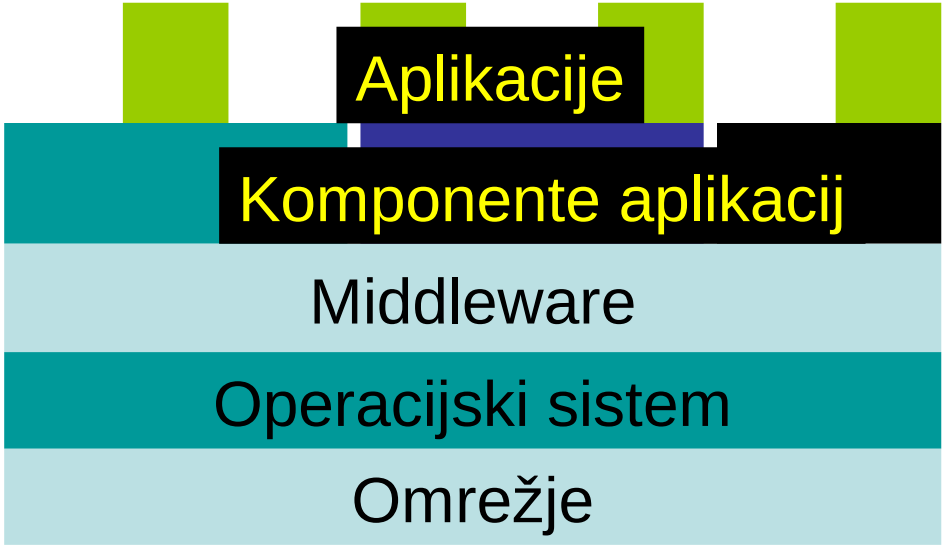
Porazdeljene aplikacije

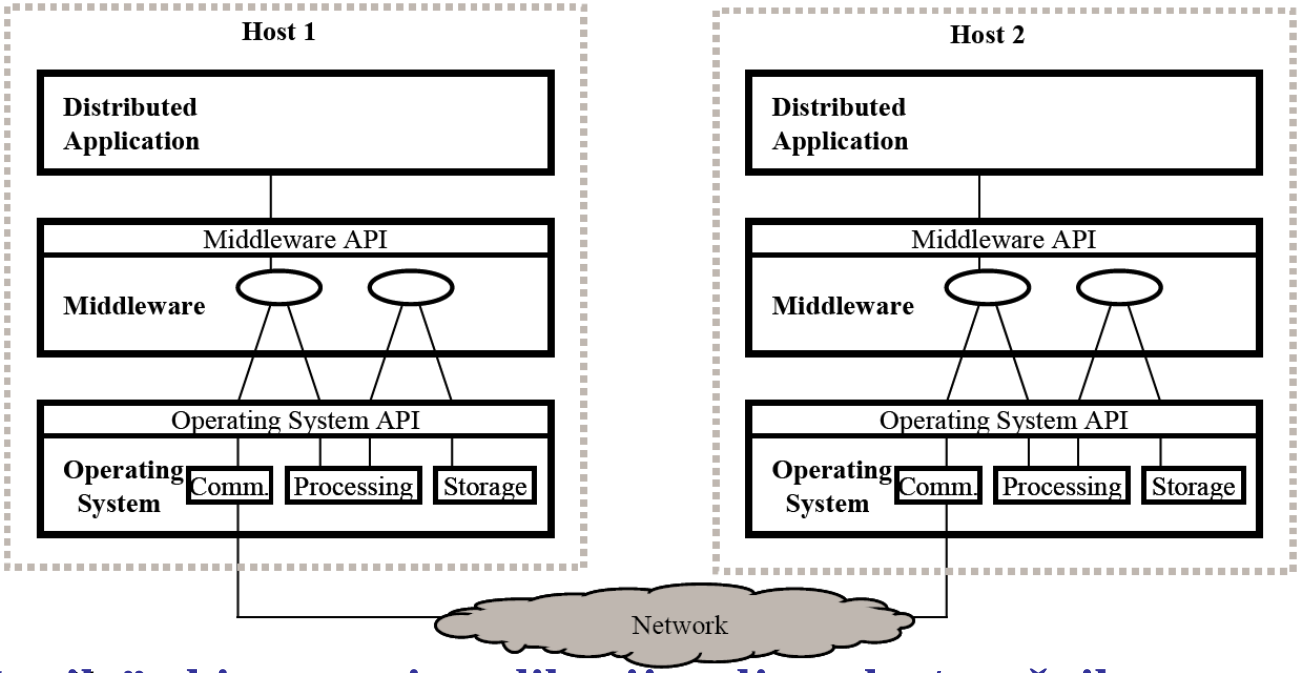
Programska infrastruktura danes

- Računalniška omrežja so prinesla poudarek na:
  - Prenosljivosti: aplikacije tečejo na različnih platformah
  - Interoperabilnosti: Različni deli aplikacij morajo med seboj sodelovati (izkoriščati možnost mrež)
- Plasti v modernih aplikacijah zagotavljajo strukturirano in enostavno doseganje teh ciljev



Middleware

- Programska oprema, ki omogoča aplikaciji ali komponentam aplikacije povezovanje in izmenjavo podatkov z drugimi aplikacijami/komponentami
- Programer in uporabnik ne potrebujeta poznavanja internega procesiranja za doseganje teh ciljev
- Programer uporablja visokonivojske API (Application program interface)



“*Lepilo*” , ki povezuje aplikacije odjemalec/strežnik

Middleware

- Plast med operacijskim sistemom in aplikacijo
  - Skriva heterogenost
  - Nudi splošne skupne storitve
  - Poveča nivo abstrakcije
- Po svoji naravi ne teče le na eni platformi in ni vezan na opremo
  - Microsoft je (velikokrat) izjema

Cilji “middleware”

- Skrivanje heterogenosti in prenosljivost
- Lokacijska transparentnost:
  - aplikacija najde storitev na omrežju, ne glede na to kje se le-ta nahaja
- Neodvisnost od omrežne arhitekture
- Zanesljivost
- Skalabilnost

- V splošnem so cilji podobni storitvam, ki jih nudijo porazdeljeni OS

Nekaj kategorij middleware

- Obdelava transakcij
  - Poenostavljanje koordinacije komplementarnih upravnikov virov
- Middleware, usmerjen v obvestila
  - Podpira obvestila in uvrščanje pri upravnikih virov, ki niso simultano na voljo
- Upravljanje s porazdeljenimi objekti
  - Podpora aplikacijam, ki so porazdeljene vzdolž heterogenih platform in organizacij
- Mobilna koda
  - Omogoča aplikacijam, da jih prestavljamo in izvajamo na heterogenih platformah
  - Brez predhodne namestitve programov

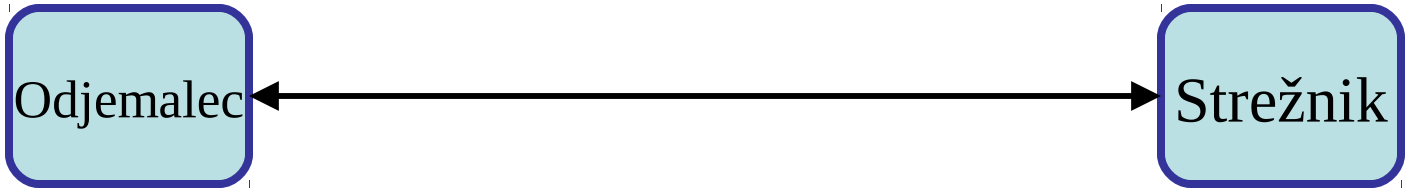
Nekaj tipov middleware

- RPC – Remote Procedure Calls (RPC)
  - aplikacije kličejo procedure, ki tečejo na oddaljenih računalnikih
  - sinhrono ali asinhrono
- Message-Oriented Middleware (MOM)
  - asinhroni klici med aplikacijami s pomočjo obvestil v vrstah
- Object Request Broker (ORB)
  - obektno usmerjena komunikacija med porazdeljenimi aplikacijami (namesto golih klicev metod kot pri RPC, se tu lahko prenašajo celotni objekti)
- Transaction Processing Monitors
  - ogrodje za upravljanje s (porazdeljenimi) transakcijami
- SQL-oriented Data Access
  - middleware med aplikacijami in strežniki podatkovnih baz

