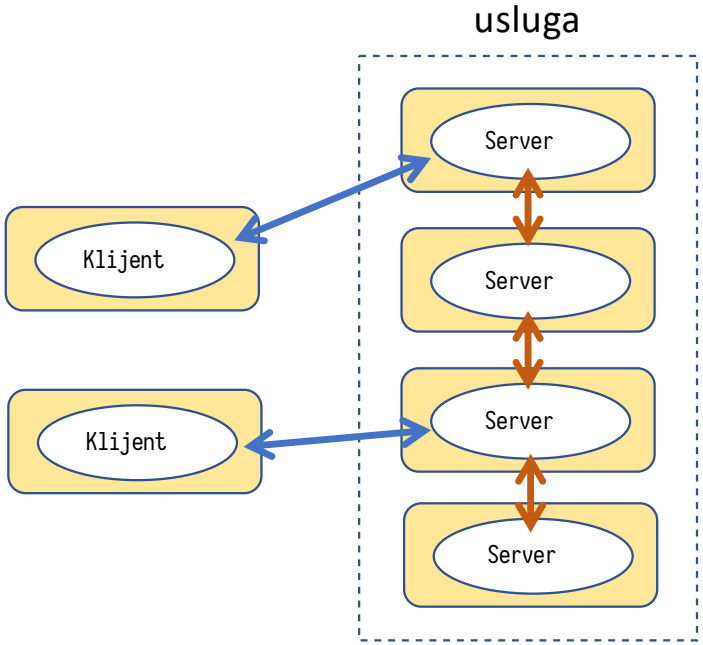
- Svi entiteti ravnopravni (engl. *peer-to-peer*)
  - isti program, iste uloge, ista sučelja
  - Dio tereta **prebačen** s pružatelji usluge **na korisnike** usluge (svi su istovremeno i korisnici i pružatelji usluge)
  - Skaliranje resursa usluge sukladno **trenutnom broju korisnika**
  - Veze između entiteta sukladno potrebama aplikacije
  - Raspodjela opterećenja po aktivnim članovima (mrežni promet, podaci, obrada)
  - Redundancija podataka i funkcionalnosti



# Mapiranje entiteta na fizičke čvorove (I)

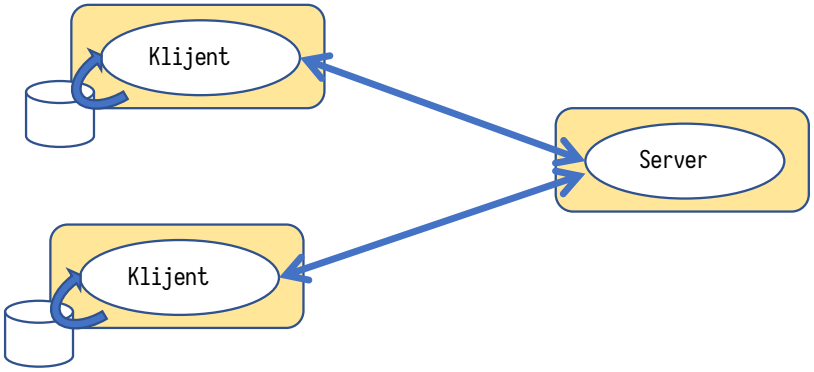
- Logički entiteti mapiraju se na stvarne izvedbene resurse

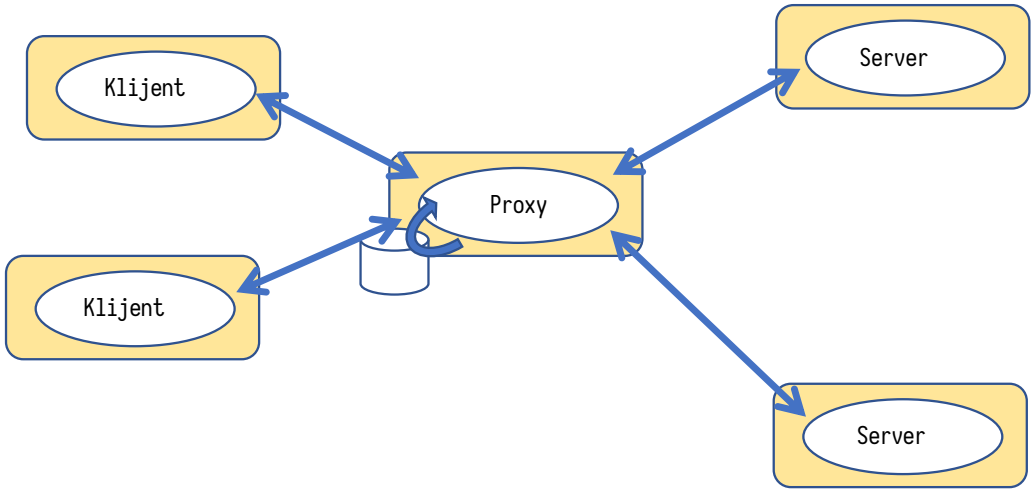
1. Višestruka računala poslužitelji
   a) **particioniranje** komponenti usluge i dijelova resursa na više računala poslužitelja
   b) **repliciranje** jedinstvene usluge i resursa na više računala poslužitelja



