

# ՅՈՒՆԵՍԿՈ ԵՎ ՆԻԿՅԱՐԻԿԵԼԻԱՆ

ՆԻԿՅԱՐԻԿԵԼԻԱՆ ԵՎ ՅՈՒՆԵՍԿՈ  
ԳՅՈՒՄՆԻԿԱԿԱՆ - ՈՐԿԱՆԻԶԱՆ

ՈՒՆԵՍԿՈ ԵՎ ՆԻԿՅԱՐԻԿԵԼԻԱՆ

# Otvoreno računarstvo

- Raspodijeljeni računalni sustavi - pojmovi
  - Slojevita arhitektura kao pojam
  - Arhitekture raspodijeljenih aplikacija
  - Primjeri arhitektura mrežnih sustava
  - Klijenti i poslužitelji kao računala
  - Međuprocesna komunikacija u raspodijeljenim sustavima
  - Aplikacijski protokoli
  - Mehanizmi protokola
  - Stanja usluga
  - URI
  - MIME

Mario Žagar



# Predgovor

---



- Svaki uređaj mora imati dva priključka:
  - jedan za napajanje
  - jedan za komuniciranje

M.Ž. 1983.



- Motto vrijedi i danas!



Nema žica,  
nema baterija!

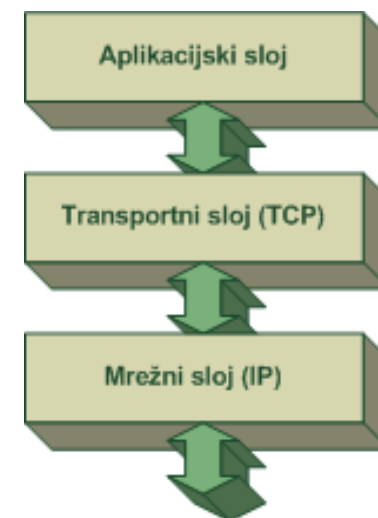
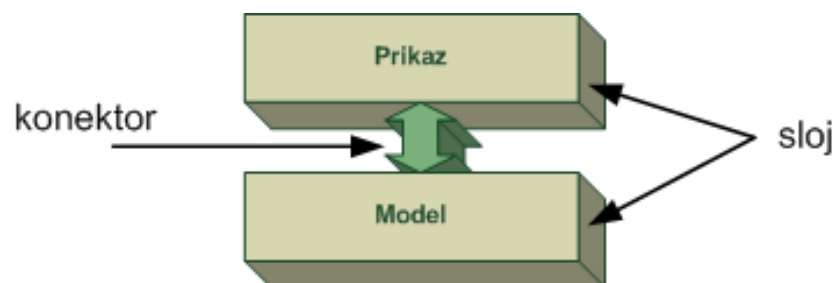


Slojevita arhitektura kao pojam

# 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
- Nedostaci slojevite arhitekture:
  - smanjene performanse sustava
  - skupa promjena protokola interakcije
  - ponekad teško identificirati jasno odijeljene slojeve



Rainbow Colours Cheese Layered Cake

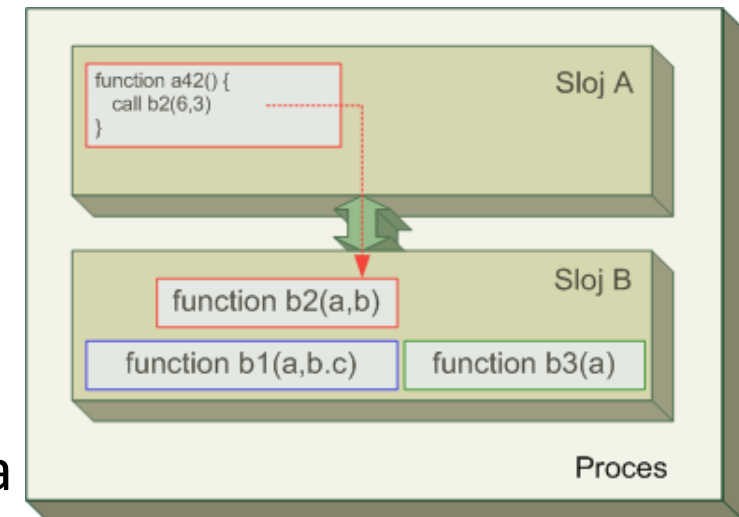


Coconut Milky Yam cheese Layered Cake

# Izvedba slojeva i konektora (I)



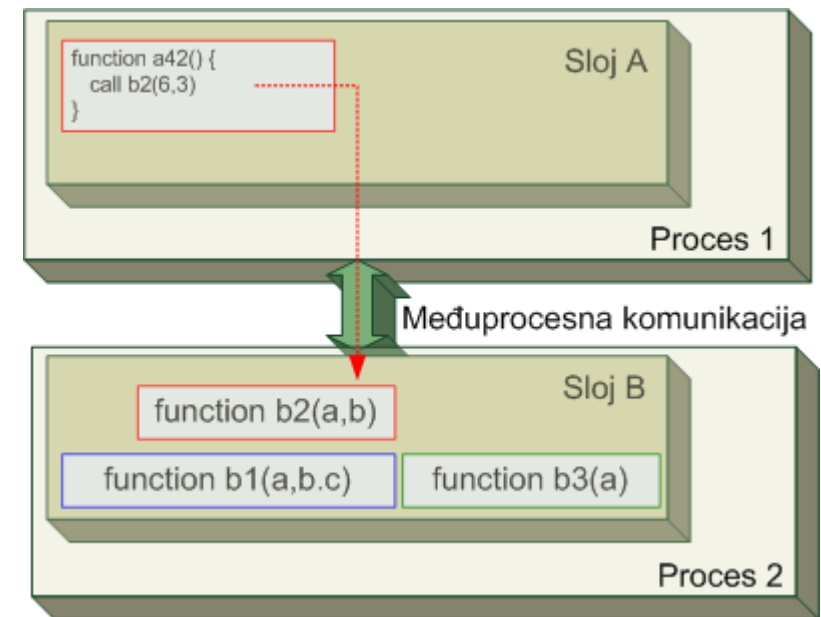
- Monolitna aplikacija:
  - aplikacija je jedan proces izvršavan u okviru operacijskog sustava jednog računala
  - slojevi – logički (i fizički?) odvojene biblioteke funkcija, no čine jedinstven izvršni kod aplikacije
  - konektori između slojeva
    - skup funkcija vidljivih iz susjednog sloja
    - komunikacija – pozivi funkcija susjednog sloja



# Izvedba slojeva i konektora (II)



- Višeprocenska 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 međuprocenske komunikacije
    - komunikacijski protokol

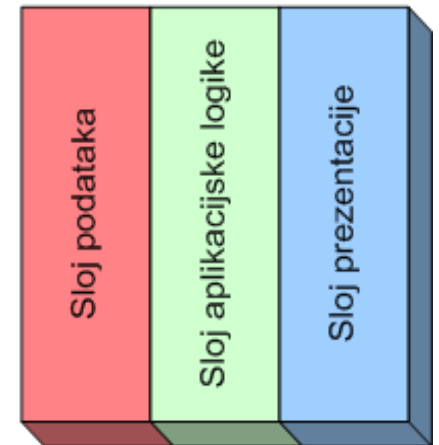
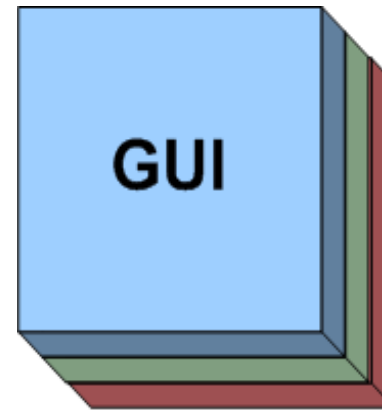




# Slojevi aplikacije

- Tipična arhitektura aplikacije sastoji se od tri sloja:

- sloja prezentacije (GUI)
- sloja aplikacijske logike
- sloja podataka



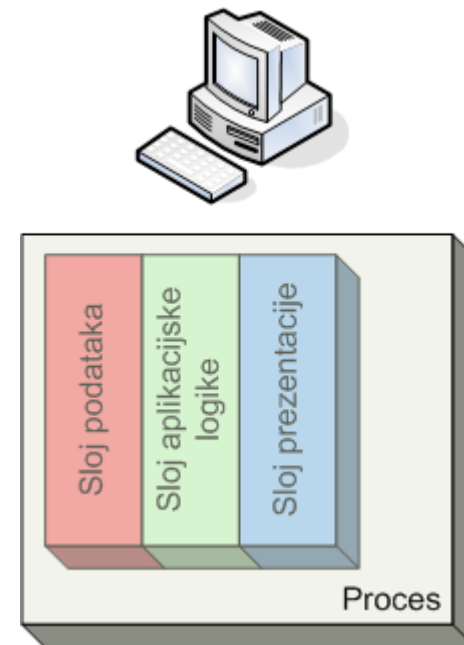
- Unutar tih slojeva mogu se identificirati i finije podjele na dodatne slojeve (ovisno o aplikaciji)

# Arhitekture raspodijeljenih aplikacija

# Monolitna arhitektura



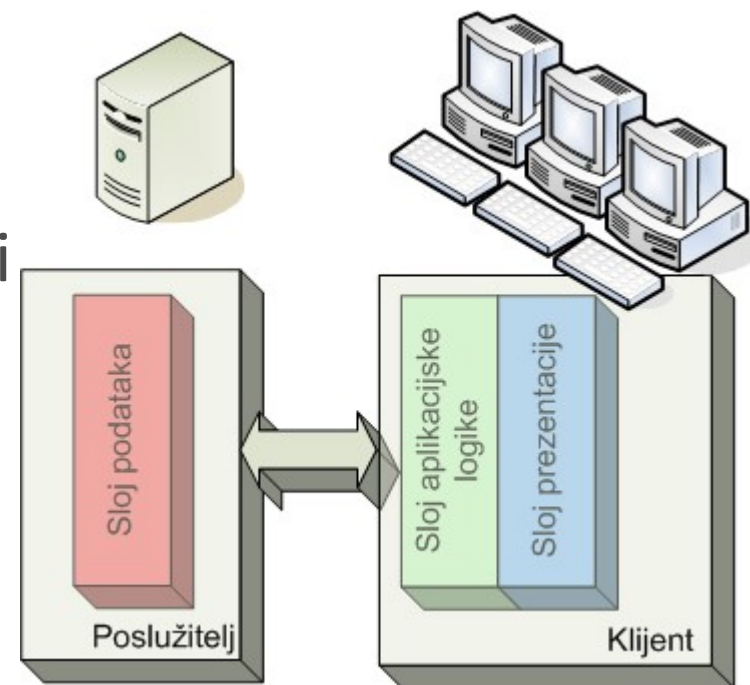
- Svi funkcionalni slojevi aplikacije unutar procesa izvođenog na jednom računalu
- Primjeri
  - obrada teksta, tablični kalkulator, razvojne okoline, *single-player* igre
- Dodatni alati potrebni za omogućavanje grupnog rada (npr. subversion (svn), cvs ...)



# Dvoslojna arhitektura



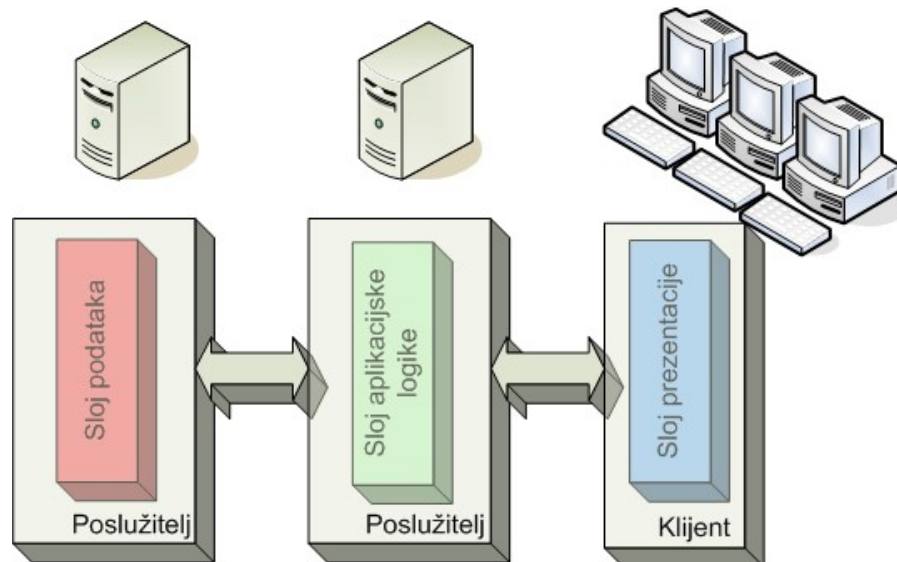
- Funkcionalni slojevi grupirani u dva zasebna arhitekturna sloja (*2-tier*), tj. procesa
  - klijentska aplikacija sadrži funkcionalne slojeve prezentacije i (ako postoji) aplikacijske logike
  - poslužiteljska aplikacija sadrži sloj podataka
- Poslužiteljska aplikacija može istovremeno pružati usluge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



# Troslojna arhitektura



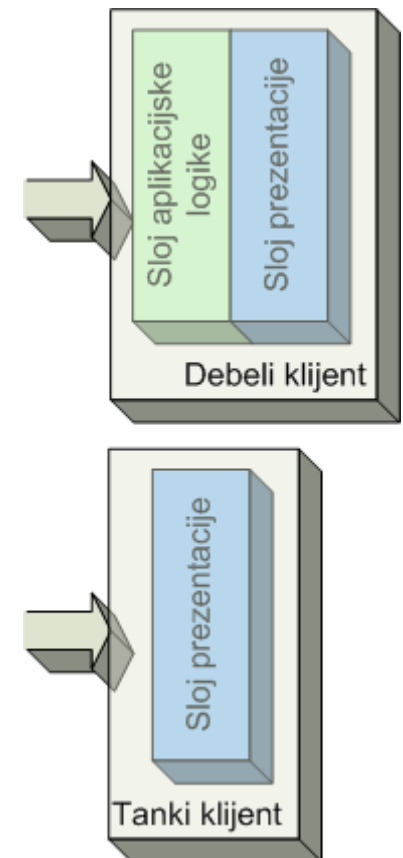
- Funkcionalni slojevi grupirani u tri zasebna arhitekturna sloja (*3-tier*), tj. procesa
  - sloj prezentacije – klijenska aplikacija
  - srednji sloj - sloj aplikacijske logike
  - sloj podataka - baza podataka, ...