Klic oddaljene procedure (RPC)

- Porazdeljen sistem tipično vsebuje številne porazdeljene komponente, ki morajo med seboj interaktirati na transparenten način.
- Mehanizem klica oddaljene procedure (*Remote Procedure Call*, RPC) omogoča običajno proceduralno programiranje, pri čemer za klice oddaljenih procedur uporabljamo model klica lokalne procedure (local procedure call, LPC).
  - programer naj bi se čimmanj zavedal, da gre za klic procedure, ki ni lokalna
- Bistvena paradigma je pri tem model odjemalec-strežnik
  - odjemalec rabi določeno storitev
  - pokliče procedura na strežniku, ki nudi to storitev
  - procedura vrne podatke

Interoperabilnost



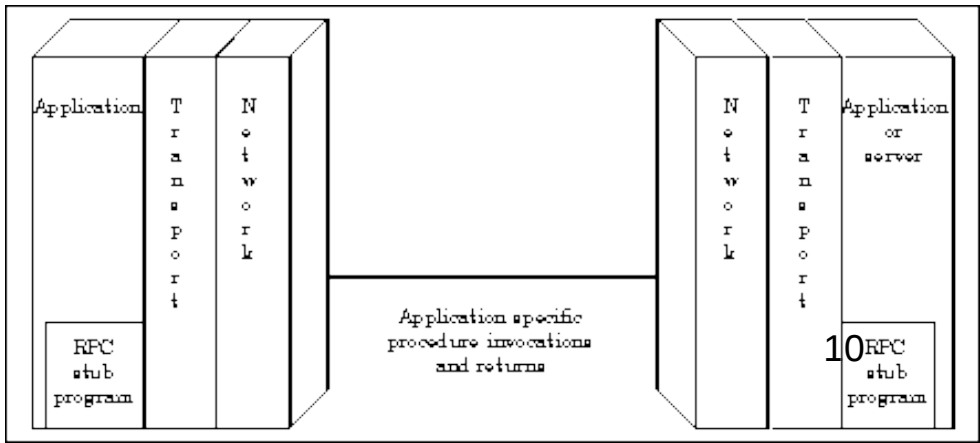
Objekt odjemalca lahko kliče procedure na strežniku, čeprav tečeta na različnih platformah in sta pisana v različnih jezikih

Interoperabilnost zahteva:

- Skupne podatkovne strukture
- Skupno interpretacijo podatkov
- Sporazum o protokolih

Klic oddaljene procedure (RPC)

- Značilnosti RPC kot kominikacijskega mehanizma:
  - Podatke prenašamo kot komunikacijske parametre.
  - Vhodni parametri morajo ubogati semantiko klica po vrednosti.
  - V porazdeljenem smislu nimamo globalnega konteksta.
  - Naslavljanje pomnilnika s kazalci nima smisla.
- Klici so lahko
  - sinhroni:
    - pokličemo funkcijo, počakamo da se izvrši in nadaljujemo z izvajanjem
  - asinhroni
    - pokličemo funkcijo, nadaljujemo z izvajanjem
    - ko se izvajanje funkcije na strežniku konča, nas z dogodkom obvesti, da je konec



Tipi RPC

- RPC je star protokol – prvič opisan 1976
- Analogne modernejše alternative:
  - Java RMI – Remote Method Invocation – objektna javanska alternativa RMI
  - Microsoft .NET Remoting – podobno kot RMI za .NET
  - Spletne storitve
    - moderno zasnovana, platformno neodvisna tehnologija, uporabna za klicanje oddaljenih procedur
    - največkrat temelji na HTTP protokolu za komunikacijo
    - temelji na XML za predstavitev vhodnih in izhodnih podatkov procedur

Tehnologije spletnih storitev

- SOAP (Simple Object Access Protocol)
  - protokol za izmenjavo XML sporočil preko omrežja, ponavadi preko HTTP protokola
  - največkrat uporabljen za opis parametrov klicov oddaljenih procedur (RPC), kjer odjemalec pokliče proceduro na strežniku, ta pa mu vrne rezultat
- WSDL (Web Services Description Language)
  - XML format za opisovanje spletnih storitev
  - pri RPC opisuje kakšne parametre procedura sprejema in kaj vrača
- UDDI (Universal Description, Discovery and Integration)
  - svetovni register spletnih storitev
  - omogoča odkrivanje storitev, ki so na voljo in pridobivanje njihovih opisov WSDL, ki omogočajo uporabo teh storitev

Spletne storitve primer:

- Napišimo funkcijo, ki sešteje dve števili in vrne rezultat
- Primer v C#, zelo podobno bi lahko naredili v Javi
- WSDL opis nam poda popoln opis storitve
  - parametre in vrednosti, ki jih metoda vrne
- Ko spletno storitev uporabimo (kličemo), se parametri in rezultati preko HTTP prenesejo kot XML dokumenti!
  - Klic AddNumbers poda parametre kot:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  ... >
  <soap:Body>
    <AddNumbers xmlns="http://adder.temp/">
      <a> 3 </a>
      <b> 4 </b>

    </AddNumbers> </soap:Body> </soap:Envelope>
```

  - Rezultate dobimo kot:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance“ ... >
<soap:Body>
  <AddNumbersResponse xmlns="http://adder.temp/">
    <AddNumbersResult> 7 </AddNumbersResult>
  </AddNumbersResponse> </soap:Body> </soap:Envelope>
```

Message-Oriented Middleware (MOM)

- Asinhrona komunikacija med komponentami
  - je sicer podprta tudi z variantami RPC
- Asinhrona pomeni, da ob klicu oddaljene procedure ne čakamo na rezultat, temveč nadaljujemo z izvajanjem
  - ko se procedura na strežniku konča, nas ta z dogodkom obvesti o rezultatu
- Ob klicu lahko ni dosegljiv strežnik; ob rezultatu odjemalec
  - zahteve takrat čakajo v sporočilnih vrstah (message queues)
- Asinhrona komunikacija
  - podpira paralelno procesiranje
  - zanesljivejša (zahteve ali rezultati čakajo v vrstah)
  - ima kompleksnejšo semantiko kot sinhrona komunikacija
  - težje je obravnavati napake kot pri sinhroni komunikaciji
- Spletne storitve podpirajo oboje, sinhrono in asinhrono komunikacijo

Object request broker (ORB)

- Podpira objektno usmerjeno komunikacijo med porazdeljenimi aplikacijami
  - namesto golih klicev metod kot pri RPC, se tu lahko kličejo metode objektov in tudi prenašajo celotni objekti s serializacijo
- Poudarek na interoperabilnosti
  - Omogoča objektom iz enega računalnika klicanje metod objektov na drugem računalniku
  - Platformna in jezikovna neodvisnost
- Ni poudarka na prenosljivosti kode
- Nudijo tudi druge storitve kot npr. porazdeljene transakcije, imenike storitev ...
- Primera ORB: CORBA in DCOM
  - CORBA – Common Object Request Broker Architecture: platformno neodvisna
  - DCOM je Microsoftov standard