# Mapiranje entiteta na fizičke čvorove (II)

2. Priručne memorije (engl. cache)
   a) na strani klijenta (npr. preglednik weba)
   b) na posrednicima (engl. proxies)



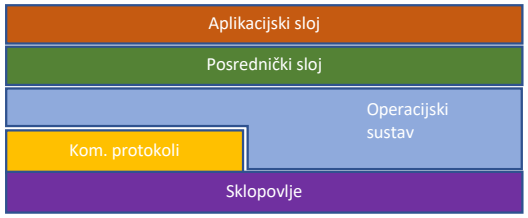
# Mapiranje entiteta na fizičke čvorove (III)

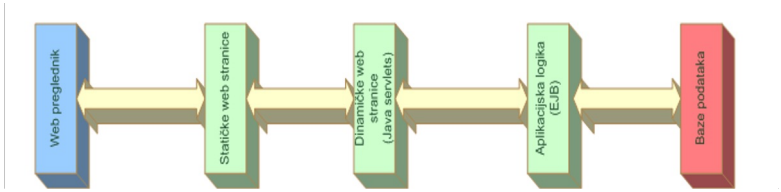
3. Pokretni programi
   - dobavljeni kôd proširuje funkcionalnost klijenta
   - lokalna ili udaljena interakcija
   - problem sigurnosti
   - Java applets – dobavljanje izvršnog kôda na stranu klijenta
   - JavaScript – dobavljanje kôda u preglednik Weba

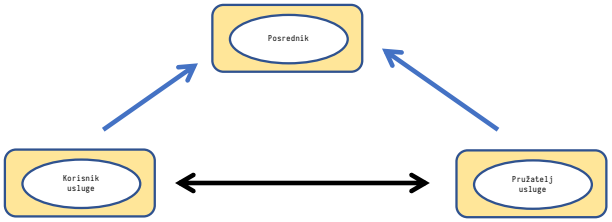
4. Pokretni agenti
   - prenošenje i kôda i stanja
   - samostalno kretanje između grupe računala i lokalno obavljanje zadataka
   - smanjenje korištenja mreže, ubrzavanje operacija

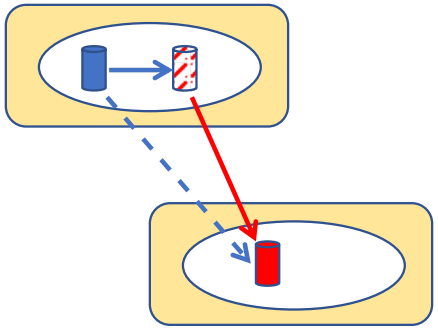
# Arhitekturni obrasci u raspodijeljenim sustavima

- Slojevi - vodoravni (engl. *layers*)
- Slojevi – vertikalni (engl. *tiers*)
- Zastupnici (engl. *proxies*)
- Posrednici (engl. *brokers*)