# 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



# Karakteristike arhitektura

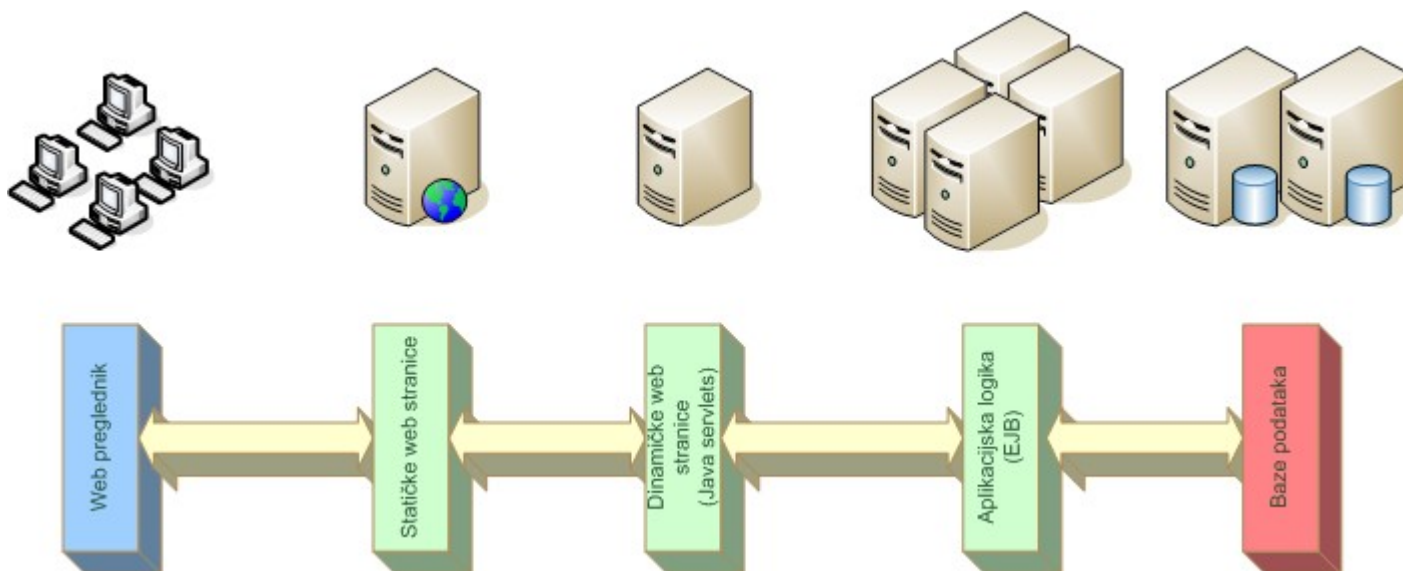
---

- Procjena s obzirom na sljedeće parametre:
  - snaga obrade računala
  - kapacitet spremišta podataka (centralnih, lokalnih, ...)
  - propusnost komunikacijske infrastrukture
  - skalabilnost sustava (npr. u slučaju povećanja broja korisnika, ...)
  - robusnost sustava (npr. nedostupnosti pojedinih komponenti sustava, ...)
  - cijena izgradnje sustava
  - cijena održavanja sustava

# 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

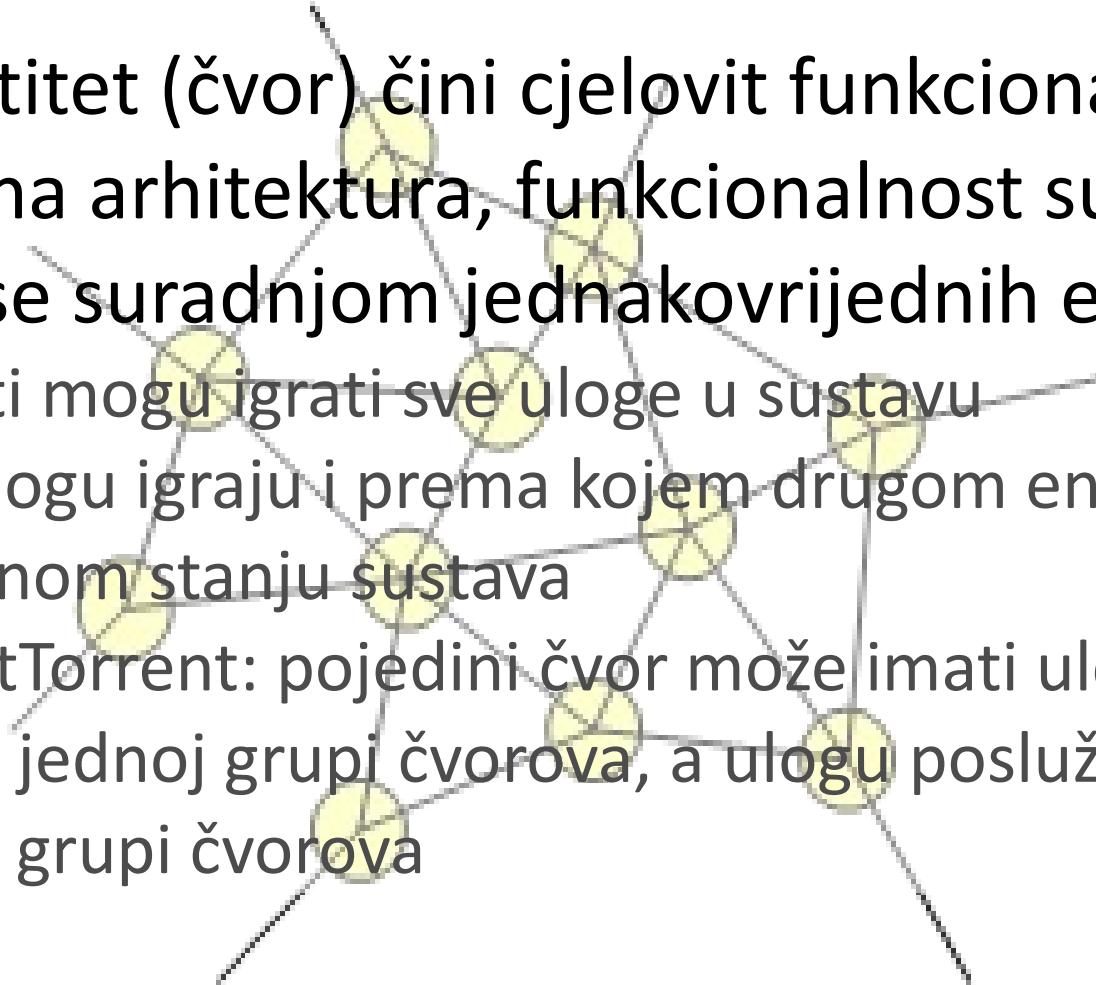




# Peer-to-peer arhitektura

---

- Svaki entitet (čvor) čini cjelovit funkcionalni entitet – monolitna arhitektura, funkcionalnost sustava postiže se suradnjom jednakovrijednih entiteta
  - entiteti mogu igrati sve uloge u sustavu
  - koju ulogu igraju i prema kojem drugom entitetu, ovisi o trenutnom stanju sustava
  - npr. BitTorrent: pojedini čvor može imati ulogu klijenta prema jednoj grupi čvorova, a ulogu poslužitelja prema drugoj grupi čvorova



# Redovi poruka (I)

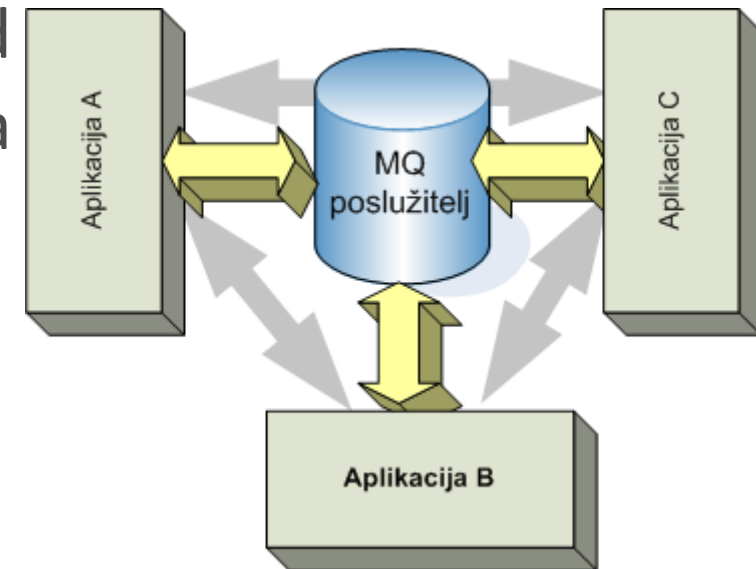
---

- Čvrsta povezanost (*tight coupling*) između procesa/čvorova u višeslojnim i p2p arhitekturama
  - uspostava komunikacijskog kanala između točno određenih sudionika komunikacije
  - sinkrona priroda komunikacije
  - implementacija specifičnog komunikacijskog protokola (jezika)
- Kod povezivanja neovisno izgrađenih sustava navedene karakteristike stvaraju velike probleme!

# Redovi poruka (II)



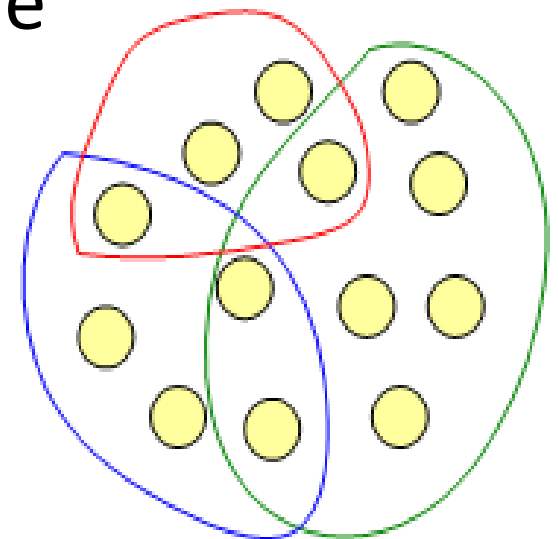
- Redovi poruka:
  - međuspremnici između sudionika komunikacije
  - **proizvođači** spremaju poruke u red
  - **potrošači** dohvaćaju poruke iz reda
  - nema izravnog komunikacijskog kanala između aplikacija
  - komunikacija je asinkrona
  - potrebno je implementirati samo jedan komunikacijski protokol




# Potpuno raspodijeljeni sustavi

---

- Javno dostupne raspodijeljene komponente
  - način implementacije nebitan, bitno sučelje za pristup
  - usluge (web servisi), procedure (RPC, XML-RPC), objekti (SOAP, CORBA)
- Svaka komponenta pruža određenu funkcionalnost, neovisna o aplikaciji
- Raspodijeljeni sustavi kao orkestriranje raspodijeljenih komponenti





Primjeri arhitektura mrežnih usluga

# Usluge na Internetu

---

- Usluge – široko korištene raspodijeljene aplikacije
  - dostupni klijenti i poslužitelji
  - dobro definirani (normirani) komunikacijski protokoli
  - definirane uobičajene adrese (vrata) usluga
- Arhitektura usluga je u pravilu klijent-poslužitelj
  - s (većim ili manjim) varijacijama na temu
- Protokoli većinom varijacije modela “zahtjev-odgovor” (*request - response*):
  - klijent šalje zahtjev poslužitelju
  - poslužitelj obrađuje zahtjev i vraća rezultat klijentu

# Arhitektura usluga klijent-poslužitelj

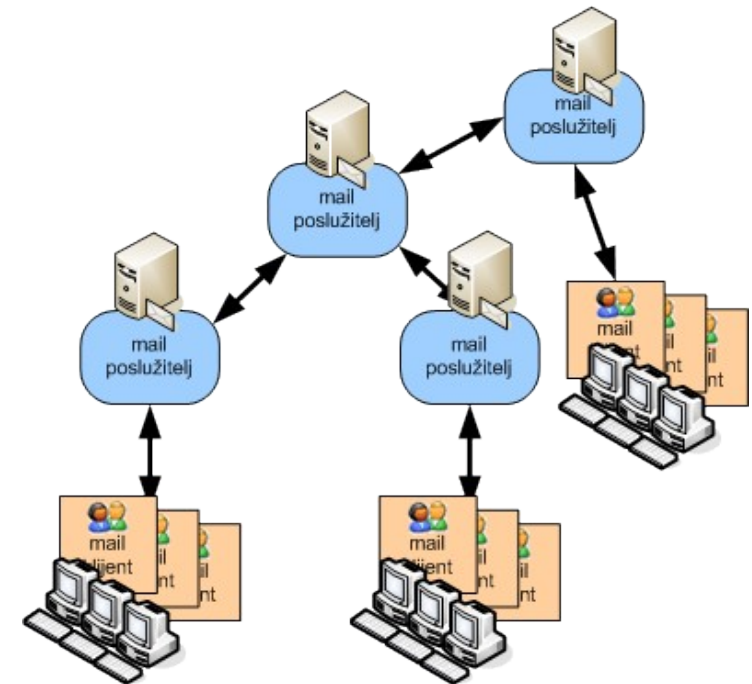
---

- www, dns, ldap, svn – informacijske usluge
  - www: jasni slojevi prezentacije i podataka
  - dns, ldap: uloga klijenta ovisna o konkretnoj primjeni (admin klijent, aplikacija), poslužitelji kao sloj podataka
  - 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



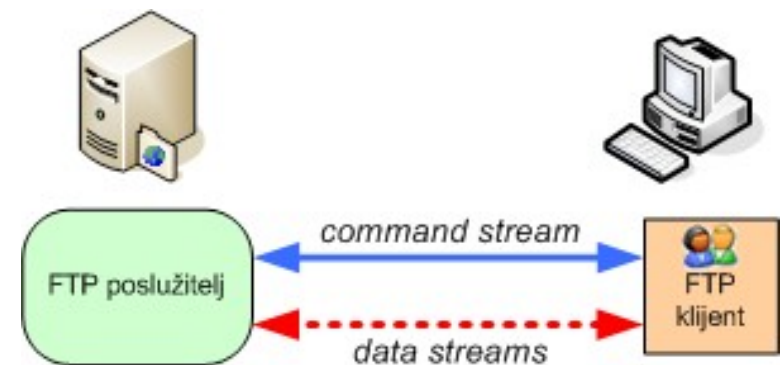
- Mail klijenti:
  - slanje pošte, dohvaćanje pošte iz poštanskog sandučića
  - prezentacijski, djelomično i podatkovni sloj (lokalna pohrana pošte)
- Mail poslužitelji:
  - podatkovni sloj prema klijentima
  - p2p organizacija prosljeđivanja (MTAx  $\leftrightarrow$  MTAy)





# Arhitektura usluge FTP

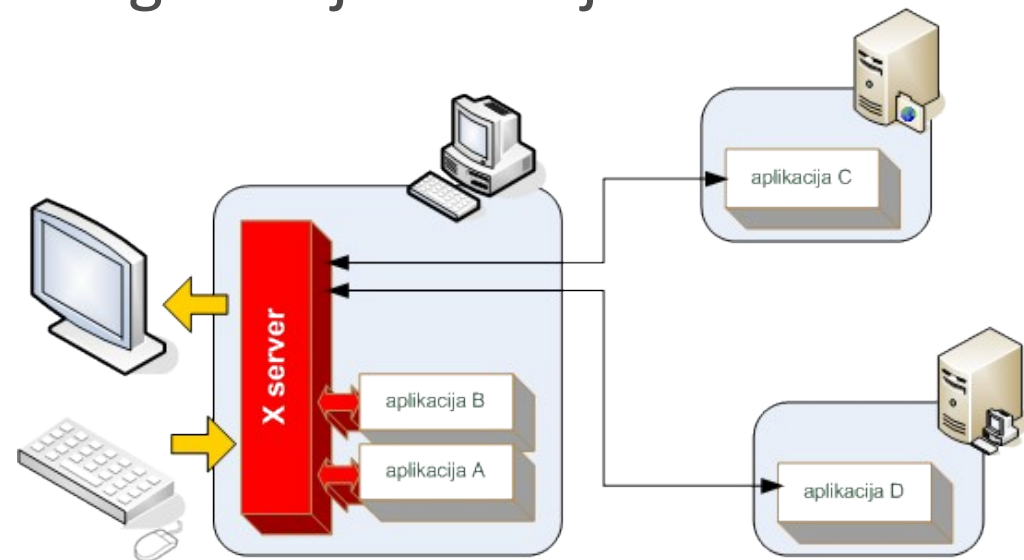
- Prijenos datoteka između dva računala
- Komunikacijski kanal između klijenta i poslužitelja za prijenos naredba i odgovora (*command stream*)
- Za svaki prijenos datoteke otvara se poseban komunikacijski kanal (*data stream*)
  - aktivni mod: poslužitelj inicira vezu
  - pasivni mod: klijent inicira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





Klijenti i poslužitelji kao računala

# Zahtjevi na resurse (I)

---

- Međuovisnost arhitekture aplikacije, nužnih resursa, troškova održavanja, robusnosti, fleksibilnosti ...
- Vrste klijenata: tanki i debeli
  - tanki – veći teret obrade na računalima poslužiteljima, potrebna manja propusnost mreže
  - debeli – manji teret na računalima poslužiteljima, većina obrade na klijentima, potrebna veća propusnost mreže
- Omjer broja klijenata i poslužitelja
  - vrlo velik broj klijenata – rasterećenje poslužitelja delegiranjem dijela poslova klijentima?