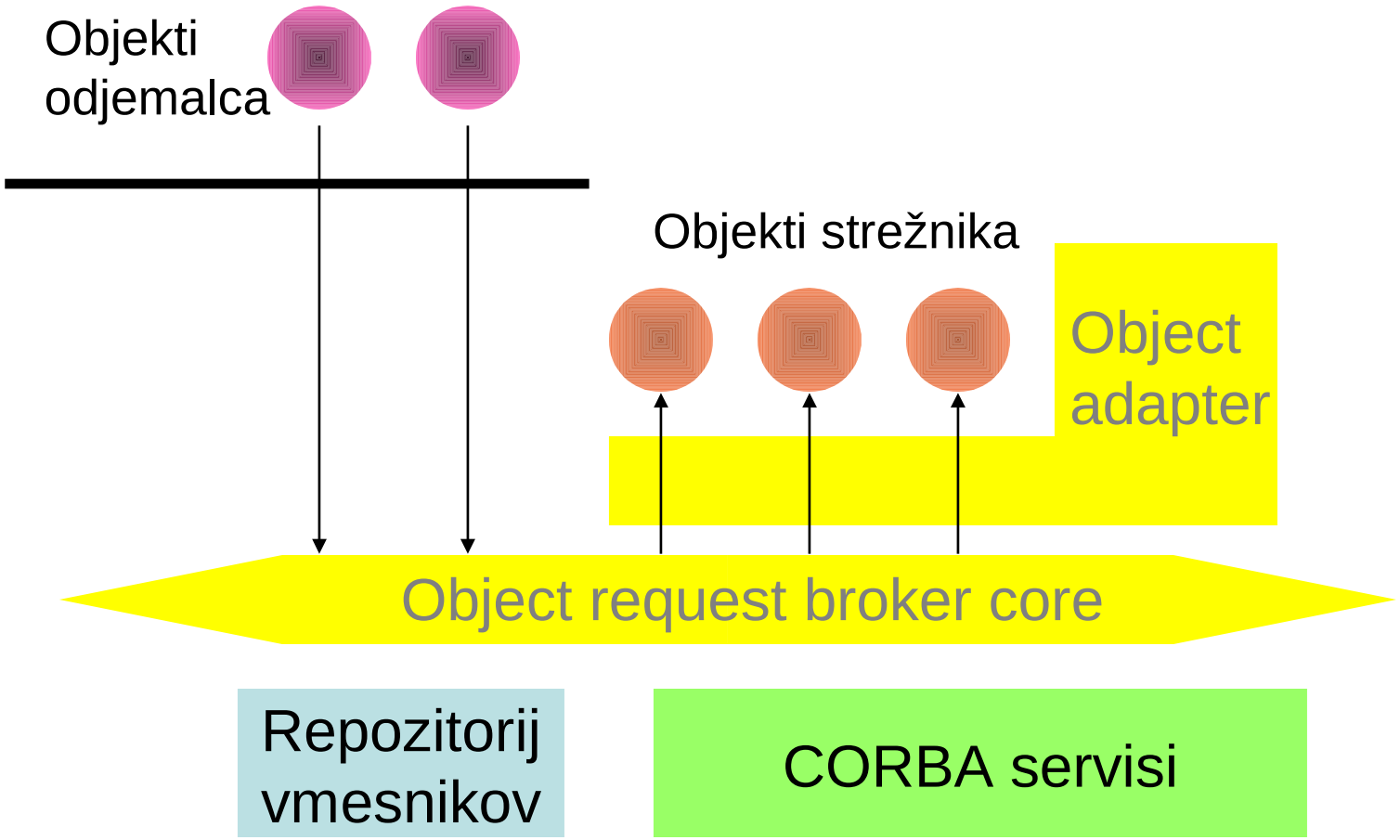
Kaj nudi CORBA?

- Interoperabilnost (ne pa prenosljivost)
- Premoščanje platform in jezikov za komunikacijo med aplikacijskimi komponentami
  - Če imamo npr. javo tudi prenosljivost objektov
- Visok nivo abstrakcije
  - enostavno klicanje metod, z oddaljenimi objekti delamo enako kot z lokalnimi
- Fleksibilnost v času izvajanja
  - Vsak objekt se sam opisuje
  - Avtomatsko odkrivanje vmesnikov, ki so na voljo
  - Dinamične podatkovne strukture in povezovanje
- Uporabni servisi
  - Imenovanje
  - Varnost
  - In še in še...

Pomembnost CORBA

- Računalništvo med podjetji (inter-enterprise computing)
  - Neodvisnost od platform in jezikov
  - e-poslovanje, upravljanje omrežij ipd.
- Zmanjšanje učinkov omrežja
  - Še ena premostitvena plast
  - Pomembnost platforme se zmanjša

Arhitektura CORBA



Protocol layer

Aplikacija, neodvisna od lokacije

Application

Object Management Group CORBA standard

Object request broker

Internet Inter-ORB Protocol (IIOP)

User datagram protocol (UDP)

Transmission control protocol (TCP)

Internet protocol (IP)

Podomrežja

Prenosljivost ni obljubljena

CORBA standard ne zagotavlja **prenosljivosti** med posameznimi ORB

Zagotavlja pa **interoperabilnost**

Kratek CORBA primer

- Razred Adder z metodo AddNumbers
- Napišemo prototip razreda z IDL (Interface Definition Language), ki je jezikovno/platformno neodvisen

```
interface Adder {
   int addNumbers(in int a, in int b) ;
};
```

- Prevedemo IDL (avtomatsko se generira nekaj razredov) in implementiramo razred v Javi, ki bo na voljo na strežniku

```
class AdderServant extends Persistent.AdderPOA {
   private ORB theOrb;
   public AdderServer(ORB orb){
           theOrb = orb;
   }
   public int addNumbers(int a, int b)
   {
      return a+b;
   }
}
```

- Zaženemo strežnik, ki bo dal ta razred na voljo za uporabo

Kratek CORBA primer (2)

- Napišemo odjemalca

```
public class Odjemalec
{
  public static void main(String args[]) {
    try {
      ORB orb = ORB.init(args, null);
      CORBA.Object obj = orb.string_to_object ("corbaname::localhost:1050#Adder");
      Adder add = AdderHelper.narrow(obj);
      int i=3, j=4, ret;
      ret = add.AddNumbers(i,j); Server:\n\t " + helloFromServer);
    } catch (Exception e) {}
  }
}
```

- Zaženemo odjemalca
  - objekt add je oddaljen objekt, ki je instanciran na strežniku
  - z njim delamo kot z lokalnim objektom
  - veliko bolj moramo pri takem programiranju paziti na morebitne napake pri izvajanju (problemi z omrežjem ali oddaljenim računalnikom ...)!

Upravljanje s porazdeljenimi objekti

- Poudarek na interoperabilnosti
  - Omogoča objektom iz enega računalnika klicanje metod objektov na drugem računalniku
  - Platformna in jezikovna neodvisnost
- CORBA v primerjavi z DCOM
- Prenosljivost ni važna

Kaj je bolj učinkovito?

- Industrijski “de facto” standard (CORBA)

  ali

- Integrirana rešitev posameznega proizvajalca (DCOM)?

CORBA v primerjavi z DCOM

CORBA	DCOM
• Integracija najboljših zamisli • Podpora več proizvajalcev • Različne platforme in jeziki	• Hitro, ni potrebnih soglasij • Ni problemov z interoperabilnostjo med proizvjalci

Dva načina interakcije aplikacij

- CORBA oziroma DCOM
- Izmenjava dokumentov (XML)

CORBA v primerjavi z XML

CORBA	XML
• Naravna razširitev OOP • Ni interpretacije dokumentov • Dobri protokoli	• Fleksibilna souporaba podatkov • Dobro za objekte, podobne dokumentom • Naravna zmožnost med platformami • Ni standardizacije protokolov

Oba potrebujeta standardizacijo interpretacije podatkov ali dokumentov

Sta Java in CORBA konkurenta ali se dopolnjujeta?