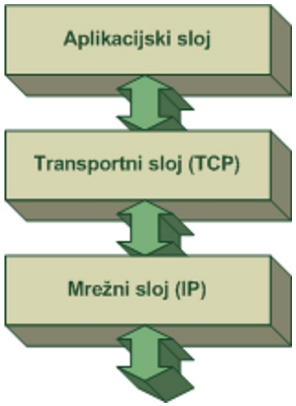



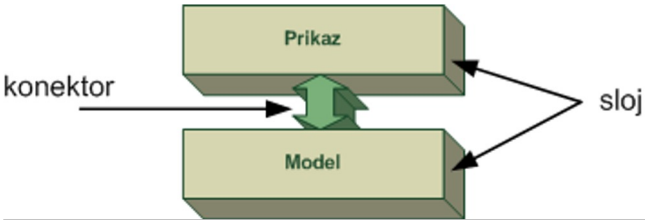



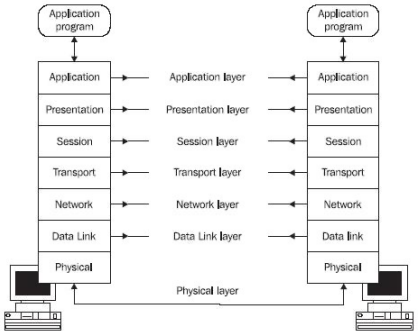
# Slojevita arhitektura

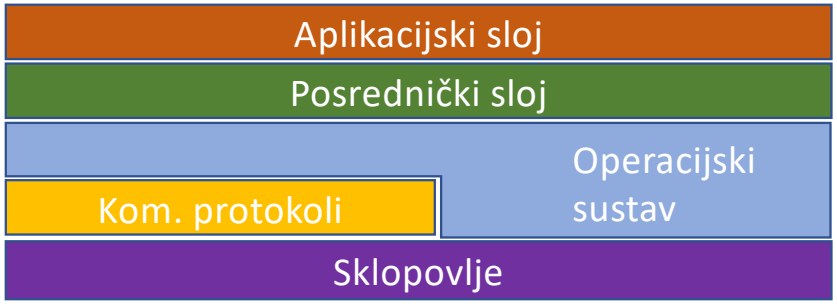
- Elementi slojevitog modela:
  - slojevi
    - razine apstrakcije, rješavaju neovisne zadatke
  - konektori
    - protokoli interakcije između susjednih slojeva




- Hijerarhijska organizacija slojeva
  - interakcija samo između susjednih slojeva
  - udaljeni slojevi “skriveni”



# Svojstva slojevite arhitekture

- **Prednosti** slojevite arhitekture:
  - sloj obavlja točno određenu ulogu
  - slojevi “slabo” povezani konektorima
  - neovisnost o implementaciji, slojevi jednostavno zamjenjivi
  - protokoli interakcije se moraju strogo poštovati

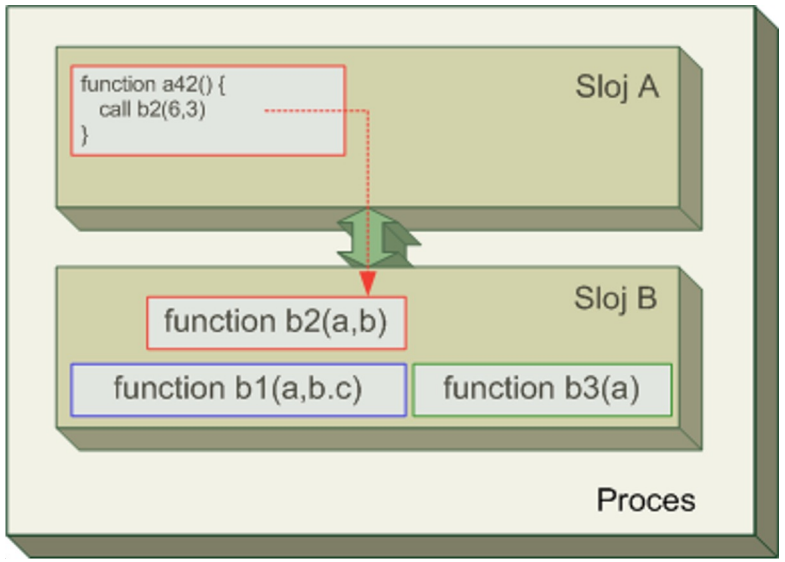
Rainbow Colours Cheese Layered Cake

- **Nedostaci** slojevite arhitekture:
  - smanjena učinkovitost sustava
  - skupa promjena protokola interakcije / sučelja
  - ponekad teško identificirati jasno odijeljene slojeve

Coconut Milky Yam cheese Layered Cake

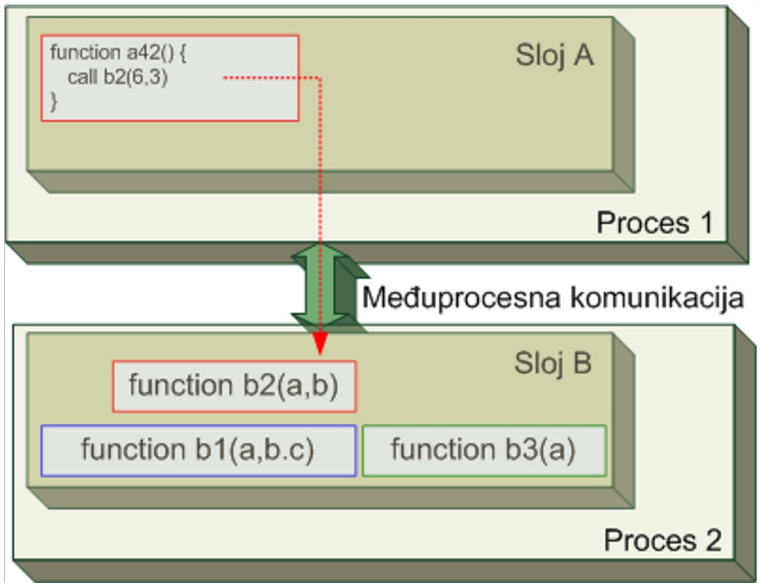
# Izvedba slojeva i konektora (I)