# Zahtjevi na resurse (II)

---

- Troškovi sustava:
  - tanki klijenti – potrebni manji resursi na strani korisnika, jeftinija računala, centralizirani resursi, manji troškovi održavanja
  - debeli klijenti – potrebni veći resursi na strani korisnika, skuplja računala, disperzirani resursi, veći troškovi održavanja

# Zahtjevi na resurse (III)

---

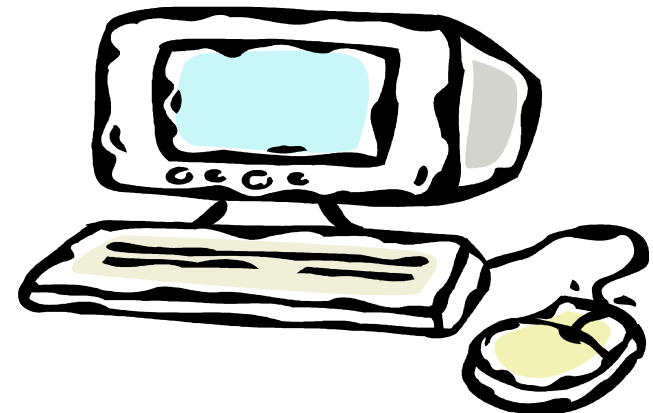
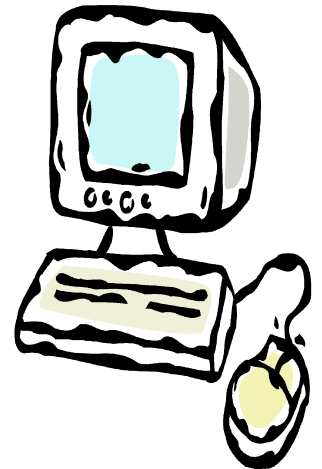
- Brzina komunikacije
  - slojevi na istom računalu: vrlo brza komunikacija, potrebni veći resursi računala poslužitelja, smanjena robusnost sustava
  - slojevi na različitim računalima: sporija komunikacija, manji pojedinačni resursi računala poslužitelja, povećana robusnost sustava
- Fleksibilnost:
  - Više slojeva: veća mogućnost raspodjele slojeva na više računala (raspodjela opterećenja), sporija komunikacija između slojeva

# Računala klijenti

---



- Jedan korisnik
- Grafičko korisničko sučelje, U/I uređaji
- Samostalno ili spojeno na lokalnu mrežu
- Izvođenje monolitnih aplikacija i klijenata raspodijeljenih aplikacija
- Podjela s obzirom na resurse:
  - debeli, hibridni, tanki



# Debeli klijenti

---

- Računala opće namjene (osobna računala, radne stanice, ...)
  - diskovni podsustav za lokalnu pohranu podataka
  - relativno velika lokalna snaga obrade podataka
  - neovisan o poslužitelju i stalnoj mrežnoj povezanosti
  - lokalno izvođenje monolitnih aplikacija
  - izvođenje raspodijeljenih aplikacija (prezentacijskog sloja ili i prezentacijskog i sloja aplikacijske logike) zahtijeva mrežnu povezanost
  - skupa u nabavi, skupa za održavanje, neiskorišteni resursi



# Hibridni klijenti

---



- Računala bez lokalnog diskovnog podsustava
  - moraju biti povezana s mrežnim datotečnim sustavom
  - ovise o stalnoj povezanosti s poslužiteljem
  - snaga obrade ekvivalentna debelim klijentima
  - smanjeni troškovi održavanja, veća pouzdanost

# Tanki klijenti

---



- Tanki klijenti
  - sučelje prema uslugama na poslužitelju
  - ovisno o mrežnoj komunikaciji
  - manja snaga obrade, nema diskova
  - klijentske aplikacije, web preglednik,...
  - koncepcija terminala. RemoteDesktop, X Windows
  - jednostavno i jeftino održavanje, robusnost sustava



# Tanki klijenti – déjà vu?



- Povijest tankih klijenata
  - tekstni terminali
    - kasne 70-te, VT-52, VT-100
    - serijska veza sa *mainframe* računalom
  - grafičke radne stanice
    - rane 90-te, X Windows
    - povezane LAN-om s poslužiteljem
  - pojam tankih klijenata iznjedrio marketing (mora zvučati kao nešto novo !)



# Tanki ili debeli klijenti?

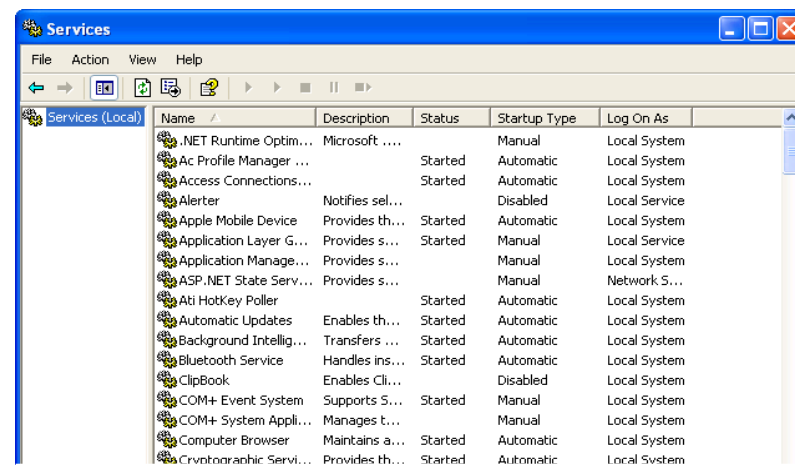
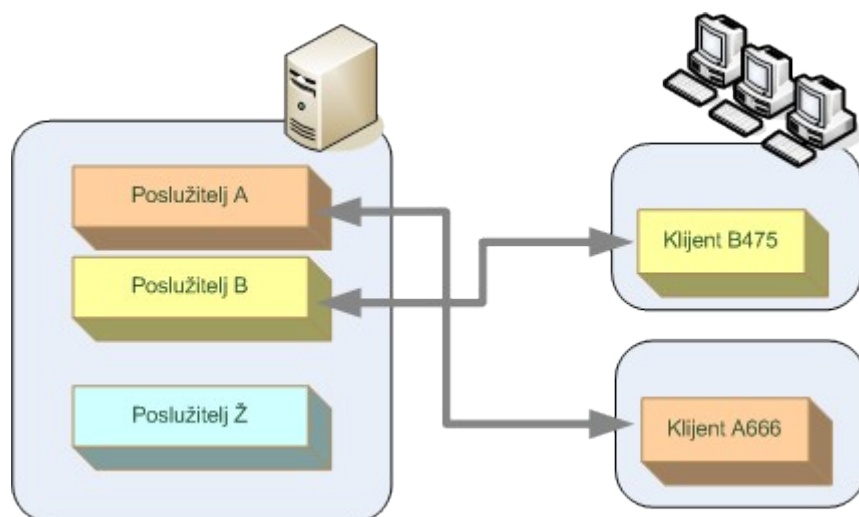
---



# Računalo poslužitelj



- Domaćin jednom ili više programa poslužitelja
- Stalno vezano na mrežu (internet)
- Poslužitelj opće namjene može biti i obično osobno računalo, s Windows ili Linux operacijskim sustavom



Nestajanje stroge granice između klijenta i poslužitelja (usluge na Win XP)

# Namjenski poslužitelji (I)

---

- Potrebna snaga obrade ovisna o aplikaciji
  - najčešće nije kritična; procesi većinom blokirani na čekanju završetka U/I operacija (npr. dohvat datoteke)
- Paralelizam bitniji od brzine slijednog izvršavanja
  - bolje usporedno množiti više matrica nego brzo jednu
  - ovo čini bitnu razliku u odnosu na osobna računala!
- Operacijski sustavi moraju podržavati
  - velik broj procesa, učinkovito prebacivanje konteksta, stabilnost sustava u slučaju pada jednog procesa ...

# Namjenski poslužitelji (II)

---

- Propusnost podataka
  - brzina pristupa i dohvata informacija na diskovnom podsustavu (npr. SCSI), propusnost sabirnice i mrežnog sučelja, velika količina memorije, hijerarhijska organizacija memorije (razine priručne memorije)
- Pouzdanost sustava
  - specijalizirane izvedbe memorijskog podsustava (brzina, detekcija grešaka, redundancija)
  - sklopovska i podatkovna redundancija (dvostruka napajanja, RAID, ...)
  - pouzdane mehaničke komponente (diskovi, hlađenje, ...)



# Poslužitelji FER-a





# Google platforma

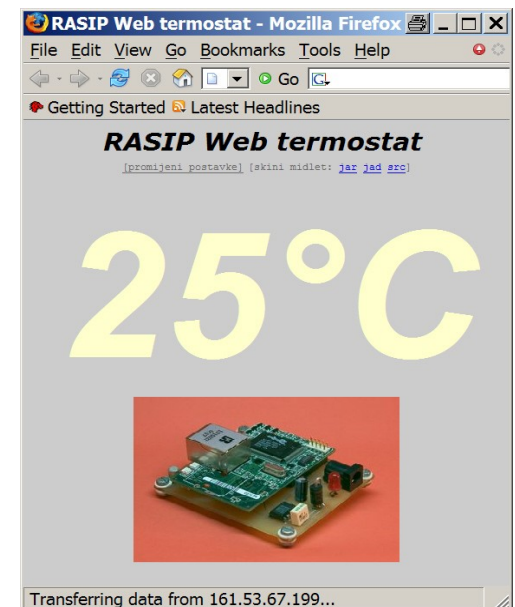
---




[http://en.wikipedia.org/wiki/Google\\_platform](http://en.wikipedia.org/wiki/Google_platform)

# Druga krajnost namjenskih poslužitelja

- Rabbit2000 CPU
  - 256KB Flash ROM
  - 128KB RAM
  - GPIO
  - 4 x ASC
  - 2 x SSC
  - RTCC
  - Ethernet kontroler



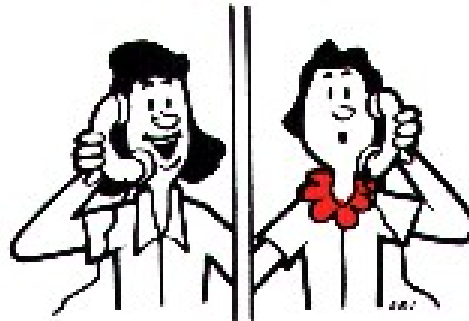


# Međuprocesna komunikacija u raspodijeljenim sustavima

# Preduvjeti komunikacije dva procesa

---

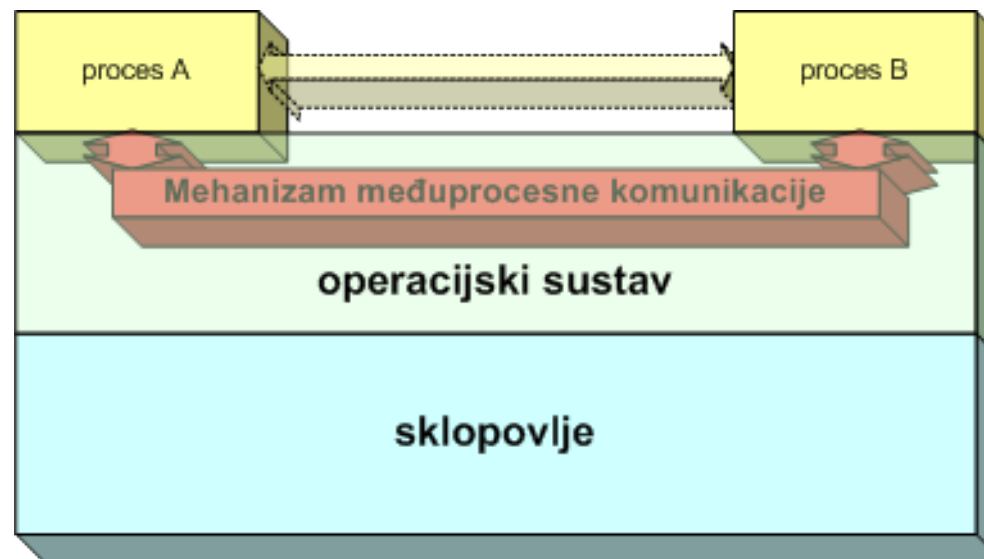
- Dva procesa koji trebaju surađivati:
  - “adrese” procesa - kako se pronalaze?
  - mehanizam komunikacije - kako komuniciraju?
  - komunikacijski kanal - putem čega komuniciraju?
  - protokol komunikacije - kako se sporazumijevaju?



# Kolocirani procesi



- Izvršavani na istom računalu, pod istim operacijskim sustavom, dostupni isti resursi
- Koriste se mehanizmi IPC operacijskog sustava
- OS posrednik između procesa



# Međuprocesna komunikacija

---

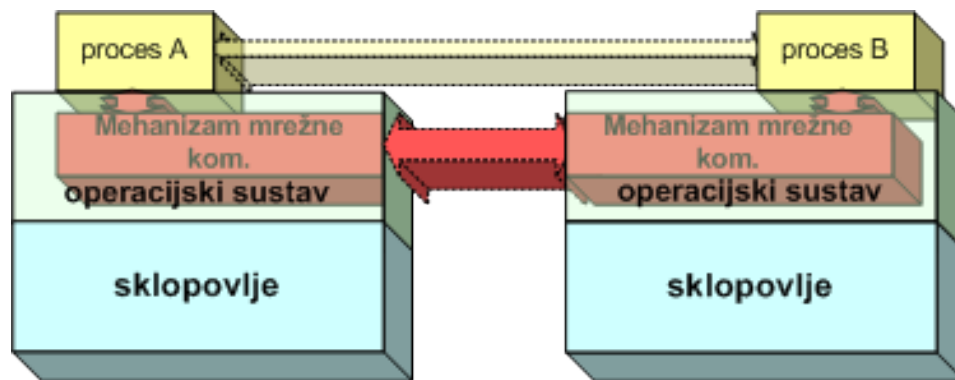
- Mehanizmi međuprocesne komunikacije:
  - jednosmjerni / dvosmjerni
  - sinkroni / asinkroni

redovi poruka (*message queues*),  
 cjevovodi (*pipelines*),  
 dijeljena memorija (*shared memory*),  
 semafori (*semaphores*),  
 signali (*signals*),  
 datoteke (*files*),  
 utičnice (*sockets*) ...