- Oba nudita interoperabilnost med različnimi platformami
- Java nudi prenosljivost
- CORBA nudi povezovanje heterogenih jezikov
- CORBA nudi veliko servisov, metapodatkov itd.
- Na kratko: med seboj sta komplementarna!
  - (marsikateri zagovornik Jave temu ugovarja)

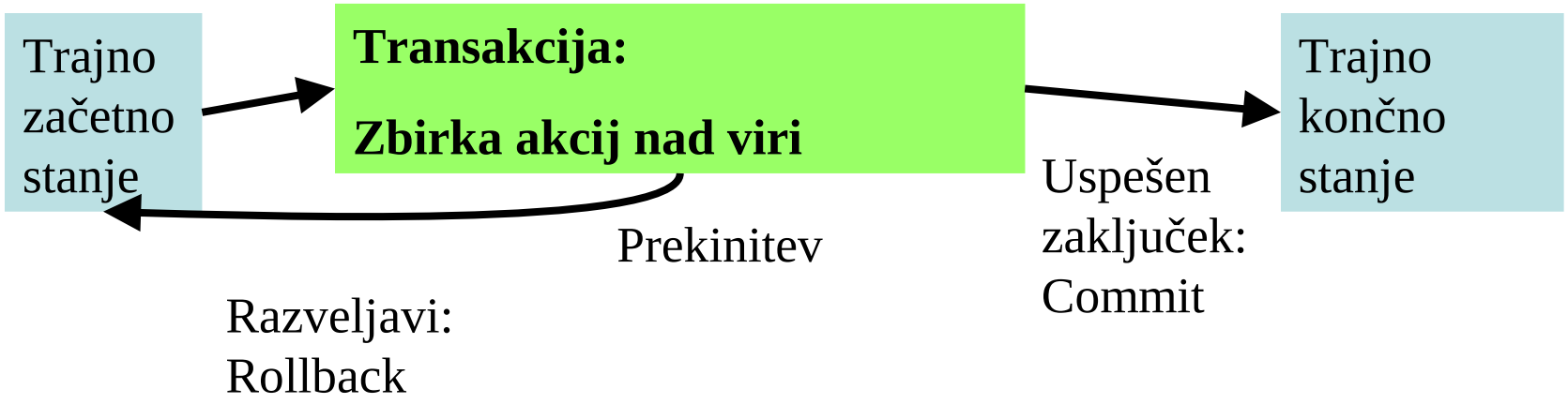
Transaction Processing Monitors

- Ogrodje za upravljanje s (porazdeljenimi transakcijami)
- Transakcija: posamezna nedeljiva enota procesiranja
  - lahko uspe v celoti ali v celoti neuspe
  - ne more ostati v nekem vmesnem stanju
- Transakcije morajo zadoščati principu ACID:
  - Atomicity: transakcija uspe v celoti ali v celoti neuspe
  - Consistency: po zaključku transakcije mora sistem ostati v konsistentnem stanju
  - Isolation: podatki, ki se spreminjajo znotraj transakcije so vidni samo aplikaciji, ki uporablja transakcijo in so izolirani od ostalih aplikacij (te morajo vedno videti konsistentno stanje)
  - Durability: ko transakcija uspe, to trajno spremeni stanje sistema (tudi če pride do sesutja sistema ...)

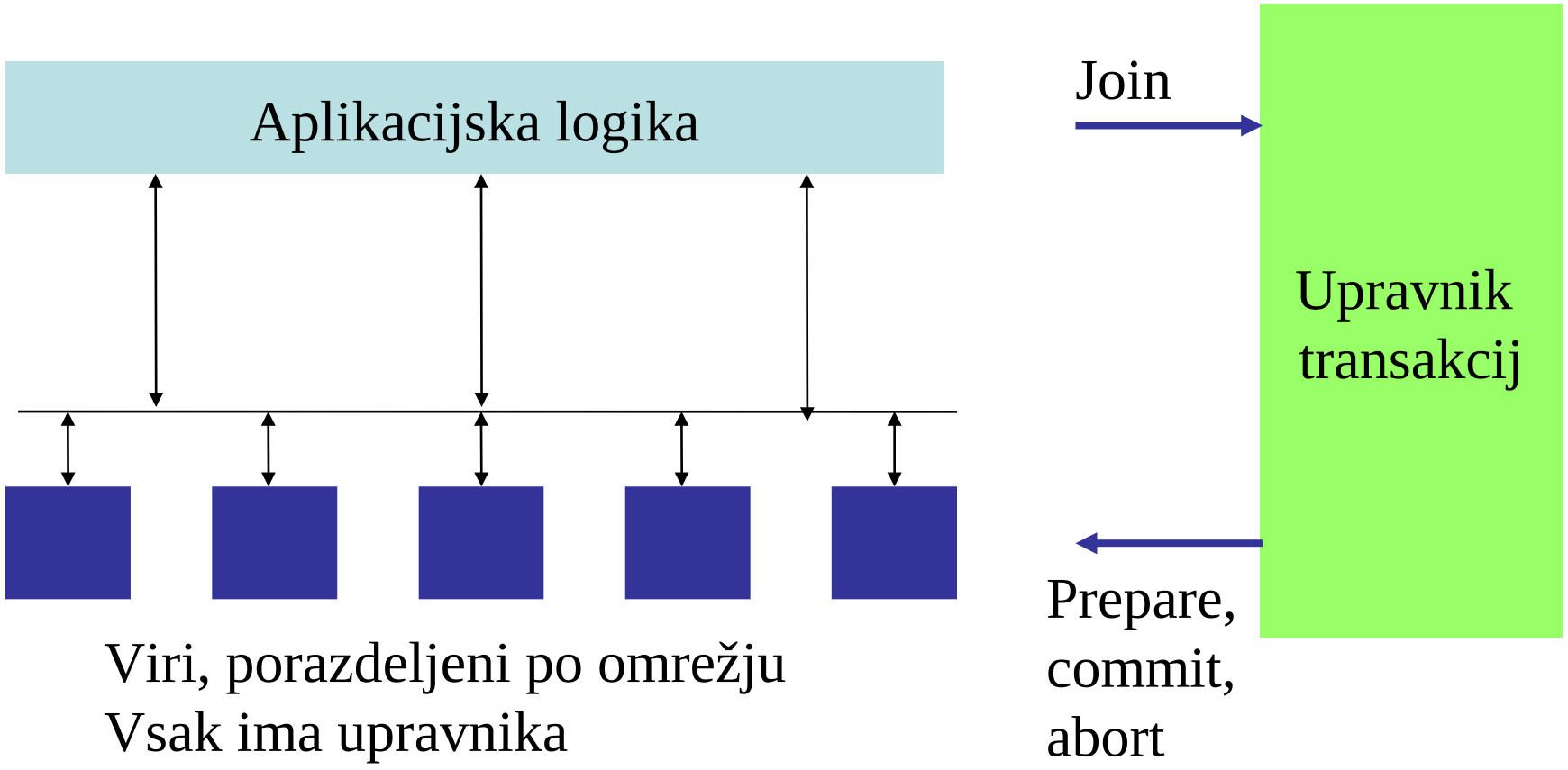
Transaction Processing Monitors

- Porazdeljene transakcije:
  - operacije, ki zajemajo več računalnikov v omrežju
  - računalniki nudijo transakcijske vire in imajo upravnike virov
  - upravnik transakcij skrbi za ustvarjanje in upravljanje s porazdeljenimi transakcijami (in za zagotavljanje principa ACID)
- Standardni algoritem za zagotavljanje pravilnosti izvedbe porazdeljene transakcije je t.i. two-phase commit (za transakcije ki ne trajajo zelo dolgo)
- Nekaj tehnologij, ki podpirajo porazdeljene transakcije:
  - Enterprise Java Beans
  - Microsoft Transaction Server
  - CORBA