- **Monolitna** aplikacija:
  - **jedan proces** izvođen u okviru operacijskog sustava jednog računala
  - slojevi
    - logički (i fizički?) odvojene biblioteke funkcija
    - čine jedinstveni izvedbeni kôd aplikacije
  - konektori između slojeva
    - skup funkcija vidljivih iz susjednog sloja
    - komunikacija pozivima funkcija susjednog sloja



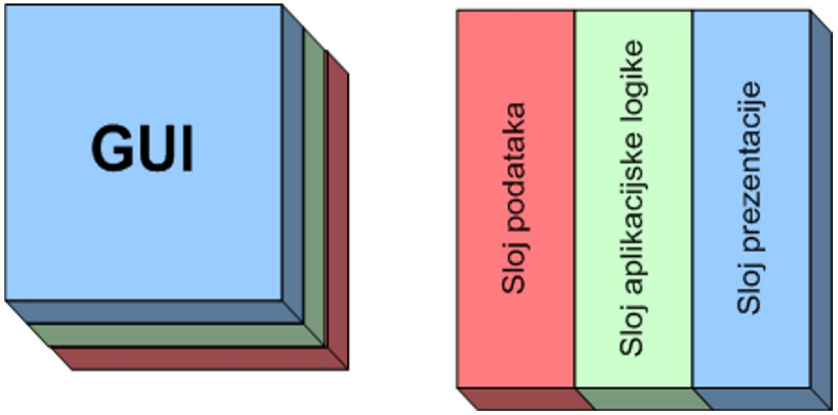
# Izvedba slojeva i konektora (II)

- **Višeprocesna** aplikacija:
  - dva ili više procesa izvođenih na jednom ili više računala
  - izvođenje **na više računala** - raspodijeljena aplikacija
  - (neki ili svi) slojevi izolirani unutar zasebnih procesa
  - konektori između slojeva
    - mehanizmi međuprocesne komunikacije
    - komunikacijski protokol



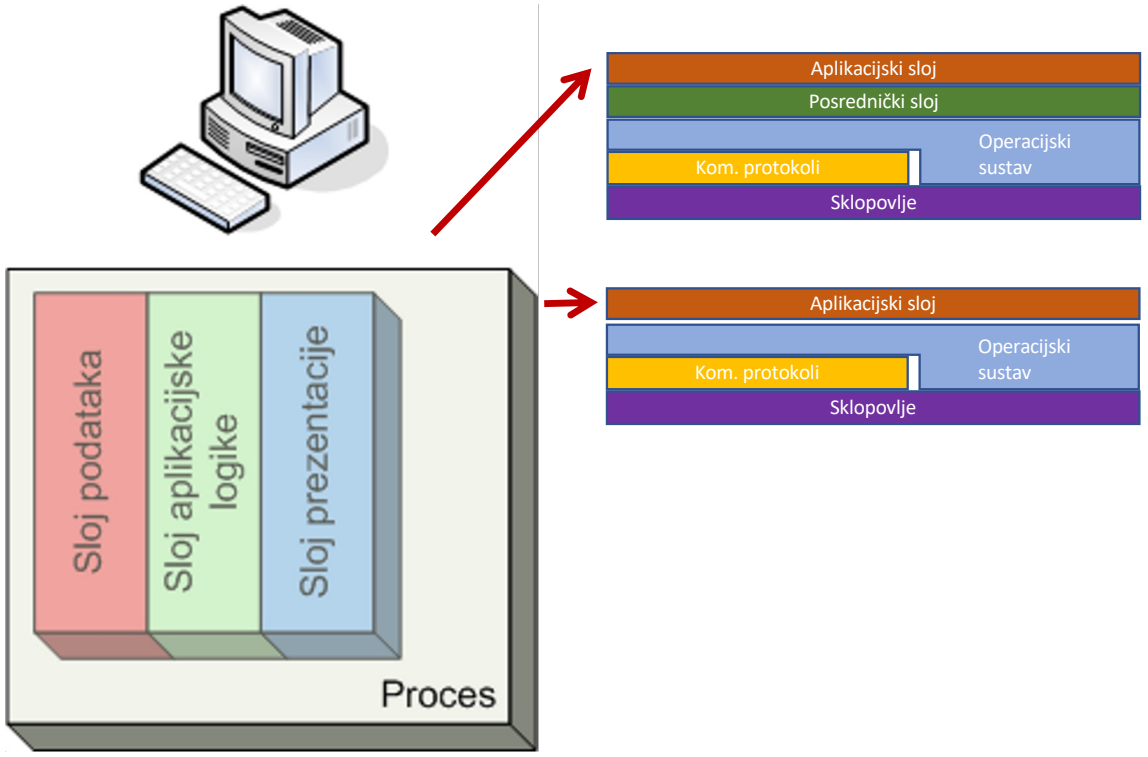
# Slojevi aplikacije

- Tipična arhitektura aplikacije sastoji se od tri sloja:
  - sloja **prezentacije** (GUI)
  - sloja aplikacijske **logike**
  - sloja **podataka**