# Raspodijeljeni procesi



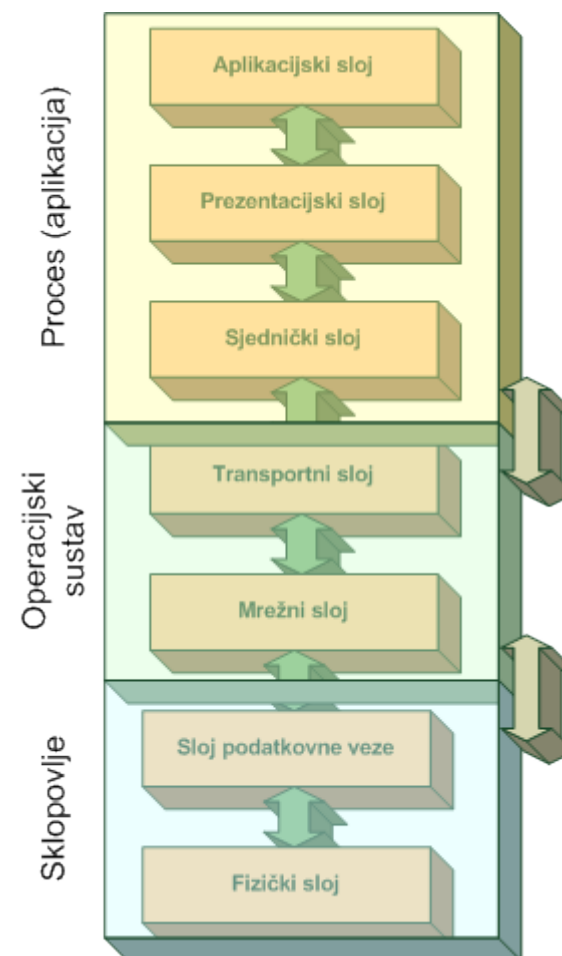
- Dva osnovna pristupa:
  - tradicionalni OS:
    - procesi “svjesni” raspodijeljenosti
  - raspodijeljeni OS:
    - privid jednog računala,
    - procesima raspodijeljenost skrivena mehanizmima OS-a



# Mehanizmi mrežne komunikacije



- ISO/OSI referentni model
- U ovisnosti o stvarnom protokolu:
  - postojanje komunikacijskog sloja
  - mjesto implementacije pojedinog komunikacijskog sloja



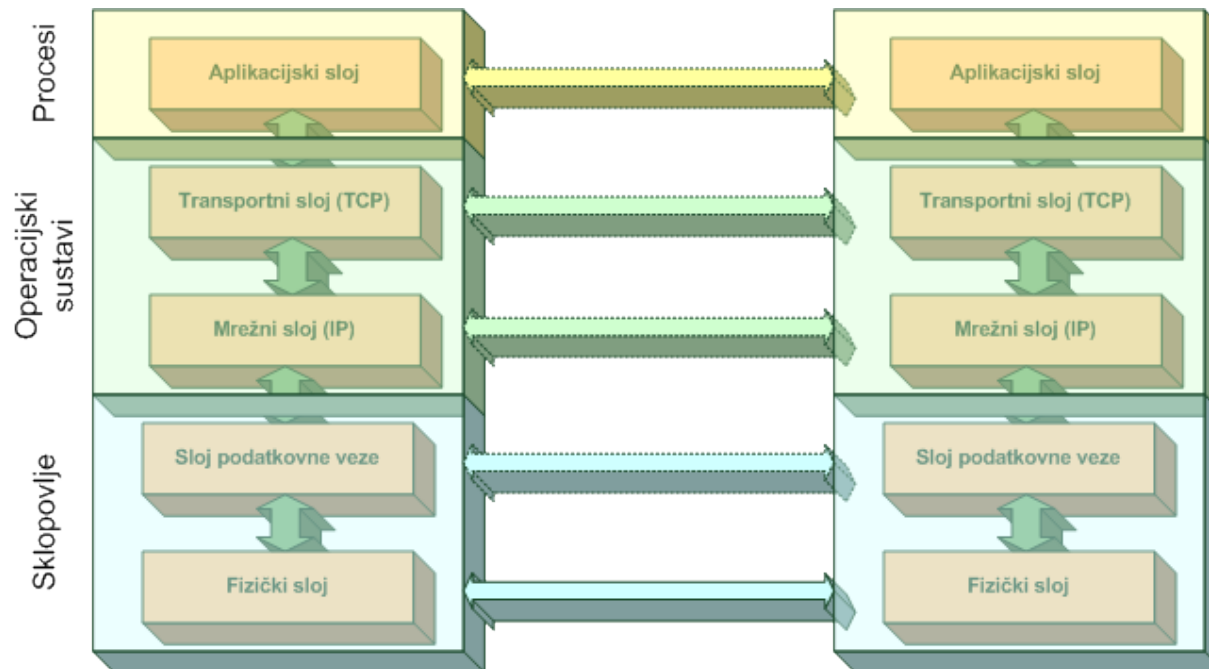


# Heterogenost platformi

---

- Komunicirajući procesi se mogu izvršavati na različitim platformama
- Problemi mogući na svim razinama:
  - složaj protokola
  - širina riječi
  - poredak okteta (*endianess*)
  - kodiranje znakova, terminiranje retka
  - ...
- Network byte order – big endian

# TCP/IP složaj protokola



- Osnova interoperabilnosti na internetu
  - dostupan na gotovo svim platformama
- Osnovna funkcionalnost jednostavno ostvariva
  - skalabilnost (od superračunala do mikrokontrolera)

# Struktura paketa TCP i IP



- Enkapsulacija slojeva unutar paketa
  - unutar *payload* dijela TCP i UDP paketa sadržaj aplikacijskog protokola
  - unutar zaglavlja TCP i UDP paketa podaci o vratima procesa pošiljatelja i primatelja (*ports*)
  - unutar *payload* dijela IP paketa smješten TCP ili UDP paket
  - unutar zaglavlja IP paketa podaci o adresama računala pošiljatelja i primatelja (*source, destination IP address*)

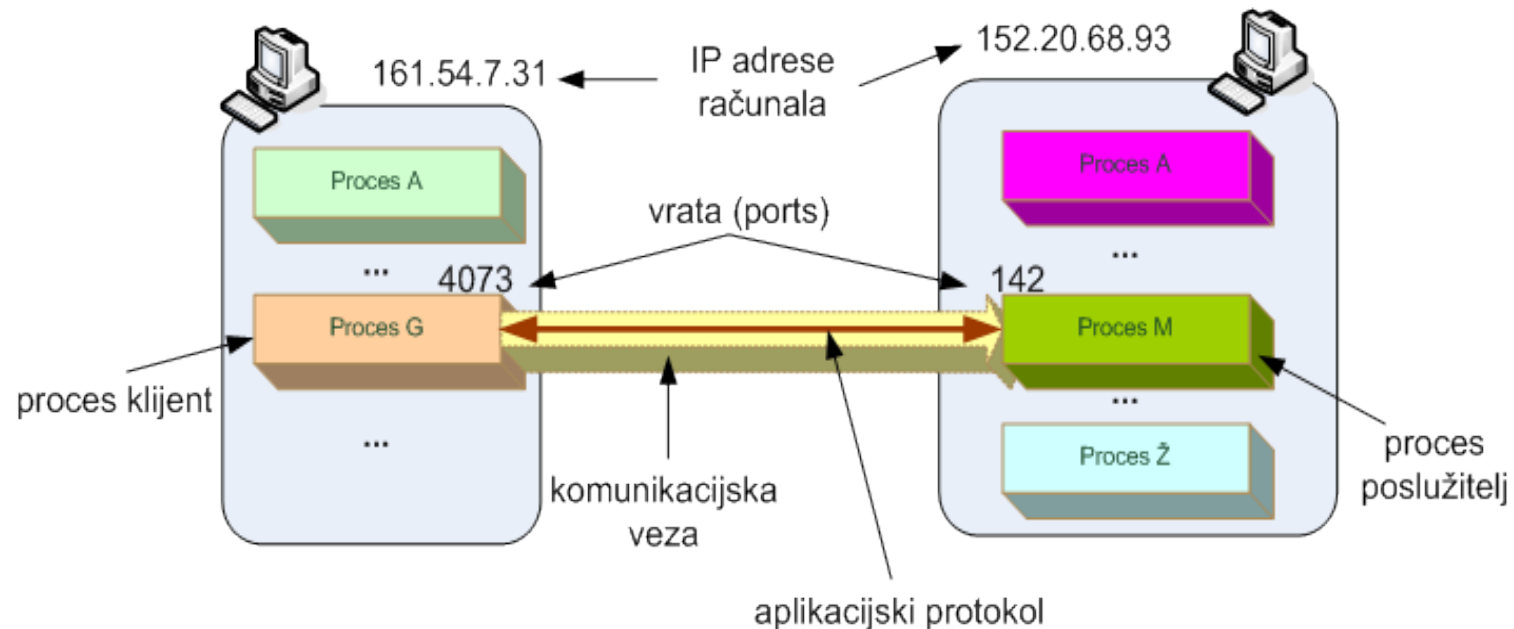
# TCP i UDP

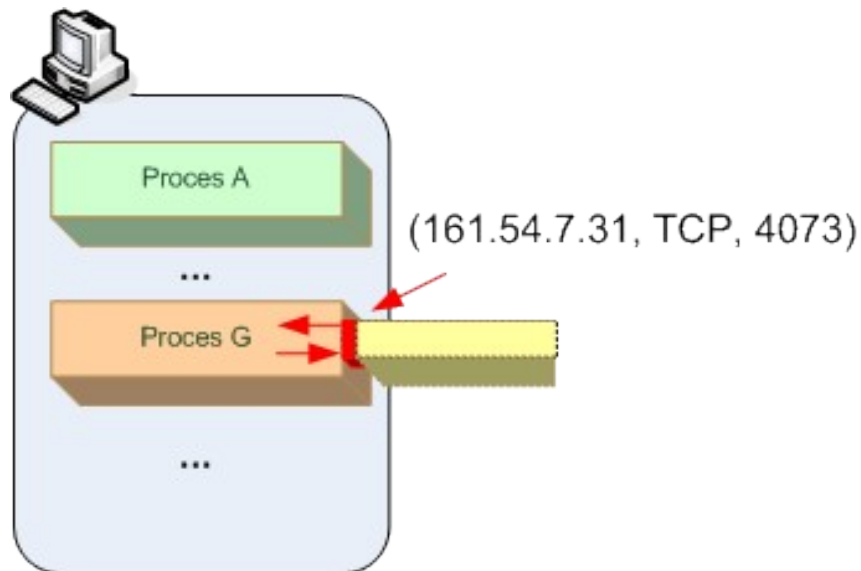
---

- TCP – spojna veza
  - ispravan redoslijed paketa
  - ispravljanje pogrešaka u prijenosu (gubitak, dupliciranje)
  - možemo promatrati kao čvrst komunikacijski kanal – dvosmjerni tok (*stream*) kroz koji se razmjenjuju podaci
- UDP – bezspojna veza
  - nema mehanizma ispravljanja pogrešaka
  - komunikacija na razini razmjene pojedinačnih paketa (datagrama), ne dvosmjernog toka

# TCP/IP međuprocesna komunikacija

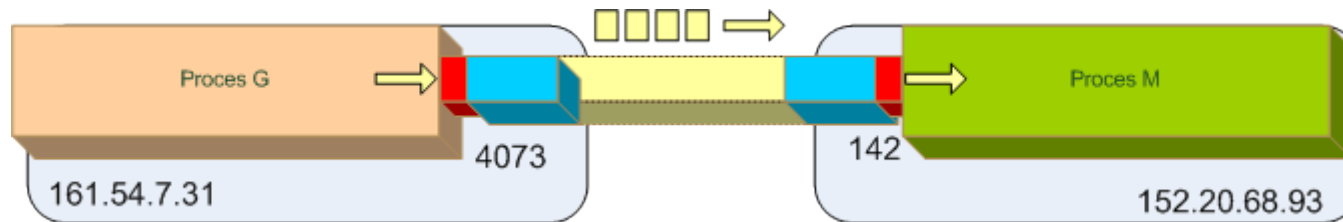
- Klijent i poslužitelj procesi
  - klijent inicira vezu (TCP) ili šalje datagram (UDP) – mora znati adresu procesa poslužitelja (IP adresa, vrata)
  - poslužitelj osluškuje na određenim vratima (*port*), prihvaća vezu (TCP) ili datagram (UDP)
  - jezik komunikacije - aplikacijski protokol





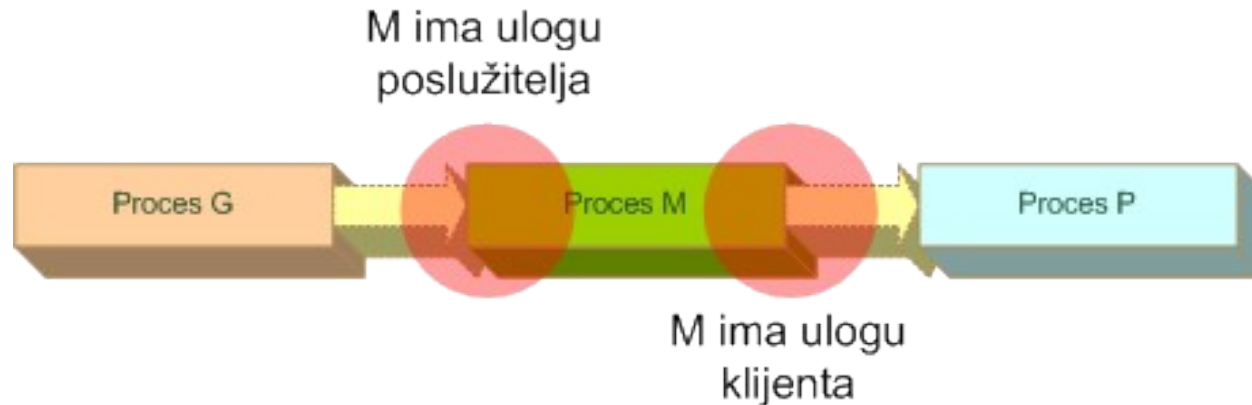
- Utičnica (*socket*)
  - programska apstrakcija krajnje točke komunikacije (communication endpoint) korištenjem TCP/IP
  - jedinstveno određena IP adresom, protokolom (TCP, UDP, ...) i brojem vrata
  - po UNIX filozofiji utičnice su sučelja prema dvosmjernom cjevovodu (cjevovod = komunikacijski kanal)

# Spojni komunikacijski kanal (TCP!)



- **Pisanje u cjevovod**
  - poslani podaci pohranjuju se u odlazni međuspremnik
  - iz međuspremnika se šalju na drugu komunikacijsku točku
  - na prijamnoj strani čuvaju se u dolaznom međuspremniku
- **Čitanje iz cjevovoda**
  - čitaju se samo trenutno raspoloživi podaci u međuspremniku!
  - problem fragmentacije podataka

# Klijent i poslužitelj kao uloge



- Proces je klijent ili poslužitelj samo u kontekstu promatrane veze
  - stalne uloge klijenta ili poslužitelja (npr. Web)
  - proces može istovremeno imati obje uloge (npr. DNS: rekurzivno razlučivanje adrese, MTA prosljeđivanje pošte)
  - proces može istovremeno imati istu ulogu prema više drugih procesa (npr. Web preglednik je višestruki klijent)



# Povezanost životnog vijeka

---

- Životi klijenta i poslužitelja većinom neovisni
  - klijenti kao korisnička sučelja prema uslugama, u pravilu kratkog životnog vijeka
  - poslužitelji dugog životnog vijeka, vezani uz rad/dostupnost računala poslužitelja
- Posebni slučajevi uske povezanosti životnog vijeka:
  - npr. X Window System - klijenti ne mogu postojati bez aktivnog poslužitelja