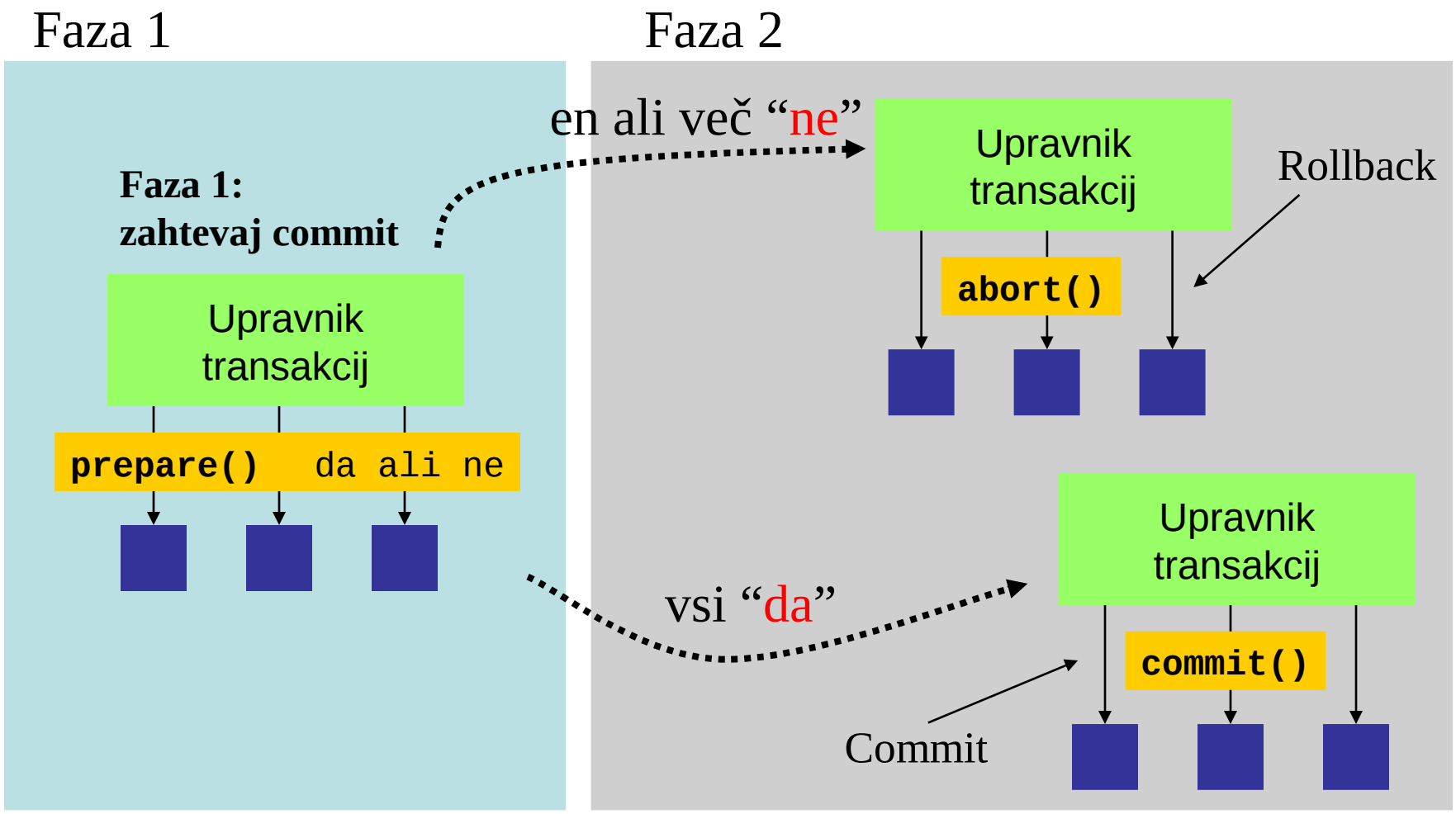
Transakcija



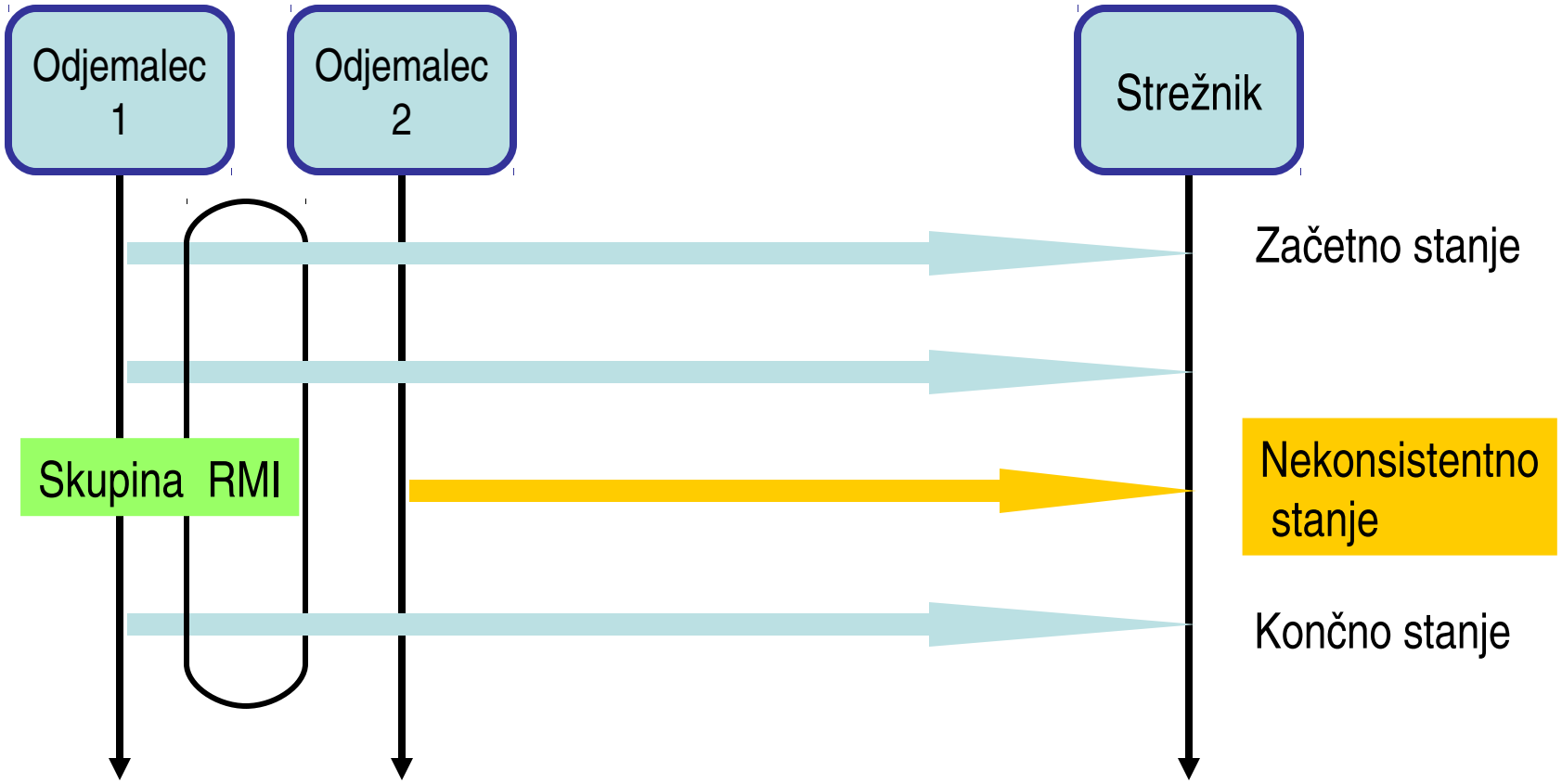
Zgradba porazdeljene transakcije



Two-phase commit

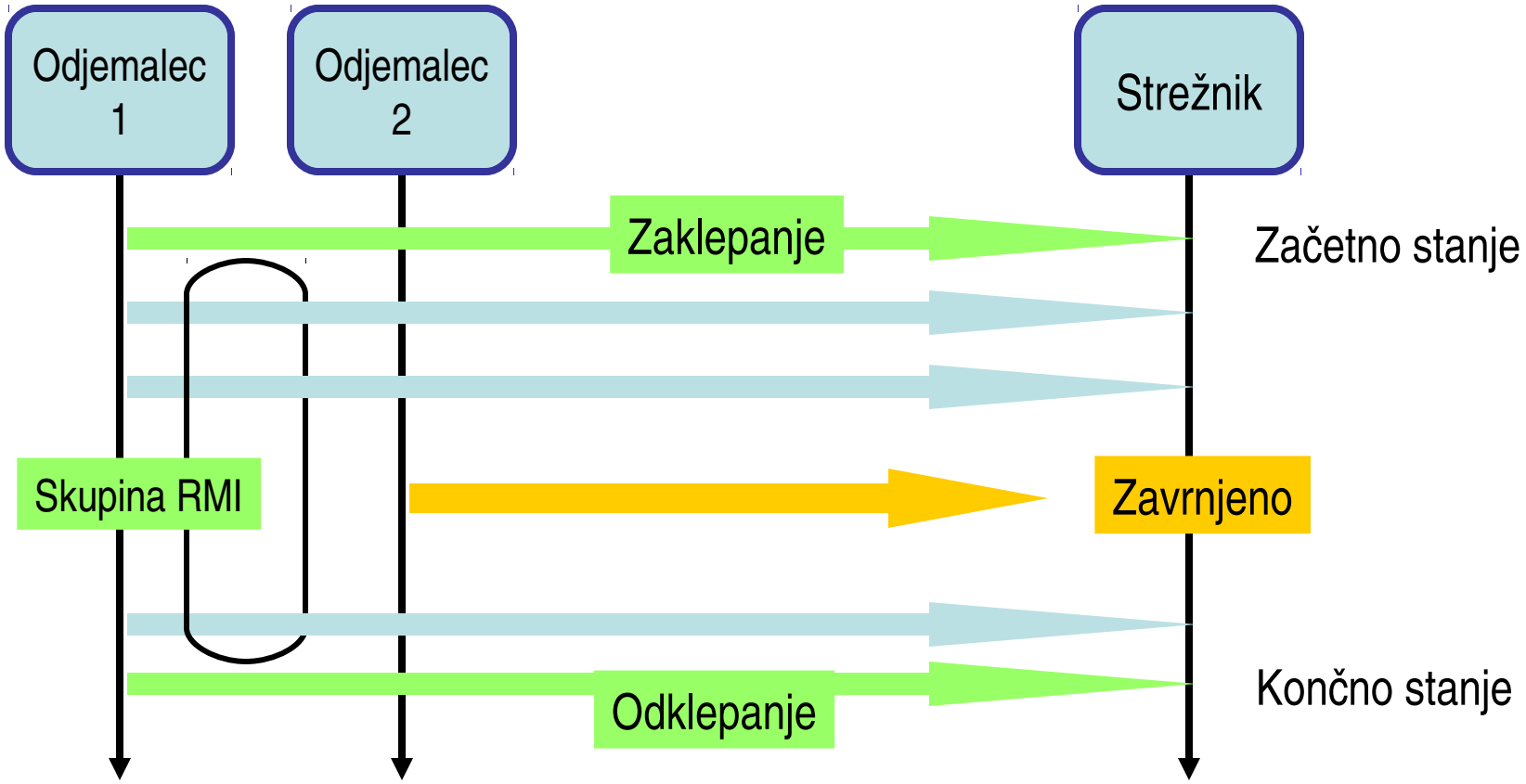


Atomarno zaporedje akcij

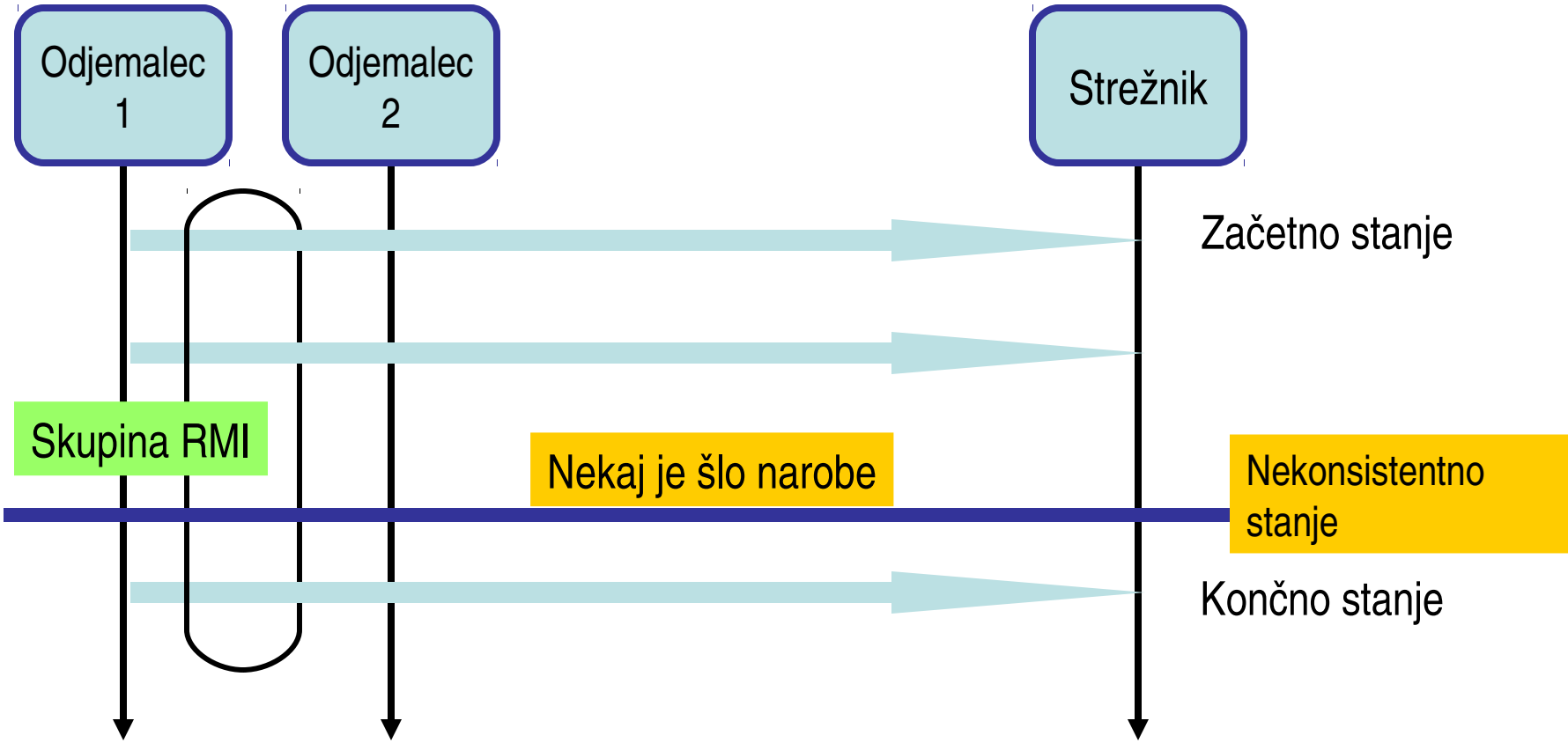


RMI == Remote Metod Invocation

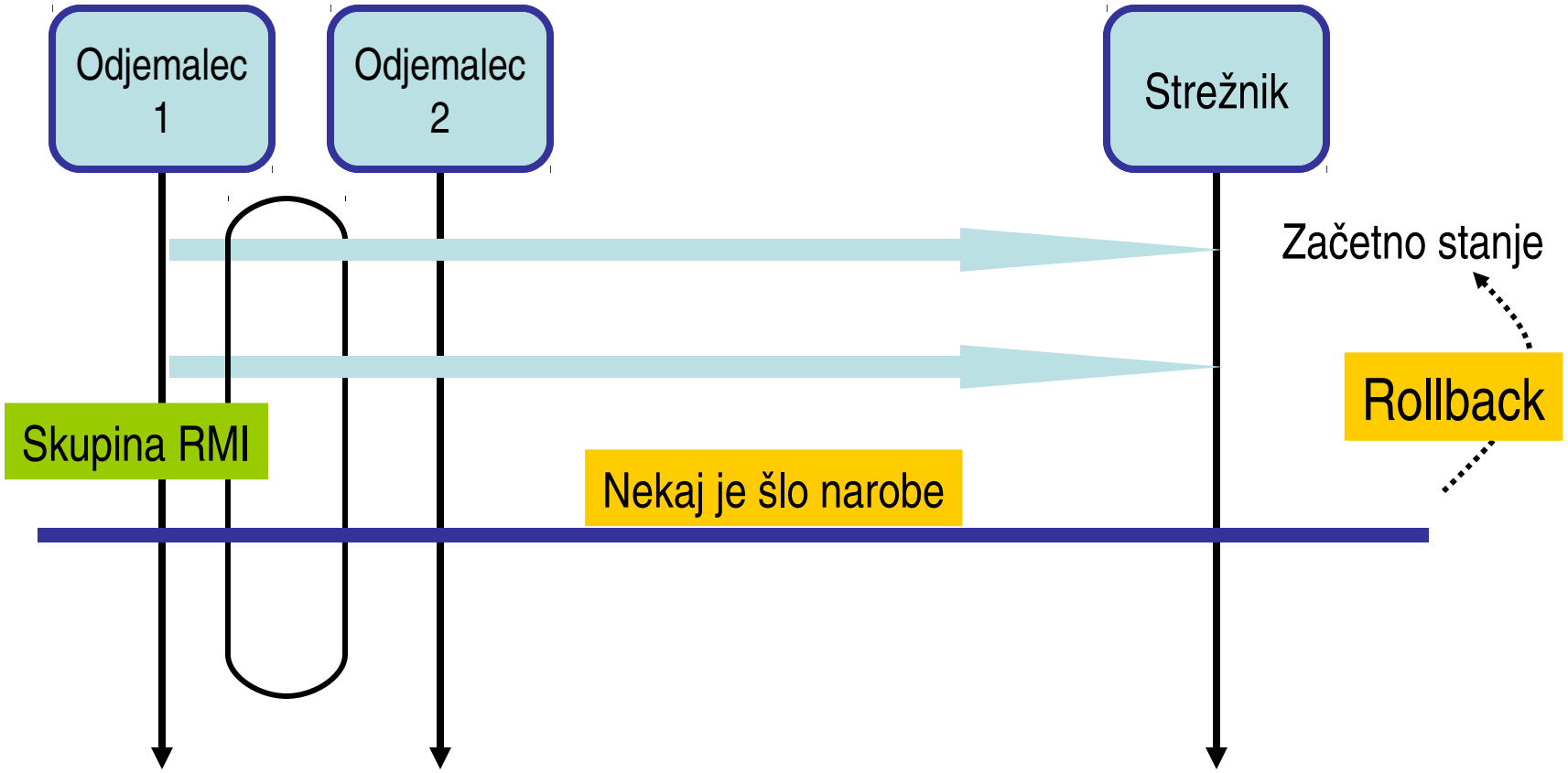