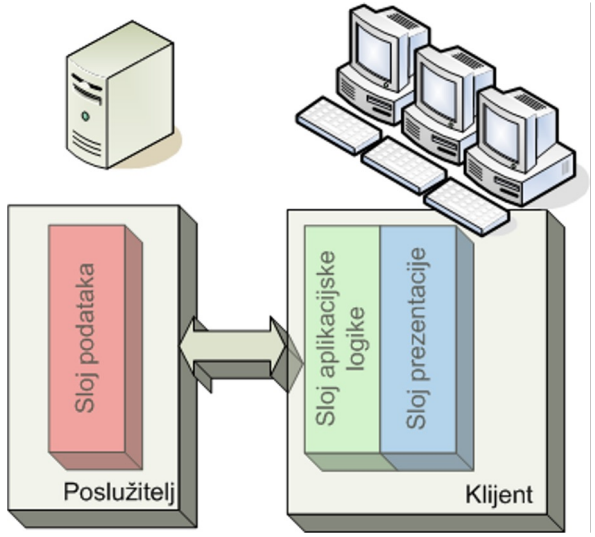
# Monolitna arhitektura

- Svi funkcionalni slojevi aplikacije unutar procesa izvođenog na jednom računalu
- Primjeri:
  - obrada teksta
  - tablični kalkulator
  - razvojne okoline
  - *single-player* igre ...
- Dodatni alati potrebni za omogućavanje grupnog rada
  - npr. *subversion* (svn, cvs ...)



# Dvoslojna arhitektura

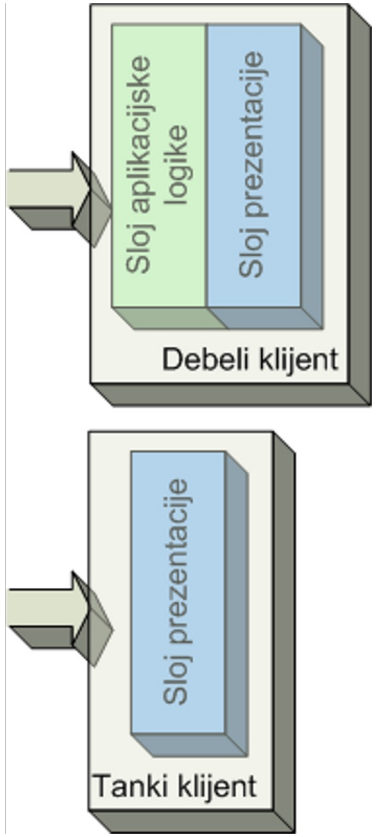
- Funkcionalni slojevi grupirani u dva zasebna arhitekturna sloja (*2-tier*), tj. procesa
- Na primjer:
  - klijentska aplikacija
    - sadrži funkcionalne slojeve prezentacije i aplikacijske logike (ako postoji)
  - poslužiteljska aplikacija
    - sadrži sloj podataka
    - može istovremeno pružati usluge jednoj ili više klijentskih aplikacija




- Ali...! Može i drugačije:

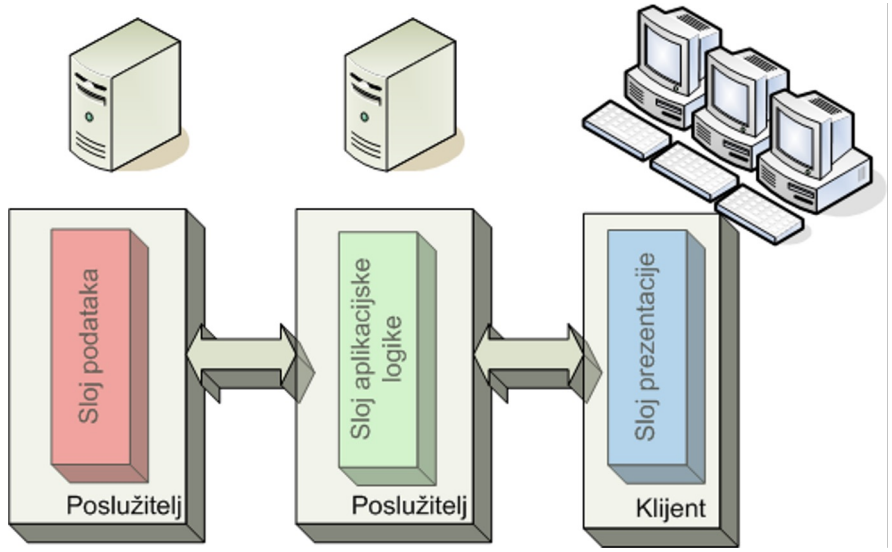
# Tanki i debeli klijent

- **Debeli** klijent (*fat client*):
  - sadrži slojeve prezentacije i aplikacijske logike
  - zahtijeva veću snagu obrade računala domaćina i veću količinu podataka prenošenih mrežom
- **Tanki** klijent (*thin client*):
  - sadrži samo sloj prezentacije
  - manja snaga obrade, manja količina prenošenih podataka



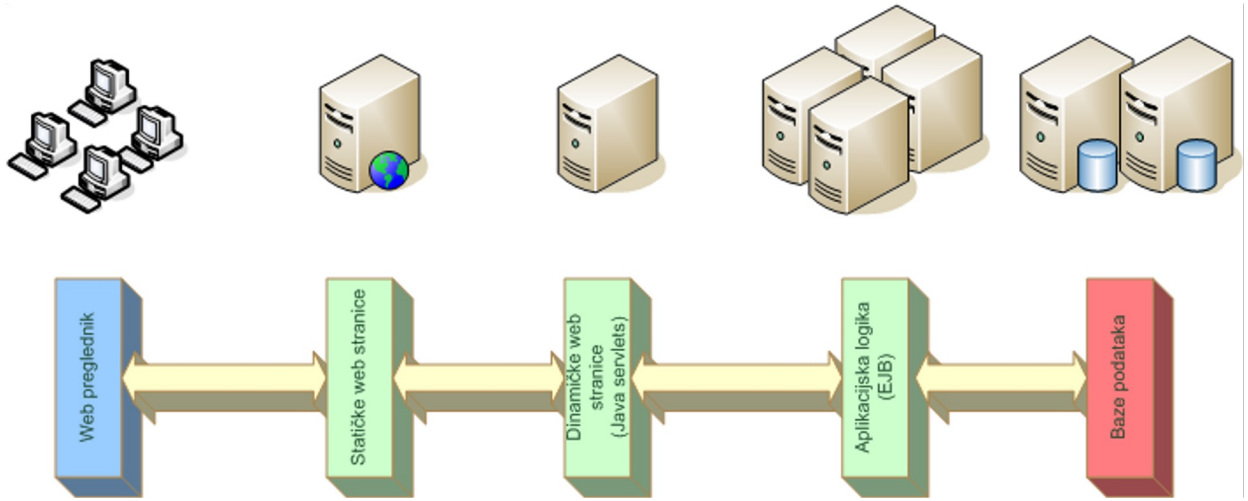
# Troslojna arhitektura

- Funkcionalni slojevi grupirani u tri zasebna arhitekturalna sloja (*3-tier*), tj. procesa
  - sloj prezentacije – klijentska aplikacija
  - srednji sloj - sloj aplikacijske logike
  - sloj podataka - baza podataka...



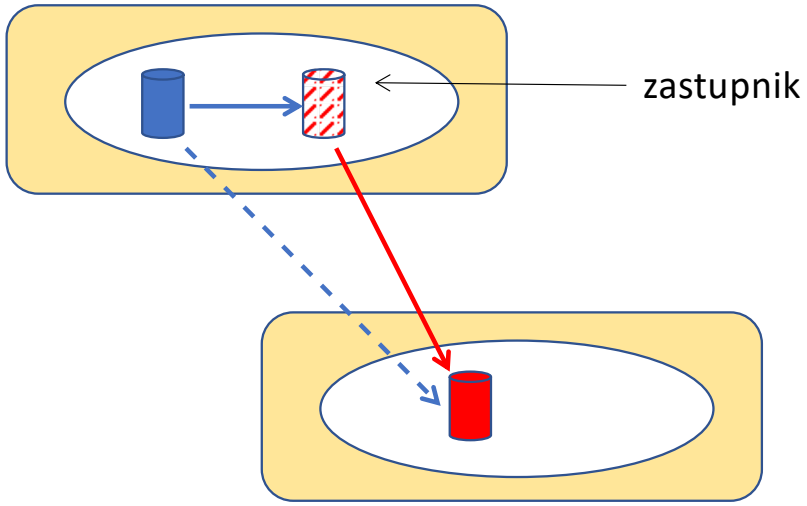

# Višeslojna arhitektura

- Višeslojna arhitektura (*multi-tier, n-tier*) sadrži višestruke (specijalizirane i/ili redundantne) aplikacijske poslužitelje i/ili baze podataka
  - ravnomjernija raspodjela opterećenja
  - potrebna veća propusnost komunikacijske infrastrukture