# Trajnost veze

---

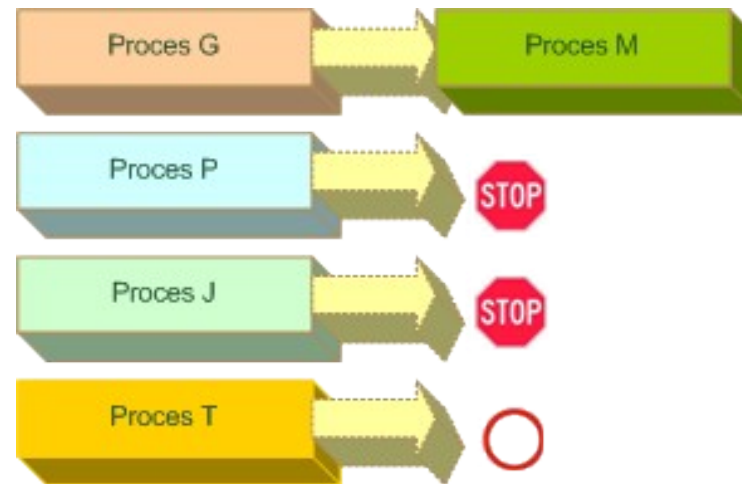
- Trajnost veze i kanala ovisna o konkretnoj primjeni:
  - bez uspostave veze (UDP): razmjena malih količina podataka, npr. DNS upiti
  - kratkotrajne veze: jedna transakcija (HTTP 1.0, FTP data)
  - produljene veze: više transakcija u kontekstu jedne veze (HTTP 1.1, SMTP, POP3, IMAP)
  - duge veze: rad klijenta ovisan o postojanju veze (npr. telnet, X11, RTP)
- Kompromis između cijene uspostave veze, količine prenošenih podataka i održavanja nekorištene veze

# Veze na strani poslužitelja

---

- Podjela po broju i načinu održavanja prihvaćenih veza od strane poslužitelja
  - poslužitelj prihvaća samo jednu vezu
  - poslužitelj prihvaća više veza
    - jedna veza po procesu
    - jedna veza po niti
    - asinkrona komunikacija s više klijenata
    - multipleksiranje komunikacije s klijentima

# Jedna veza istovremeno



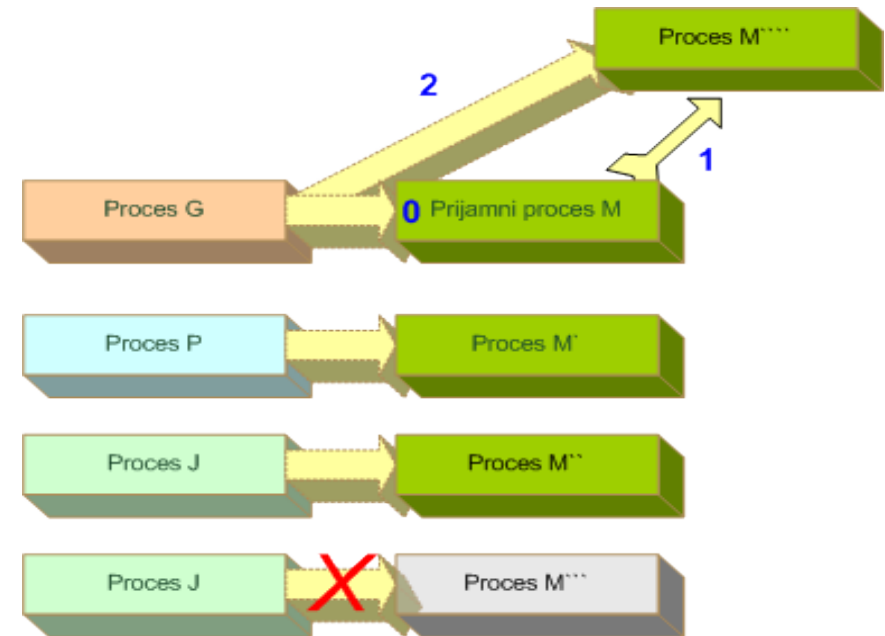
- Jedan proces poslužitelj prihvaća vezu
  - može istovremeno održavati vezu samo s jednim klijentom
  - novi zahtjevi pristigli tijekom trajanja veze stavljaju se u red čekanja ili odmah odbijaju
  - pogodno za kratkotrajne veze, manji broj zahtjeva za vezom pravilno vremenski raspoređen

# Jedan proces po vezi



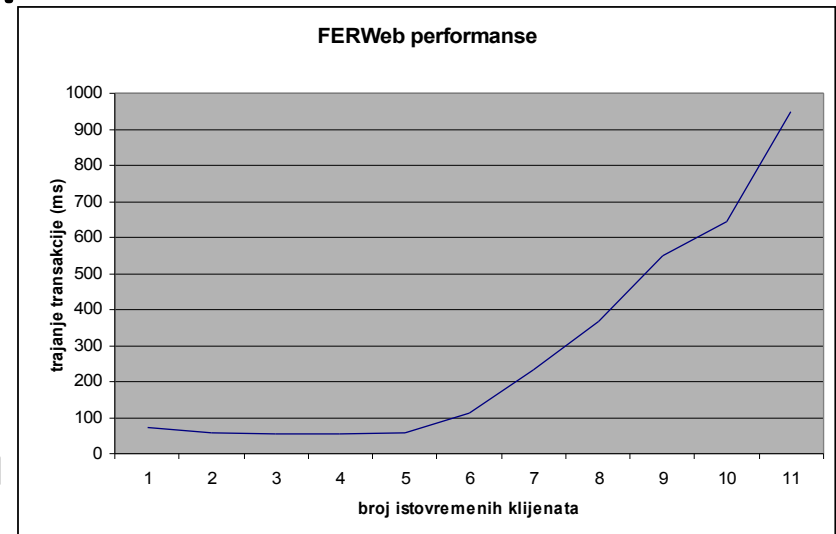
- Prijamni proces poslužitelj

- (0) čeka na uspostavu veze
- (1) stvara kopiju procesa
- (2) preusmjerava vezu na kopiju
- (0) prijamni proces ostaje slobodan za prihvatanje nove veze
- Nakon zatvaranja veze s klijentom kopija prestaje s radom
- pogodno za duže veze, veći broj zahtjeva, neravnomjerno raspoređenih u vremenu



# Utjecaj broja procesa na performanse

- Utjecaj na procesorsko vrijeme:
  - stvaranje procesa (jednokratno)
  - prebacivanje konteksta (trajno)
- Utjecaj na memoriju:
  - kopija podataka procesa (trajno)
- Zagušenje računala poslužitelja
  - prebacivanja konteksta
  - korištenja virtualne memorije
  - rezultira nemogućnošću pristupa klijenata poslužitelju

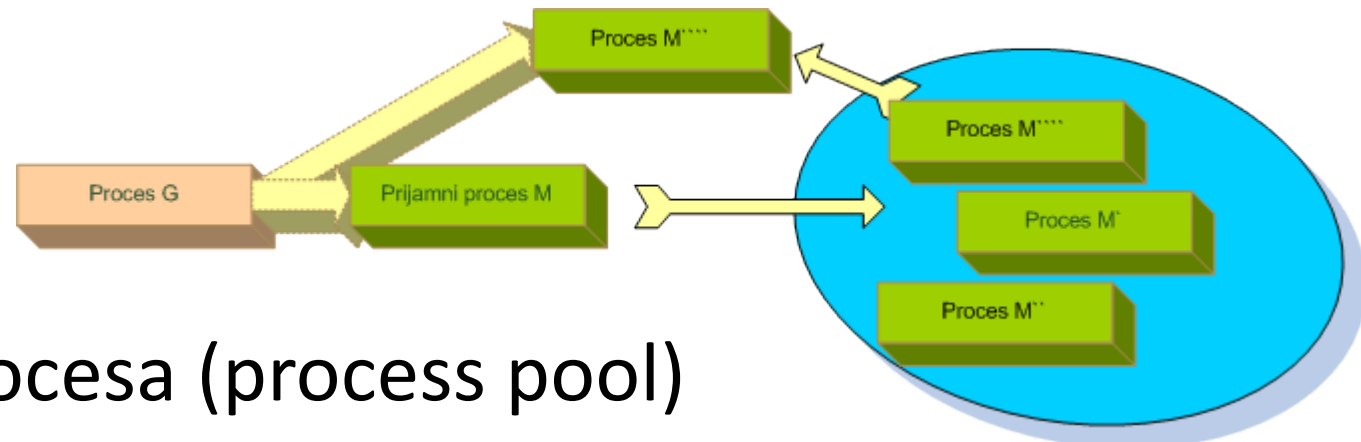


# Utjecaj broja klijenata na performanse

---

- Više veza od strane jednog klijenta:
  - paralelizam obrade zahtjeva, bolje performanse?  
**KRIVO!**
  - problem sporog učenja TCP veze (*sliding window*)
  - problem opterećenja mreže
  - problem broja procesa na računalu poslužitelju
  - problem potpunog zastoja u nedostatku veza/zagušenju
  - fair odnos prema drugim klijentima
- Korištenje politike: jedan klijent – jedna veza
  - paralelizam obrade na razini aplikacijskog protokola, ne veza

# Bazen procesa



- Bazen procesa (process pool)
  - određen broj procesa poslužitelja stalno pokrenut, neaktivni čekaju “u bazenu”
    - troše memorijski prostor
    - ne troše vrijeme procesora
  - dolaskom zahtjeva za vezu prijamni proces “dohvaća” proces poslužitelj iz bazena i preusmjerava vezu
    - kompromis između stalno pokrenutih i pokretanja novih procesa
  - po završetku veze proces se vraća u bazen



# Jedna nit po vezi

---

- Višeprocenost je karakteristika UNIX-a
- Windows, Java – paralelizam izvođenja temeljen većinom na nitima (*threads*)
  - niti “lakše” od procesa (memorija, kontekst, ...)
  - složenija implementacija zbog dijeljenja istog konteksta izvršavanja (memorije, funkcija, ...)
  - veća osjetljivost na pogreške (jedna nit ruši čitav proces)
- Problem zagušenja poslužitelja velikim brojem niti
- Bazen niti (*thread pool*)

# Asinkrona komunikacija s klijentima

---



- Pretplate na komunikacijske događaje
  - resurs na čije događaje se pretplaćuje
  - događaj na koji se pretplaćuje
  - *callback* funkcija koja se poziva pojavom događaja
    - signali – mehanizam međuprocene komunikacije
- Asinkronost:
  - proces/program se izvršava sinkrono
  - izvršavanje *callback* funkcija asinkrono s obzirom na glavni program
- Slično prekidnim potprogramima (FRISC, ARM ...)

# Multipleksirana komunikacija

---



- Jedan proces/nit prihvaća i opslužuje više veza
- Multipleksiranje komunikacije poslužitelja i više aktivnih klijenata:

1. osluškiju se sve veze istovremeno

- blokirajuće ili neblokirajuće osluškivanje

2. detektira se aktivnost na nekoj od veza

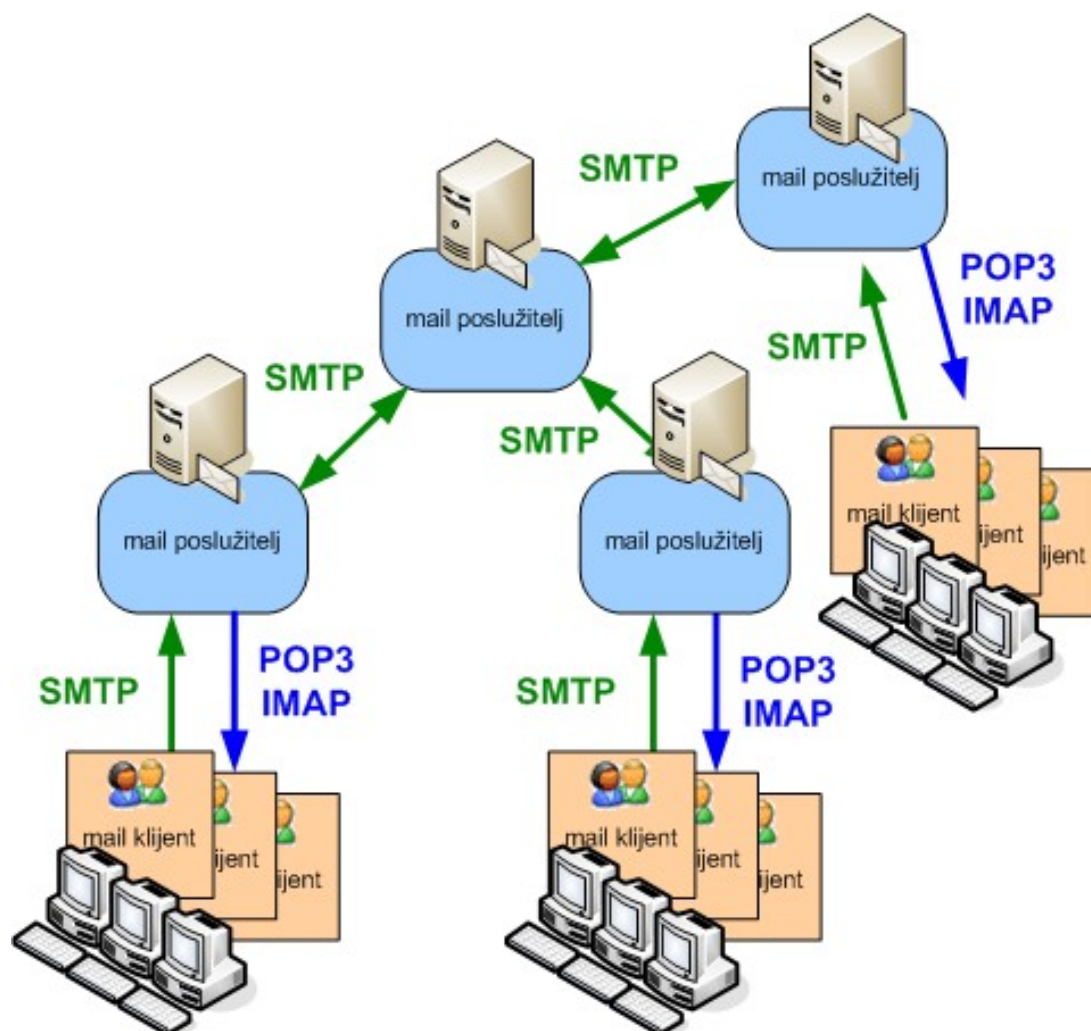
- podaci pristigli od klijenta
- međuspremnik spreman za privat podataka
- zahtjev za uspostavom nove veze

3. poslužuje se aktivna veza ili prihvaća nova veza

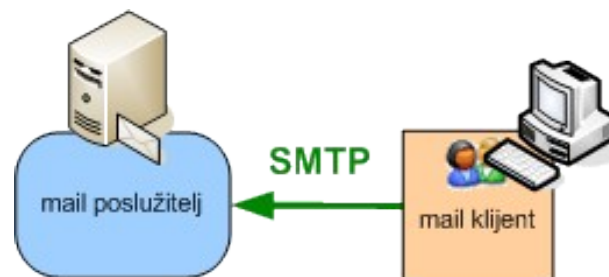
The background of the slide features large, light green letters 'O' and 'R' that are partially visible, with the 'O' on the left and the 'R' on the right, both having white cutouts.

# Aplikacijski protokoli

# Sustav elektroničke pošte (e-mail)



# Primjer konverzacije - SMTP



C: <client connects to service port 25>

C: HELO snark.thyrsus.com

S: 250 OK Hello snark, glad to meet you

C: MAIL FROM: <esr@thyrsus.com>

S: 250 <esr@thyrsus.com>... Sender ok

C: RCPT TO: cor@cpmy.com

S: 250 root... Recipient ok

C: DATA

S: 354 Enter mail, end with "." on a line by itself

C: Scratch called. He wants to share

C: a room with us at Balticon.

C: .