Preprečevanje konfliktov z zaklepanjem



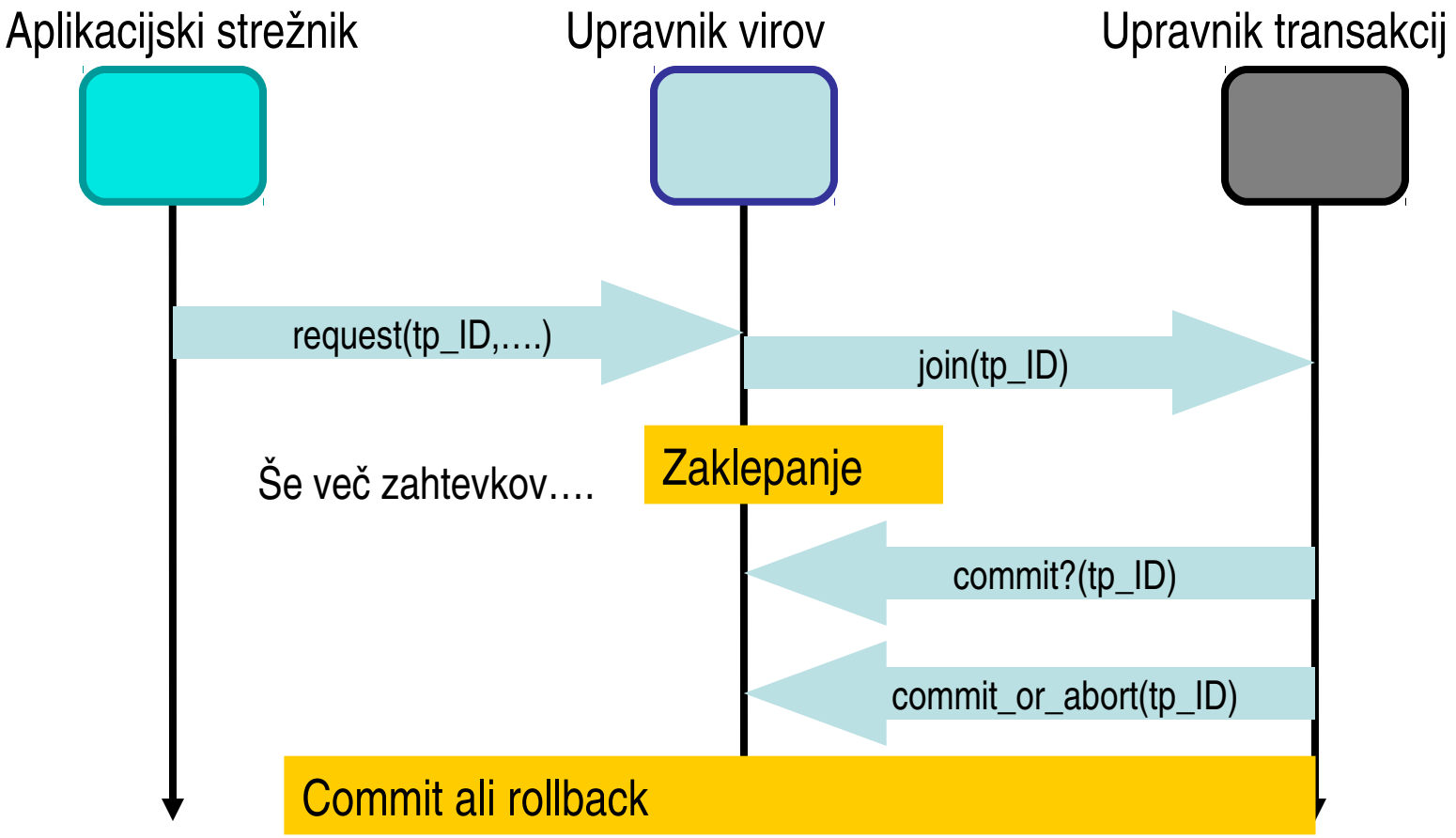
Abort (prekinitev)



Rollback (odvijanje nazaj)



Protokoli transakcij



Middleware podatkovnih baz

- ODBC – Open Database Connectivity
  - To podpira večina proizvajalcev relacijskih podatkovnih baz
- OLE-DB
  - Microsoftova izboljšava ODBC