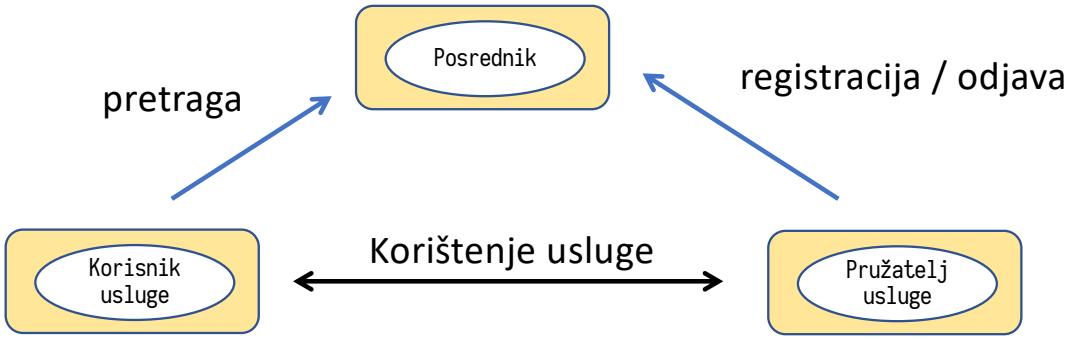
# Zastupnici

- Entitet *korisnik* želi komunicirati s *udaljenim* entitetom
  - udaljeni entitet se nalazi unutar odvojenog procesa na istom ili drugom računalu
- Entitet *korisnik* komunicira korištenjem lokalnog *zastupnika* udaljenog objekta
  - komunikacija *korisnika* i *zastupnika* je lokalna – oboje su unutar istog procesa
  - posredna komunikacija s *udaljenim* entitetom – *zastupnik* preuzima stvarnu komunikaciju
  - kompleksnost mrežne komunikacije je skrivena od entiteta *korisnika* – *mrežna transparentnost*



# Posrednici

- Posrednik sadrži popis usluga i njihovih svojstava
- Pružatelji usluga oglašavaju usluge kod posrednika
- Potencijalni korisnici usluge traže usluge odgovarajućih svojstava kod posrednika



# Modeli slanja/primanja podataka

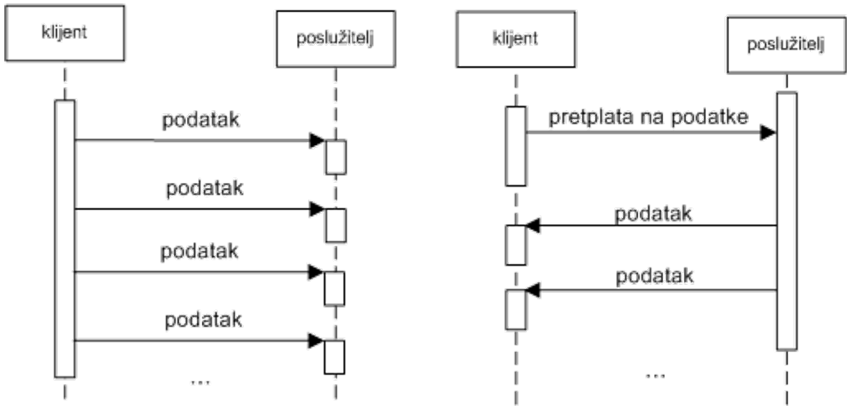
- povlačenje podataka (*pull*)
- guranje podataka (*push*)
- prozivanje (*polling*)

# Povlačenje podataka

- **Povlačenje** informacije (*pull*):
  - zahtjev za podacima entitetu koji ih posjeduje
  - najčešće klijent zahtijeva podatke od poslužitelja
  - tipičan primjer: *request-response* protokoli (HTTP)
  - priroda veze: većinom povremena (trajanje dohvata)

# Guranje podataka

- **Guranje** informacije (*push*):
  - guranje podataka od stane klijenta
    - npr. slanje i prosljeđivanje e-pošte, messenger-i, p2p ...
    - veza većinom privremena
  - guranje podataka od strane poslužitelja
    - npr. FTP, HTTP push ...
    - veza je trajnija (problem kod velikog broja klijenata!)