S: 250 WAA01865 Message accepted for delivery

C: QUIT

S: 221 cpmy.com closing connection

C: <client hangs up>

*sending host identifies self*

*receiver acknowledges*

*identify sending user*

*receiver acknowledges*

*identify target user*

*receiver acknowledges*

*end of multiline send*

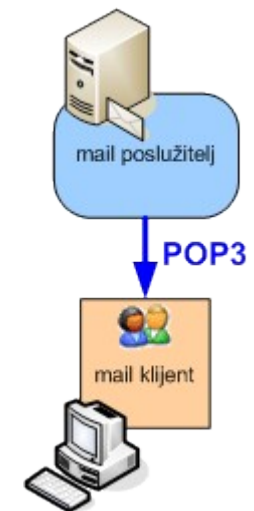
*sender signs off*

*receiver disconnects*

# Primjer konverzacije - POP3



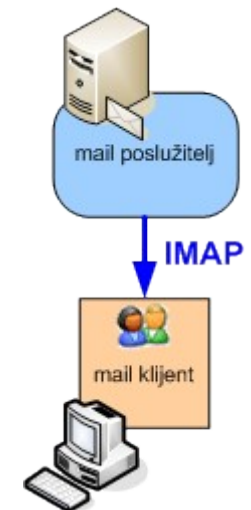
```
C: <client connects to service port 110>
S: +OK POP3 server ready <1896.6971@mailgate.dobbs.org>
C: USER bob
S: +OK bob
C: PASS redqueen
S: +OK bob's maildrop has 2 messages (320 octets)
C: STAT
S: +OK 2 320
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends the text of message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: RETR 2
S: +OK 200 octets
S: <the POP3 server sends the text of message 2>
S: .
C: DELE 2
S: +OK message 2 deleted
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
C: <client hangs up>
```



# Primjer konverzacije - IMAP



```
C: <client connects to service port 143>
S: * OK example.com IMAP4rev1 v12.264 server ready
C: A0001 USER "frobozz" "xyzzzy"
S: * OK User frobozz authenticated
C: A0002 SELECT INBOX
S: * 1 EXISTS
S: * 1 RECENT
S: * FLAGS (\Answered \Flagged \Deleted \Draft \Seen)
S: * OK [UNSEEN 1] first unseen message in /var/spool/mail/esr
S: A0002 OK [READ-WRITE] SELECT completed
C: A0003 FETCH 1 RFC822.SIZE
S: * 1 FETCH (RFC822.SIZE 2545)
S: A0003 OK FETCH completed
C: A0004 FETCH 1 BODY[HEADER]
S: * 1 FETCH (RFC822.HEADER {1425}
<server sends 1425 octets of message payload>
S: )
S: A0004 OK FETCH completed
C: A0005 FETCH 1 BODY[TEXT]
S: * 1 FETCH (BODY[TEXT] {1120}
<server sends 1120 octets of message payload>
S: )
S: * 1 FETCH (FLAGS (\Recent \Seen))
S: A0005 OK FETCH completed
C: A0006 LOGOUT
S: * BYE example.com IMAP4rev1 server terminating connection
S: A0006 OK LOGOUT completed
C: <client hangs up>
```





# Aplikacijski protokoli

---

- Jezik sporazumijevanja komunicirajućih entiteta
- Binarni ili tekstni format
  - binarni: kompaktniji, manje opterećenje mreže, manje opterećenje računala, nečitak za čovjeka
  - tekst: dulji, veće opterećenje mreže i računala, lakše praćenje i ispravljanje pogrešaka
- Uloge u protokolu:
  - svaki od entiteta igra ulogu definiranu protokolom
  - uloga određuje ponašanje tijekom konverzacije

# Aplikacijski protokoli i nespojna veza

---

- Primjeri korištenja nespojne veze (UDP)
  - jednostavni zahtjev-odgovor temeljeni protokoli, podaci stanu unutar jednog UDP paketa (npr. DNS)
  - bitna brzina prijenosa (npr. NFS), aplikacija se brine o ispravnosti prenošenih podataka
  - tokovi podataka (streaming) – audio, video
  - (višesmjerno) odašiljanje (*broadcast, multicast*) podataka

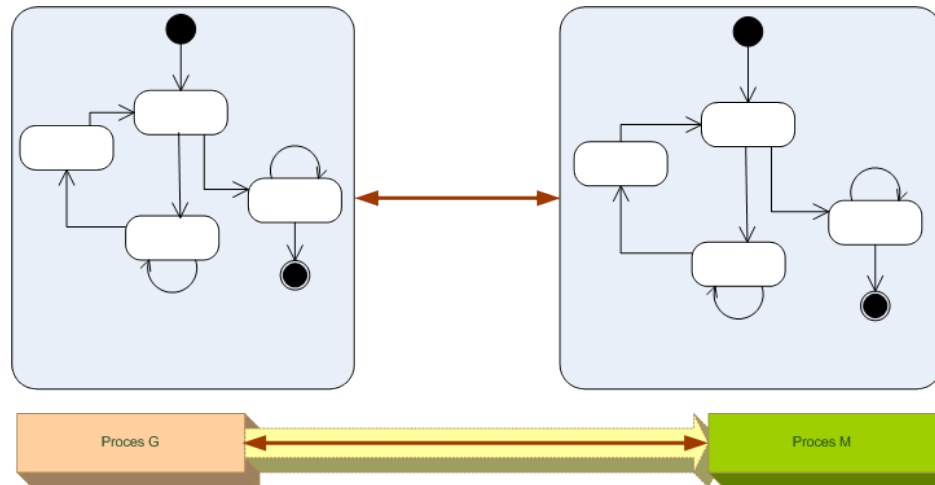
# Aplikacijski protokoli i spojna veza

---



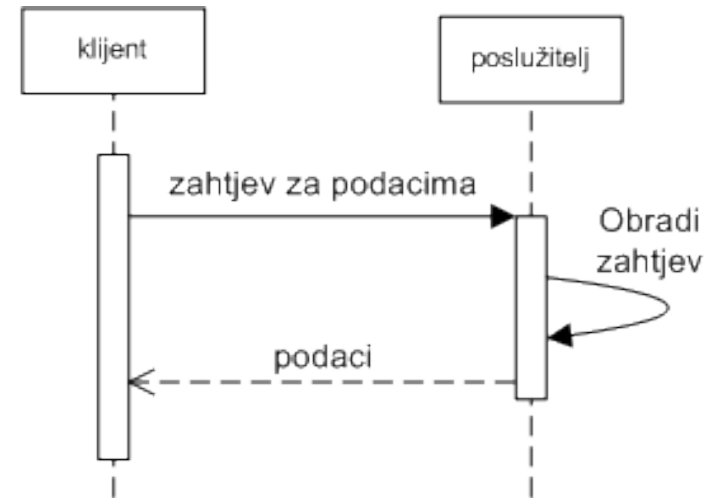
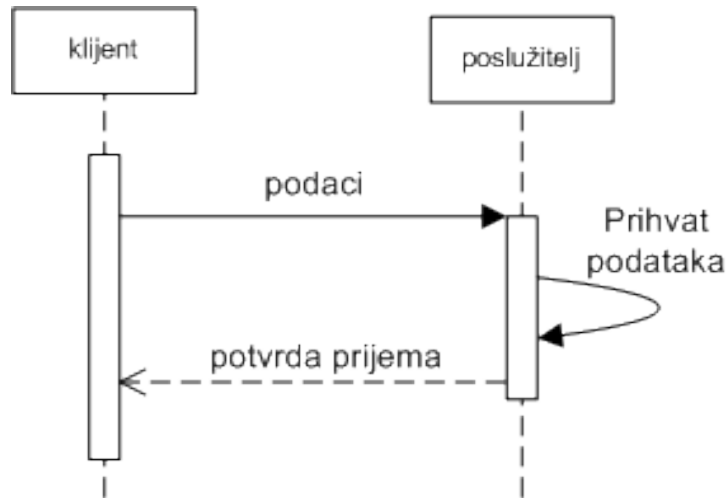
- Primjeri korištenja spojne veze (TCP)
  - pouzdani prijenos veće količine podataka tokovima (npr. FTP, HTTP)
  - dugotrajnije veze (npr. FTP command, telnet)
  - složene konverzacije, bitan redoslijed poruka (npr. SMTP, IMAP)
  - najčešće korištena u ostvarenju informacijskih usluga,
  - naš fokus razmatranja na ovoj vrsti protokola

# Protokoli i konverzacije



- Konverzacija je instanca protokola
  - ... kao što je objekt instance razreda
  - procesi učesnici konverzacije razmjenjuju poruke u skladu s korištenim protokolom
  - konverzacija se odvija unutar komunikacijske veze
- Praćenje stanja konverzacije korištenjem DKA

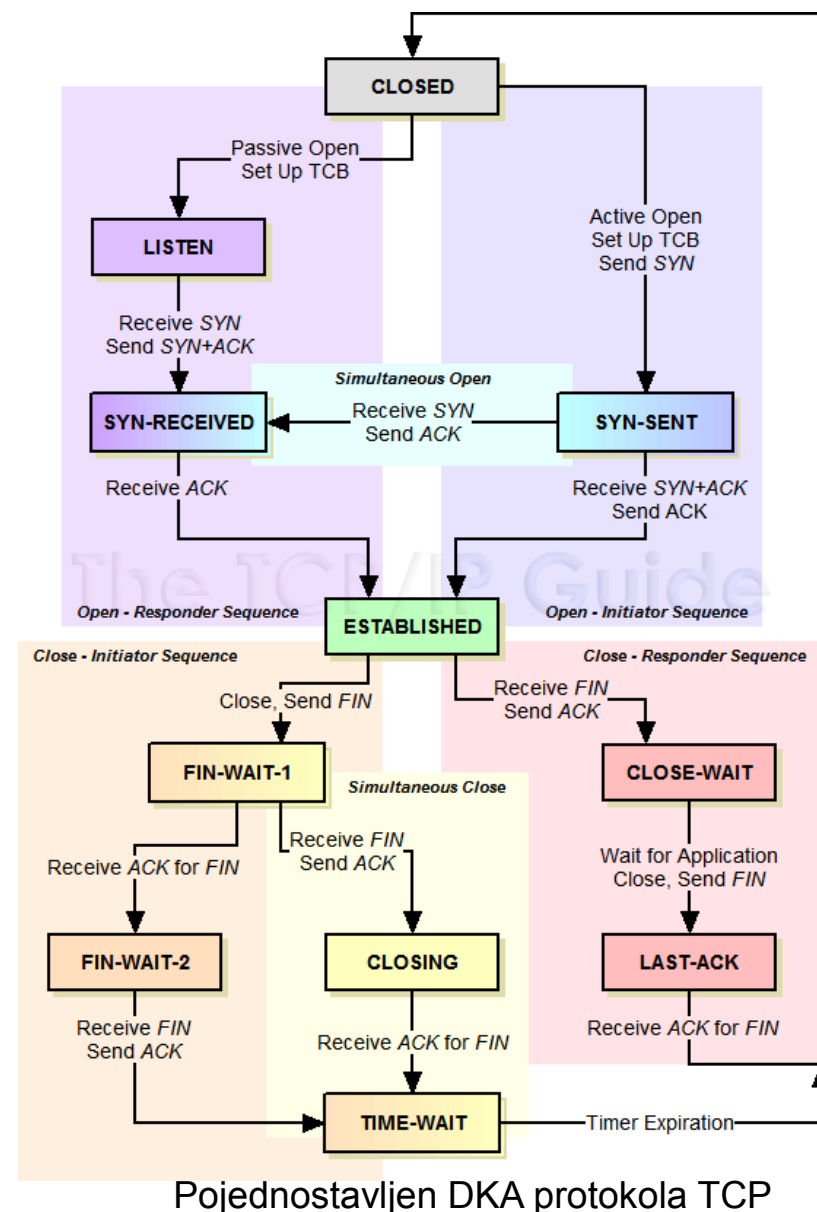
# Jednostavni protokoli



- Primjeri jednostavnih protokola
  - jednosmjernan tok podataka
  - razmjena zahtjeva i odgovora (*request-response*)
- Jednostavno praćenje stanja pojedine konverzacije
  - nema potrebe za eksplicitnim praćenjem stanja (~~DKA~~)

# Složeni protokoli (I)

- U složenim protokolima:
  - mora se poštovati redoslijed konverzijskih akcija definiran komunikacijskim protokolom
  - sljedeće akcije ovise o stanju konverzacije, također mogu ovisiti i o stanju usluge



# Složeni protokoli (II)

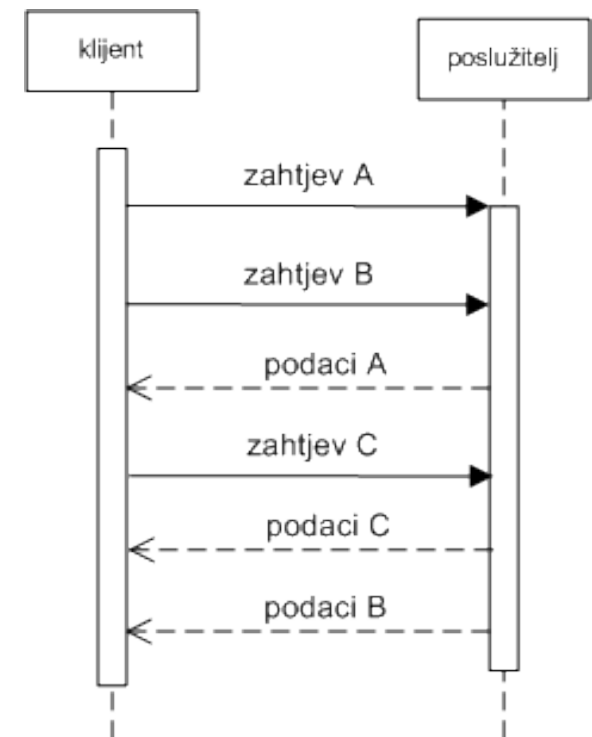
---

- Prisjećanje na DKA (Uvod u teoriju računarstva)
  - Mooreov, Mealyjev konačni automat -  $(\Sigma, \Gamma, S, s_0, \delta, \omega)$
  - razlika u trenutku stvaranja izlaznog simbola (funkcija  $\omega$ )
- Definiranje prokola
  - DKA na strani klijenta (DKAc) i na strani poslužitelja (DKAs)
  - $\Sigma_c = \Gamma_s, \Sigma_s = \Gamma_c$
  - trenutno stanje konverzacije određeno stanjem i na strani klijenta (DKAc) i na strani poslužitelja (DKAs)

# Slijedne i paralelne konverzacije



- Slijedne konverzacije
  - samo jedna istovremeno aktivna konverzacija po vezi
- Paralelne konverzacije
  - može biti aktivno više konverzacija u kontekstu jedne veze
  - kojoj konverzaciji akcija pripada?
    - identifikatori konverzacije!
  - zaseban (par) DKA po konverzaciji za praćenje njena stanja





# Simetrični i asimetrični protokoli

---



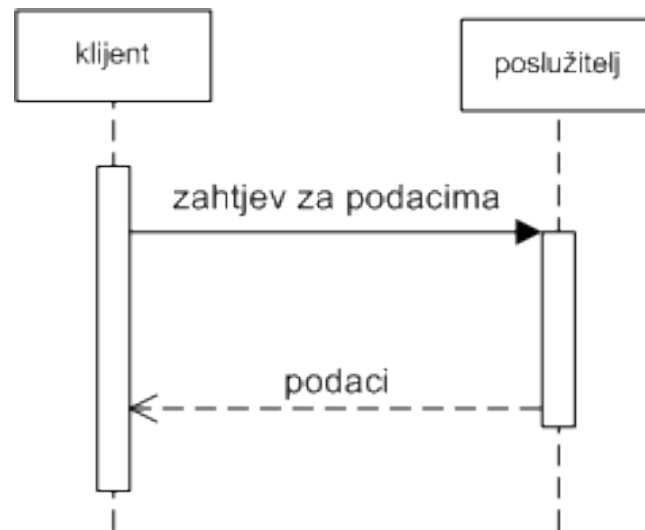
- Uloga u konverzaciji (klijent ili poslužitelj)
  - dodijeljena u trenutku uspostave veze
  - npr. e-mail klijent dobiva ulogu klijenta u komunikaciji s poslužiteljem korištenjem POP3 protokola
- Asimetrični protokoli:
  - uloga nepromjenljiva tijekom trajanja veze (npr. HTTP)
  - u nekim protokolima entiteti mogu zamijeniti uloge (SMTP: promjena smjera prosljeđivanja poruka)
- Simetrični protokoli
  - uloge određena na razini konverzacije, ne veze (p2p)