  - ADO, ADO.NET – visokonivojska APIja za lažjo uporabo OLE-DB
- JDBC – Java Database Connectivity
  - Posebni javanski razredi, ki omogočijo povezavo apletov ali aplikacij na podatkovne baze, implementirajo ODBC

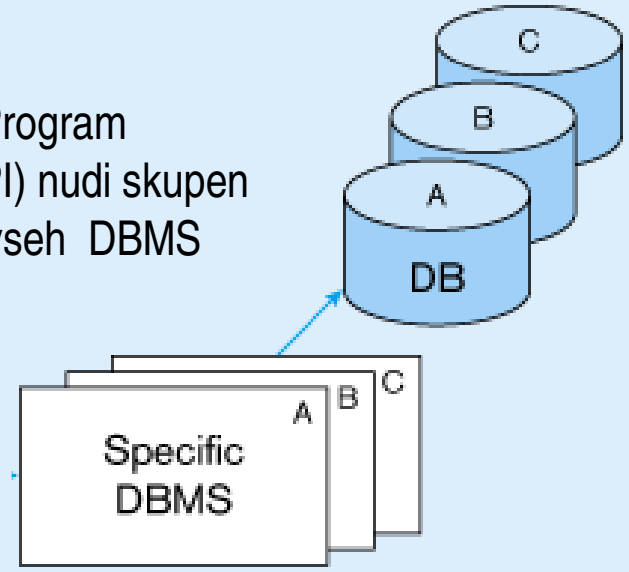
Uporaba ODBC za povezavo na zunanje podatkovne baze, pomnjene na strežniku podatkovnih baz