# Prozivanje

- **Simuliranje guranja** podataka **prozivanjem** (*polling*)
  - klijent **periodički** uspostavlja vezu s poslužiteljem i provjerava dostupnost podataka
  - podaci se dohvaćaju povlačenjem
  - nema trajne veze kao kod “čistog” guranja
  - značajno opterećenje mreže i poslužitelja kod velikog broja klijenata
  - primjeri:
    - guranje e-pošte na klijenta (POP, IMAP)
    - *RSS feeds*

# Arhitektura usluga klijent-poslužitelj

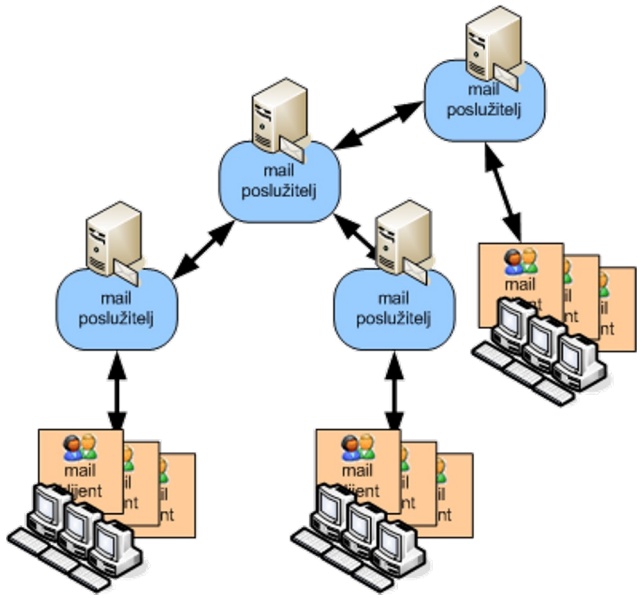
- WWW, DNS, LDAP, SVN – informacijske usluge
  - WWW
    - jasni slojevi prezentacije i podataka
  - DNS, LDAP
    - uloga klijenta ovisna o konkretnoj primjeni
  - SVN
    - poslužitelj kao sloj podataka, klijent (ni)je sloj prezentacije

- Telnet, SSH – rad na udaljenom računalu
  - klijent predstavlja prezentacijski sloj (konzolu), poslužitelj nije “skladište podataka”
  - što je u ovim slučajevima zahtjev, a što odgovor?

# Arhitektura usluge e-pošte

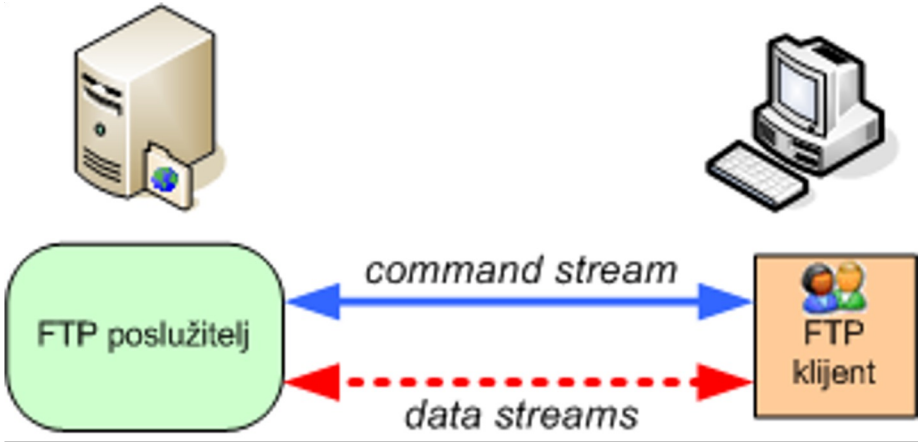
- *Mail* klijenti:
  - slanje pošte, dohvaćanje pošte iz poštanskog sandučića
  - prezentacijski, djelomično i podatkovni sloj (lokalna pohrana pošte)

- *Mail* poslužitelji:
  - podatkovni sloj prema klijentima
  - p2p organizacija prosljeđivanja (MTAx ↔ MTAy)



# Arhitektura usluge FTP

- Prijenos datoteka između dva računala
- Komunikacijski kanal između klijenta i poslužitelja za prijenos naredaba i odgovora (*command stream*)
- Za svaki prijenos datoteke otvara se poseban komunikacijski kanal (*data stream*)
  - aktivni mod: poslužitelj inicira vezu
  - pasivni mod: klijent inicira vezu



# X Window System

- Prikaz korisničkog sučelja
  - poslužitelj: grafički prikaz, window manager
  - klijenti: korisničke aplikacije (konzola, GUI)
  - klijenti šalju podatke X poslužitelju o izgledu sučelja (sadržaju prozora u kojem se sučelje aplikacije prikazuje)
  - poslužitelj prikazuje sadržaj prozora, akcije korisnika pretvara u događaje i šalje odgovarajućim klijentima