# Strategije prijenosa podataka

---

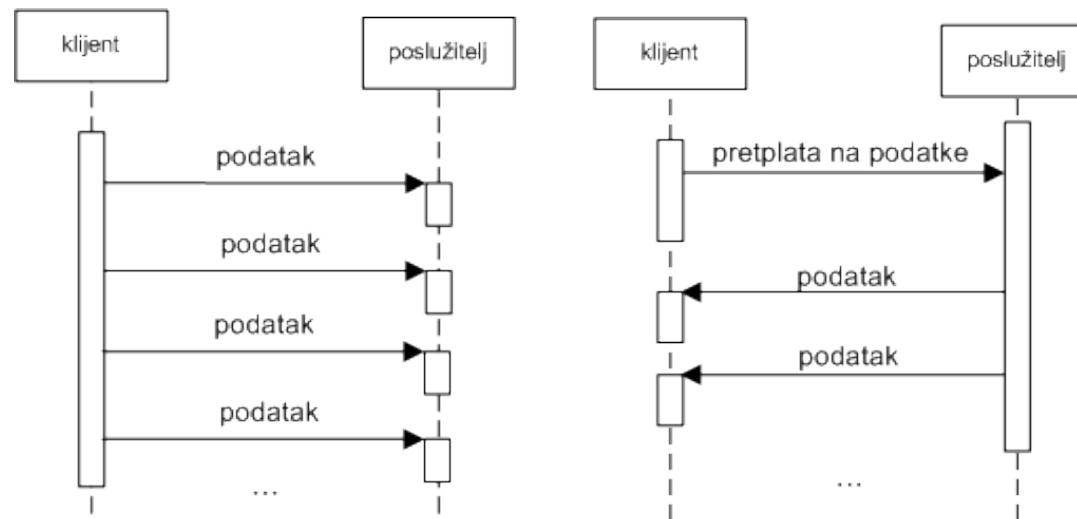
- Dva ključna pitanja:
  - tko posjeduje podatke?
  - tko ima inicijativu kod njihova prenošenja?
  
- Mogući odgovori:
  - korisnik ili poslužitelj
  - opet korisnik ili poslužitelj

# 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 strane 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

# Zatvaranje veze

---

- **Eksplicitno zatvaranje veze**
  - protokol sadrži naredbe za zatvaranje veze
  - npr. IMAP: klijent šalje LOGOUT, poslužitelj vraća BYE i LOGOUT, klijent zatvara vezu
- **Implicitno zatvaranje veze**
  - slanjem zadnje poruke u konverzaciji, zadnjeg okteta prenošenih podataka
  - npr. zatvaranje FTP data veze nakon prenesenih svih podataka

The background of the slide features large, light green, stylized letters 'O' and 'R' that are partially visible, framing the central text.

Mehanizmi protokola

# Mehanizmi protokola

---

- RFC 3117 - On the Design of Application Protocols:
  - uokvirenje poruka (*framing*)
  - kodiranje sadržaja (*encoding*)
  - izvještavanje o stanju (*reporting*)
  - asinkronost konverzacija (*asynchrony*)
  - vjerodostojnost (*authentication*)
  - zaštita podataka (*privacy*)



# Uokvirenje poruka

---

- Jednostavna detekcija kraja jednorednih poruka
  - npr. REQUEST index.html<CR><LF>
- Problem detekcije kraja duljih poruka
  - npr. prenošene binarne datoteke ili duljeg teksta
- Tri osnovne metode okvirenja poruka:
  - umetanjem okteta (*octet stuffing*)
  - brojanjem okteta (*octet counting*)
  - uništavanjem veze (*connection blasting*)

# Uokvirenje umetanjem okteta

```
C: DATA
S: 354 Enter mail. end with "." on a line by itself
C: Scratch called. He wants to share
C: a room with us at Balticon.
C: .
S: 250 WAA01865 Message accepted for delivery
```

- **Primjer prenošenja sadržaja e-pošte u SMTP**
  - poruka se terminira retkom u kojem se nalazi samo točka
  - ako je takav redak valjan sadržaj poruke, prije slanja na početak retka dodaje se još jedna točka (po prijemu briše)
  - prednost: u trenutku početka prenošenja poruke pošiljatelju ne mora biti poznat čitav njen sadržaj
  - mana: sporo, dodatna obrada poruke i na pošiljatelju i na primatelju, nije pogodno za binarne podatke

# Uokvirenje brojanjem okteta

```
...  
C: A0004 FETCH 1 BODY[HEADER]  
S: * 1 FETCH (RFC822.HEADER {1425}  
  <server sends 1425 octets of message payload>  
S: )  
S: A0004 OK FETCH completed  
C: A0005 FETCH 1 BODY[TEXT]  
...
```

- Primjer dohvata e-pošte IMAP klijentom
  - prije početka slanja poruke pošiljalac primatelju šalje duljinu poruke u oktetima
  - prednost: brzina, minimalna obrada kod slanja i primanja
  - mana: čitava poruka mora biti raspoloživa prije slanja (kako bi se odredila njena duljina)

# Uokvirenje uništavanjem veze

---

- Stvaranje nove veze za prijenos jedne poruke
  - tipičan primjer korištenja u protokolu FTP (*data* veza)
  - Potrebno vrijeme za prenošenje podataka o parametrima nove veze (host, port), za otvaranje nove veze ...
  - pogodno za dulje (binarne) datoteke, za manje datoteke vrlo neučinkovito

# Kodiranje sadržaja poruka

---

- Poruka se sastoji od zaglavlja i tijela (MIME)
- Zaglavlje: jedan ili više redaka s atributima (opis prenošenih podataka)
  - <ime>=<vrijednost><CR><LF>
  - prazan redak terminira zaglavlje
- Tijelo poruke sadrži podatke
  - u “sirovom” obliku, ili
  - kodirane prije transporta, dekodirane nakon transporta
    - Base64, ...

# Reprezentacija stanja

...	
C: RCPT TO: mario.zagar@fer.hr	<i>identify target user</i>
S: <b>250</b> root... Recipient ok	<i>receiver acknowledges</i>
C: DATA	
S: <b>354</b> Enter mail, end with "." on a line by itself	
C: Pozdrav svim studenticama i studentima	
C: na predmetu Otvoreno računarstvo.	
C: .	<i>end of multiline send</i>
S: <b>250</b> WAA01865 Message accepted for delivery	
C: QUIT	<i>sender signs off</i>
S: <b>221</b> fer.hr closing connection	<i>receiver disconnects</i>
C: <client hangs up>	

- Mehanizam prenošenja rezultata naredbe i stanja sustava na udaljenoj strani (većinom poslužitelju):
  - uspješno izvedene naredbe
  - trajne ili privremene greške
  - ostalih stanja konverzacije
- brojke namijenjene programu, tekst čovjeku

# Asinkronost

---




- Način obrade naredaba unutar jedne konverzacije:
  - **sljedno:** ne može se zaprimiti nova naredba dok izvođenje prethodne nije završeno
    - jednostavna izvedba klijenta i poslužitelja
    - neučinkovito – vrijeme između izvođenja dvije konverzacije
  - **protočna struktura naredaba:** poslužitelj prihvata naredbe i pohranjuje ih u FIFO strukturu, izvršava sljedno
    - nema gubitka vremena kod čekanja nove naredbe od klijenta
    - zahtijeva paralelizam izvršavanja u izvedbi poslužitelja
  - **paralelno izvršavanje:** naredbe se prihvataju u FIFO i paralelno izvršavaju (u ovisnosti o broju raspoloživih niti)
    - za jednostavne konverzacije (request-response)
    - složena izvedba i na klijentu i na poslužitelju
    - problemi *fair* korištenja veze (kontrola toka, segmentacija poruka), izgladnjivanje, potpuni zastoј

# Sigurnosni aspekti

---

- Vjerodostojnost
  - provjera identiteta korisnika ili procesa u komunikaciji
- Zaštita podataka
  - zaštita od prisluškivanja i izmjene prenošenih podataka





Stanja usluga

# Stanja usluge


---

- Usluge bez očuvanja stanja (*stateless*)
  - svaka akcija neovisna o prethodnim akcijama
  - jednostavne usluge, *request-response* protokoli
    - npr. osnovna funkcionalnost web poslužitelja
- Usluge s očuvanjem stanja (*stateful*)
  - rezultat akcije (i samo odvijanje konverzacije) ovisi o prethodnim akcijama

# Stanja usluge – kontekst stanja

---

- Kontekst očuvanja stanja:
  - kontekst veze
    - npr. FTP – radno kazalo na udaljenom računalu
  - kontekst klijenta
    - npr. stanje sandučića e-pošte korisnika
  - globalni kontekst
    - npr. sadržaj tablice baze podataka



# Otvoreno računarstvo

## Uniform Resource Identifier (URI)

# Lokalna identifikacija resursa

---

- Put u datotečnom sustavu (jedinствeno) identificira resurs
  - vrijedi samo unutar konteksta pojedinog računala
- Mrežni datotečni sustavi uvode nešto širi kontekst identifikacije resursa
  - putovi vrijede unutar konteksta lokalne mreže

`\\fileserver\shared\nastava\or\pred\uri.pdf`

# Globalna identifikacija resursa

---



- Problemi na globalnoj razini (Internet, Zemlja, svemir)
  - ne postoji jedan “globalni datotečni sustav” koji bi određivao jedinstvenu identifikaciju resursa
  - različite vrste resursa i moguće akcije nad njima
  - različite lokacije resursa i načini pristupa resursima
- Potreban odgovor na sljedeća pitanja:
  - kako globalno identificirati resurs?
  - kako locirati taj resurs?
  - a zatim i kako pristupiti resursu?

# URI

---

*“A Uniform Resource Identifier (URI) is **a compact sequence of characters** that identifies an abstract or physical resource.”*

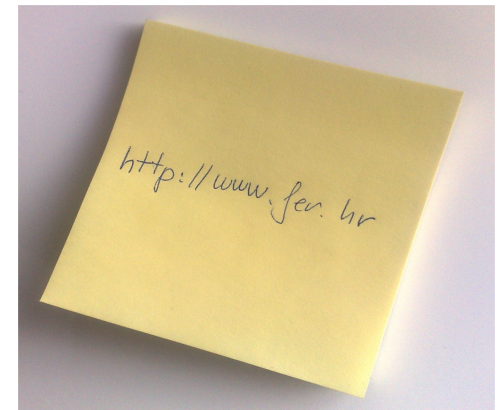
- ◇ predstavljen nizom znakova iz vrlo ograničenog skupa znakova, može postojati u različitim oblicima
- ◇ omogućuje razlikovanje pojedinog resursa naspram svih drugih resursa
- ◇ resurs je sve što se može identificirati URI-jem 😊, može i ne mora biti dohvatljiv

# Primjeri URI-a

---



Preuzeto iz <http://www.ietf.org/rfc/rfc3986.txt>



`ftp://ftp.is.co.za/rfc/rfc1808.txt`

`http://www.ietf.org/rfc/rfc2396.txt`

`ldap://[2001:db8::7]/c=GB?objectClass?one`

`mailto:John.Doe@example.com`

`news:comp.infosystems.www.servers.unix`

`tel:+1-816-555-1212`

`telnet://192.0.2.16:80/`

`urn:oasis:names:specification:docbook:dtd:xml:4.1.2`



# Uloga URI-a

---

- Definira okvir (sintaksna pravila, strukturu i mehanizme obrade) oblikovanja i korištenja raznih vrsta identifikatora resursa
- Način ostvarenja identifikacije je prepušten pojedinoj **shemi**
  - primjeri shema: http, ftp, ssh, svn, news, mailto ...
- **URI parser** rastavlja URI na osnovne komponente
  - obrada sadržaja pojedinih komponentata je specifična za korištenu shemu

# Operacije nad URI-om

---

- Ne garantira pristup identificiranom resursu
- Ne definira akcije nad resursom
- Dvije operacije definirane nad URI-jem: **rezolucija i dereferenciranje**
  - Rezolucija: proces određivanja mehanizma pristupa resursu
    - <http://www.fer.hr/predmet/otvrac>
      - rezolucija određuje korištenje protokola http za pristup resursu
  - Dereferenciranje: korištenje mehanizma pristupa za obavljanje akcije nad resursom identificiranim URI-jem

# URL – Uniform Resource Locator

---

- URI koji sadrži informaciju o lokaciji resursa
- Ne garantira jedinstvenost i trajnost resursa
  - isti dokument može postojati na više lokacija
  - dokument može biti premješten s određene lokacije ili može prestati postojati
- U formalnim specifikacijama više nije u upotrebi
  - uglavnom se koristi u stručnoj terminologiji kao sinonim za adresu html stranice

**`http://www.fer.hr/predmet/otvrac_a`**

# URN – Uniform Resource Name

---

- Podskup URI-ja koji sadrže ime resursa, garantira jedinstvenost i trajnost identifikacije

**<URN> ::= "urn:" <NID> ":" <NSS>**

- **NID** – prostor imena (namespace identifier)
- **NSS** – namespace specific string

**urn:isbn:0-395-36341-1**

**urn:ietf:rfc:3187**

**urn:isan:0000-0000-9E59-0000-O-0000-0000-2**

# Korišteni znakovi

---

- URI koristi ograničeni skup US-ASCII znakova
  - [a-zA-Z0-9], neke posebne znakove
- Rezervirani znakovi
  - odvajaju generičke komponente (gen-) i podkomponente URI-a (sub-)
  - ne smiju biti izravno korišteni u sadržaju URI-a

**gen-delims** = : / ? # [ ] @  
**sub-delims** = ! \$ & ' ( ) \* + , ; =

[http://www.fer.hr/predmet/otvrac\\_a](http://www.fer.hr/predmet/otvrac_a)      <mailto:dekan@fer.hr>

# Postotni oblik zapisa znakova

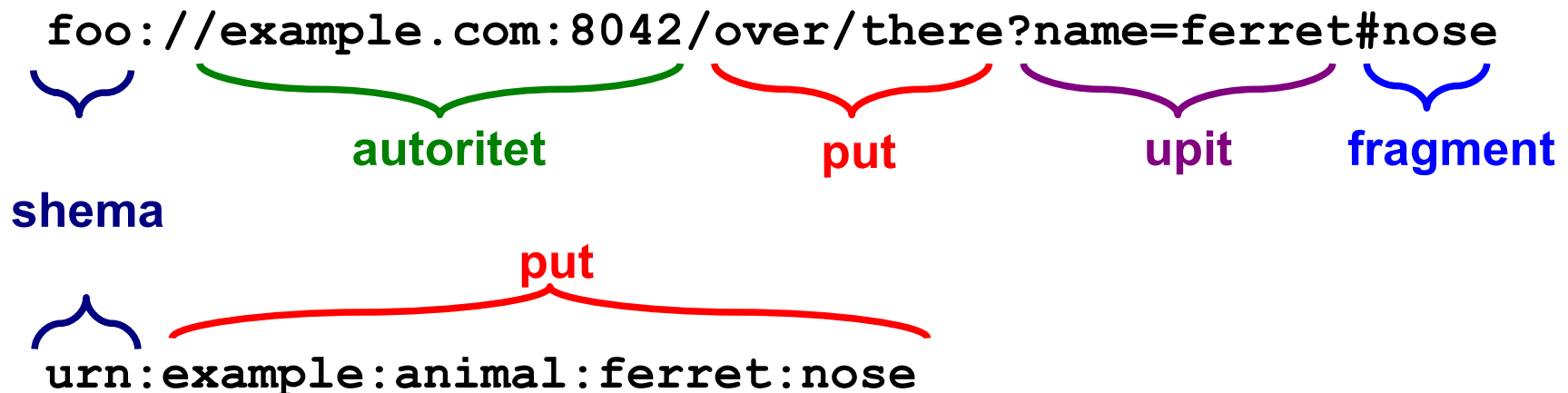
---

- Način korištenja zabranjenih znakova
  - rezervirani znakovi, razmak, ne-ASCII znakovi itd.
  - **format:** `"%" HEXDIGIT HEXDIGIT`
  - `%20` predstavlja ASCII `0x20` - " "
  - `%25` predstavlja znak `%` - mora biti kodiran ako je dio sadržaja URI-ja

`http://www.fer.hr/rasip/otvoreno%20racunarstvo.html`

# Komponente URI-a

- Sastoji se od pet osnovnih komponenata: **scheme** (schema), **autoriteta** (authority), **puta** (path), **upita** (query) i **fragmenta** (fragment)
- Svaki URI mora se sastojati barem od scheme i puta (put može biti prazan)



# Schema

---

- Određuje sintaksu i semantiku identifikatora definiranih u shemi, sadržaj, rezervirane znakove, ...
  - obavezna komponenta URI-a
  - primjeri: `ftp:` `http:` `file:` `svn:` ...
  - terminirana dvotočkom iza naziva sheme (npr. `ftp:` )
- Ime sheme u većini slučajeva “daje savjet” o protokolu za pristup resursu
  - na taj način definirana većina naziva shema
  - `http` “savjetuje” korištenje HTTP-a (tko bi rekao!?)



# Postojeće sheme

---

- Sheme su definirane u posebnim dokumentima (službene uglavnom u RFC dokumentima)
- Službene sheme registrirane kod IANA-e
  - dns, fax, file, ftp, http, https, imap, ldap, mailto, news, nfs, nntp, pop, snmp, tel, telnet, urn, ...
  - <http://www.iana.org/assignments/uri-schemes.html>
- Neslužbene sheme
  - cvs, irc, mms, notes, secondlife, skype, ssh, sftp, smb, svn, unreal, ...

# Autoritet

---

- Autoritet određuje identificirani resurs
  - analizira informacije sadržane u komponentama puta, uputa, fragmenta URI-ja, na osnovu tih informacija određuje resurs
  - neobavezna komponenta URI-ja
  - započinje sa //, a završava sa /, ?, # ili krajem URI-ja
  - pod-komponente [userinfo@] host [:port]

`http://www.fer.hr`

`svn://mferkovic@svn.fer.hr`

`http://www.fer.hr:8080`

`http://161.53.67.80`

`ftp://mferkovic:enter@ftp.fer.hr`

`ssh://mferkov@161.53.67.13`

`ssh://mferkovic@pinus.cc.fer.hr:8080`

# Komponente autoriteta

---

- **userinfo**
  - opcionalna komponenta, sadrži podatke o korisniku i/ili načinima pristupa resursu
- **host**
  - obavezna komponenta, sadrži DNS ime ili IP adresu računala na kojem se nalazi autoritet
- **port**
  - opcionalna komponenta, sadrži TCP/UDP port na kojem se kontaktira autoritet, standardni portovi ne moraju se navoditi (npr. za http standardni port je 80, ...)

- Identificira resurs u dosegu sheme
  - obavezna komponenta URI-a, no može biti prazan
  - ako u URI postoji autoritet, put započinje s prvim / nakon oznake početka autoriteta (//),
  - ako autoritet ne postoji, put ne smije započeti s //
  - komponenta puta završava s ?, # ili krajem URI-a
- Put može biti ravan ili hijerarhijski
  - ravan put – ne može se razložiti na komponente
  - hijerarhijski put - segmenti puta odvojeni / , mogu sadržavati . i ..

# Primjeri putova

---

`http://www.fer.hr`

(prazan put)

`mailto:John.Doe@example.com`

`tel:+1-816-555-1212`

`telnet://192.0.2.16:80/`

(prazan put)

`file:///C:/Temp/todo.txt`



*nema autoriteta, aplikacija sama mora brinuti o tumačenju puta `c:/Temp/todo.txt`*

`http://www.ietf.org/rfc/rfc2396.txt`



*autoritet brine o tumačenju puta `/rfc/rfc2396.txt`*

# Upit

---

- Sadrži “ravne” podatke koji, zajedno s putem, identificiraju resurs
  - opcionalna komponenta URI-a
  - započinje s prvim ?, završava s # ili krajem URI-a
- Upotreba npr. kod metode *get* u HTTP-u
  - upit u obliku niza parova ime=vrijednost, parovi povezani sa znakom &

`http://www.google.com/search?q=FER&ie=utf-8&oe=utf-8&aq=t&rls=org.mozilla:en-US:official&client=firefox-a`

`ldap://ldap.fer.hr/c=GB?objectClass?one`

# Fragment

---

- Identificira sekundarni resurs unutar primarnog
  - sadrži dodatnu informaciju o sekundarnom resursu
  - započinje s #, a završava s krajem URI-ja
  - opcionalna komponenta URI-ja
  - primjer sidra u HTML dokumentu

`http://gbiv.com/protocols/uri/rfc/rfc3986.html#reference-resolution`

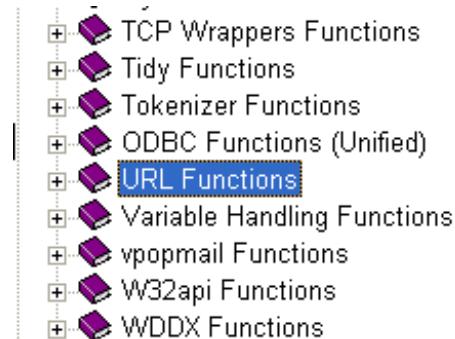
# PHP i URL

---



`parseURL()`      *rastavlja URL na komponente  
(shema, autoritet ...)*

`urlencode()` ,  
`urldecode()` ,  
`rawurlencode()` ,      *postotno kodiranje i  
dekodiranje znakova*  
`rawurldecode()`



`base64_encode`      *kodiranje i dekodiranje*  
`base64_decode`      *podataka u formatu base64*



# URI reference

---

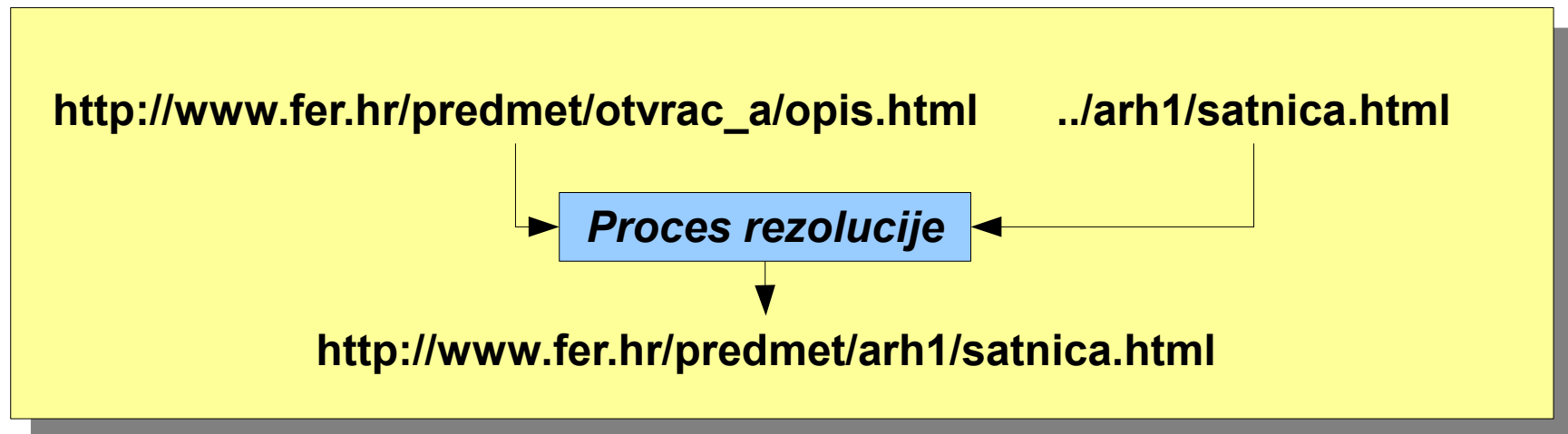
- URI referenca “pokazuje” na resurs
- Vrste URI referenci: **apsolutne** i **relativne**
- Apsolutna referenca formirana po svim pravilima tvorbe URI-a
  - posjeduje schema i path elemente, ...
- Relativna referenca ne posjeduje schema element
  - za određivanje resursa mora se nadopuniti na apsolutnu referencu
  - nadopunjavanje - proces **rezolucije**

# URI reference

---

- URI referenca
  - niz znakova koji reprezentira URI (tj. ciljni resurs)
  - postoji u kontekstu nekog resursa i baznog URI-ja (bazni URI je najčešće URI resursa u kojem se referenca nalazi)
- **Apsolutna URI referenca**
  - formirana po svim pravilima tvorbe URI-ja
- **Relativna URI referenca**
  - ne posjeduje schema element
  - ne mora posjedovati autoritet, apsolutni put, ...
  - za određivanje ciljnog resursa mora se nadopuniti na apsolutnu referencu tijekom procesa **rezolucije**

# Rezolucija URI reference



- Relativna URI referenca je  $\Delta$  koja se primjenjuje na bazni URI
- Bazni URI mora biti apsolutan (potpun)!
- Određivanje baznog URI-a
  - npr. URL HTML stranice unutar koje se nalaze relativne veze na druge stranice

# Primjeri rezolucije



[http://www.fer.hr/predmet/otvrac\\_a/opis.html](http://www.fer.hr/predmet/otvrac_a/opis.html)      [satnica.html](#)



[http://www.fer.hr/predmet/otvrac\\_a/satnica.html](http://www.fer.hr/predmet/otvrac_a/satnica.html)

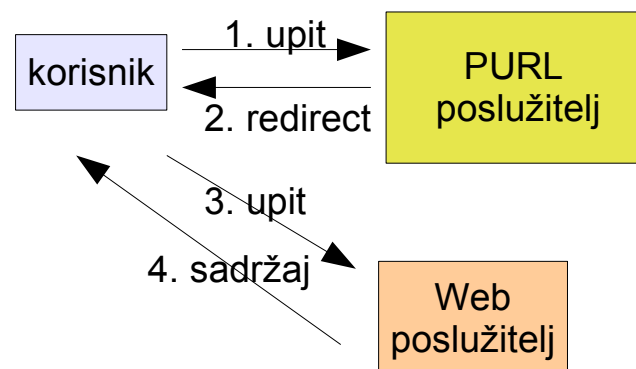
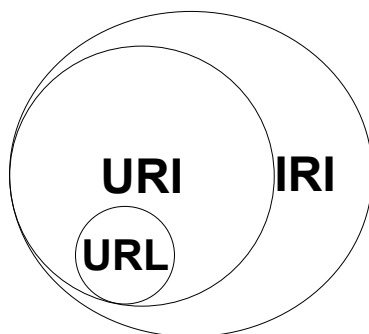
[http://www.fer.hr/predmet/otvrac\\_a/opis.html](http://www.fer.hr/predmet/otvrac_a/opis.html)      [/predmet/arh1/satnica.html](#)



<http://www.fer.hr/predmet/arh1/satnica.html>

# Daljnji razvoj URI-ja

- Internationalised Resource Identifiers (IRI)
  - <http://tools.ietf.org/html/rfc3987>
  - dozvoljeni znakovi iz Unicode/ISO 10646
- Persistent Uniform Resource Locator (PURL)
  - problem trajnosti URL-ova
  - posrednička usluga preslikavanja PURL u trenutno važeći URL, privremeno rješene do širkog uvođenja URN



# Otvoreno računarstvo

Multipurpose Internet Mail Extension (MIME)

# MIME

---

- Multipurpose Internet Mail Extension
  - Zamišljen kao proširenje e-pošte
  - Prerastao okvire e-pošte i prihvaćen u drugim protokolima (npr. HTTP, ...)
  - Opisan u RFC 2045, 2046, 2047, 4288, 4289, 2077.
- Proširuje poruke - osim 7-bitnog ASCII teksta sadrže
  - Tekst u drugim kodiranjima osim ASCII-ja
  - Privitke u drugim oblicima osim teksta (slike, dokumente, ...)
  - Višedjelne poruke (*multi-part*)
  - Podatke u zaglavljima u drugim kodnim stranicama (ne-ASCII)

# MIME

---



- Norma za
  - Označavanje tipa podataka
  - Označavanje načina kodiranja podataka
  - Strukturiranje poruka
- Primjena
  - Elektronička pošta
  - Internetski protokoli (HTTP, ...)
- Resursi s dodanim oznakama tipa i načina kodiranja podataka prema MIME normi nazivaju se **MIME objektima**





# Oznake tipa

---

- Razvijena normirana struktura tipova podataka
  - Označuje se parom *tip /podtip (type/subtype)*
    - application – razni podaci
      - /javascript
      - /octet-stream – proizvoljni podaci (npr. izvršne datoteke)
      - /xhtml+xml – XHTML datoteke
      - /zip – ZIP archive
    - audio – zvučne datoteke
      - /mpeg – mp3 ili druge podvrste MPEG normi
    - image – slikovne datoteke
      - /gif, /jpeg, /png
    - text – tekstualne (čitljive) datoteke
      - /html, /plain

# Oznake tipa

---

- Podrška nenormiranim tipovima
  - Oznaka: počinje s x-
- Podrška tipovima pod kontrolom proizvođača
  - Oznaka: *vnd*
- Oznaka tipa zapisana u zaglavlju poruke
  - Content-type: text/plain
- Poruke tipa multipart sadrže stablastu strukturu podataka
  - Poruka označena tipom multipart (najčešće /mixed)
  - Svaki dio poruke označen dodatnim zaglavljem tipa

# MIME – kodiranje

---

- Kodiranje riječi (*encoded-word*)
  - Za zaglavlja
  - Oblik =?kodna stranica?kodiranje?kodirani tekst?=
    - Kodna stranica – bilo koje normirano kodiranje znakova
    - Kodiranje – Q (*quoted printable*) ili B (*Base64*)
    - Npr. =?utf-8?Q?=Otvoreno\_ra=C4=8Dunarstvo?=
- Kodiranje poruke
  - *Quoted printable*
    - Svaki znak koji nije *običan* ASCII znak (*printable*) kodira se sa =HH (HH - heksadekadski zapis okteta znaka)
  - *Base64*
    - Znakovi se pretvaraju u grupe po 6 binarnih znamenki i svaka grupa zamjenjuje za znakom A-Z,a-z,0-9,+,/ (ukupno 64 znaka)
  - Linije duže od 76 znakova prekidaju se znakom =

# Višedjelne poruke



- Dijelovi poruke odijeljeni *granicom*

*MIME-version: 1.0*

*Content-type: multipart/mixed; boundary="granica"*

*Ovo je multipart poruka. Oznaka granice razdvaja dijelove..*

*--granica*

*Content-type: text/plain*

*Ovo je tekstualno tijelo poruke.*

*--granica*

*Content-type: text/html*

*Content-transfer-encoding: quoted-printable*

*<html>*

*<body>*

*<h1>MIME poruke</h1>*

*Ovo je tijelo poruke pisano <b>HTML-om</b>.*

*</body>*

*</html>*

*--granica*

*Content-type: image/jpeg*

*Content-transfer-encoding: base64*

*(... kodirana JPEG slika... )*

*--granica--*