- Open Database Connectivity (ODBC)
  - API, ki aplikacijam nudi skupen jezik za dostop in obdelavo SQL podatkovnih baz neodvisno od konkretnega RDBMS
- Zahtevani parametri:
  - ODBC gonilnik
  - Ime strežnika
  - Ime podatkovne baze
  - Ime in geslo uporabnika
- Dodatni podatki:
  - Data source name (DSN)
  - Ime računalnika - odjemalca
  - Ime aplikacije na odjemalcu

ODBC Arhitektura

Odjemalcu ni treba poznati specifičnega DBMS

Application Program Interface (API) nudi skupen vmesnik do vseh DBMS



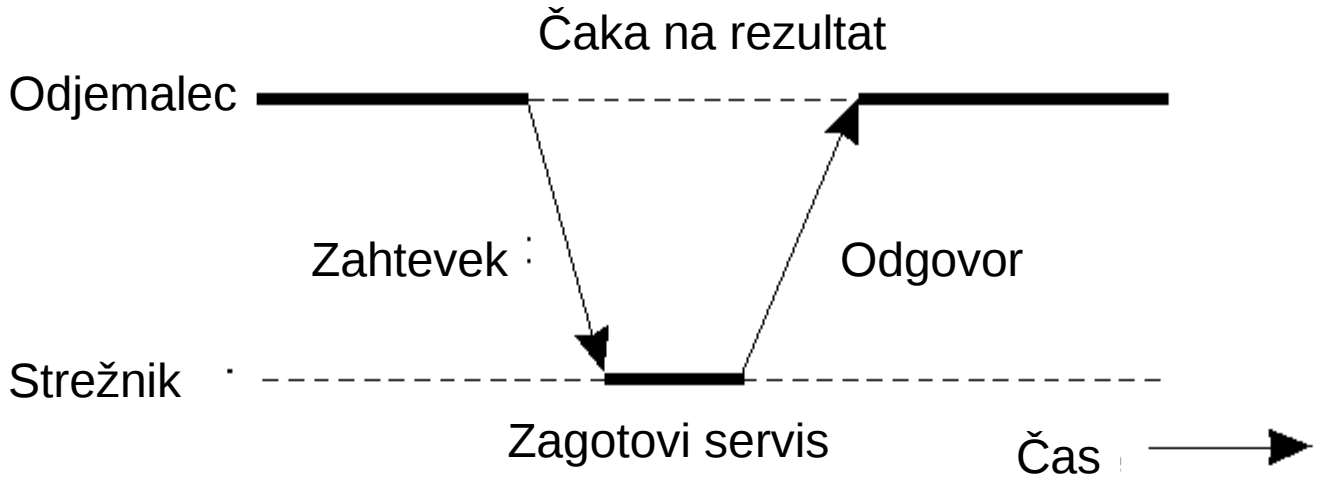
Vsak DBMS ima svoj lasten ODBC-compatibilen gonilnik

Sistemi odjemalec/strežnik (Client/Server)

- Model omreženega računalništva
- Procesi so porazdeljeni med odjemalce in strežnike
- Odjemalec – računalnik (pogosto PC), ki zahteva in uporablja nek servis
- Strežnik – Računalnik (lahko PC ali večji), ki zagotavlja servis
- Primer: strežnik podatkovne baze

Odjemalci in strežniki

- Splošen potek interakcije med odjemalcem in strežnikom.



Aplikacijska logika v sistemih C/S

(C/S = client/server)

- Predstavitvena logika
  - Vhod – tipkovnica, miška
  - Izhod – monitor/tiskalnik

Uporabniški vmesniki

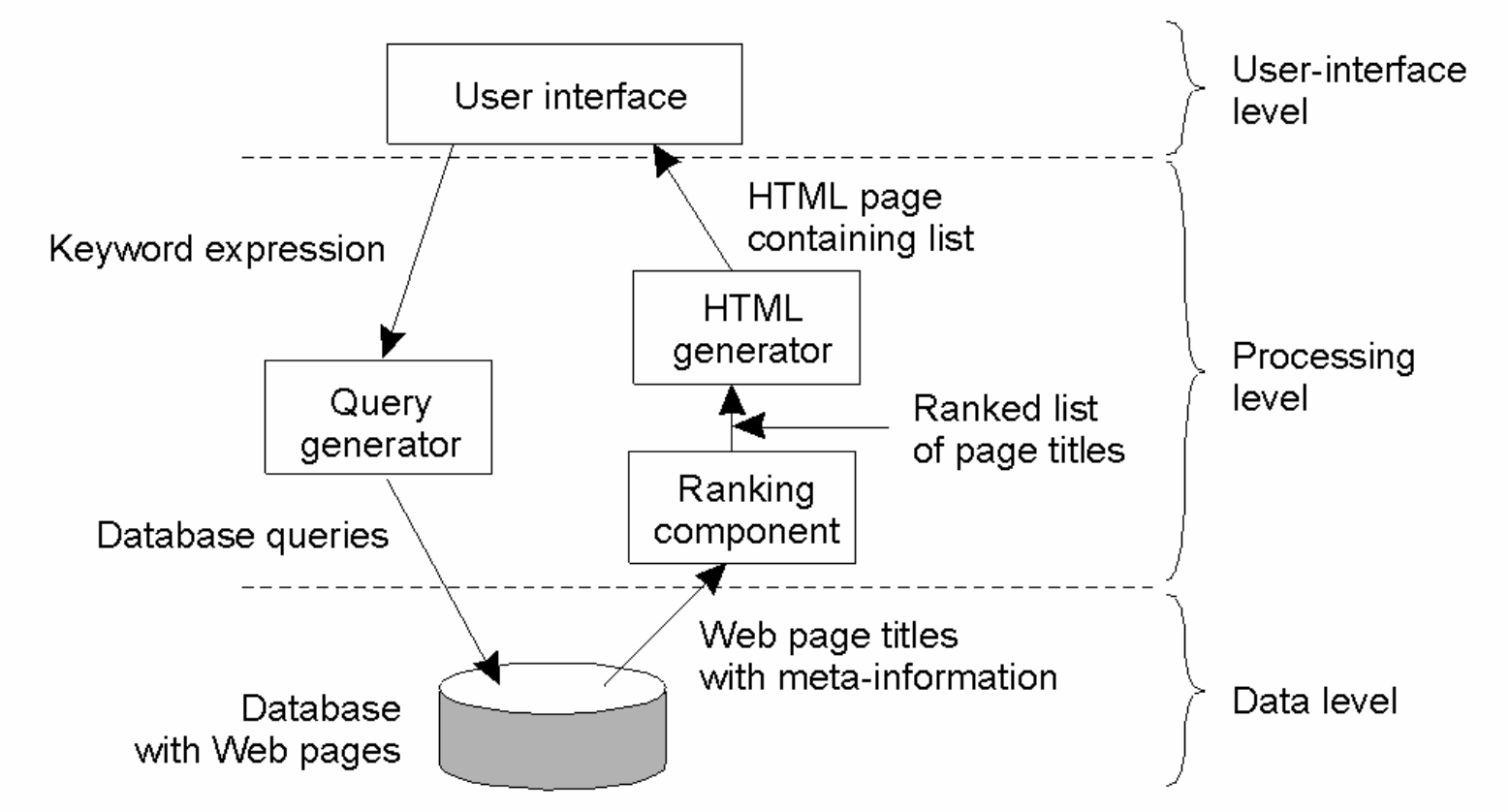
- Procesna logika
  - I/O obdelava
  - Poslovna pravila
  - Upravljanje s podatki

Procedure, funkcije, programi

- Pomnilna logika
  - Pomnenje in iskanje podatkov

DBMS aktivnosti

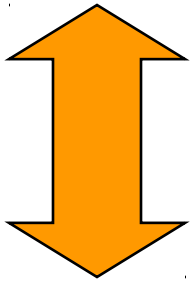
Primer: spletni iskalnik - nivoji obdelave



Arhitekture odjemalec/strežnik

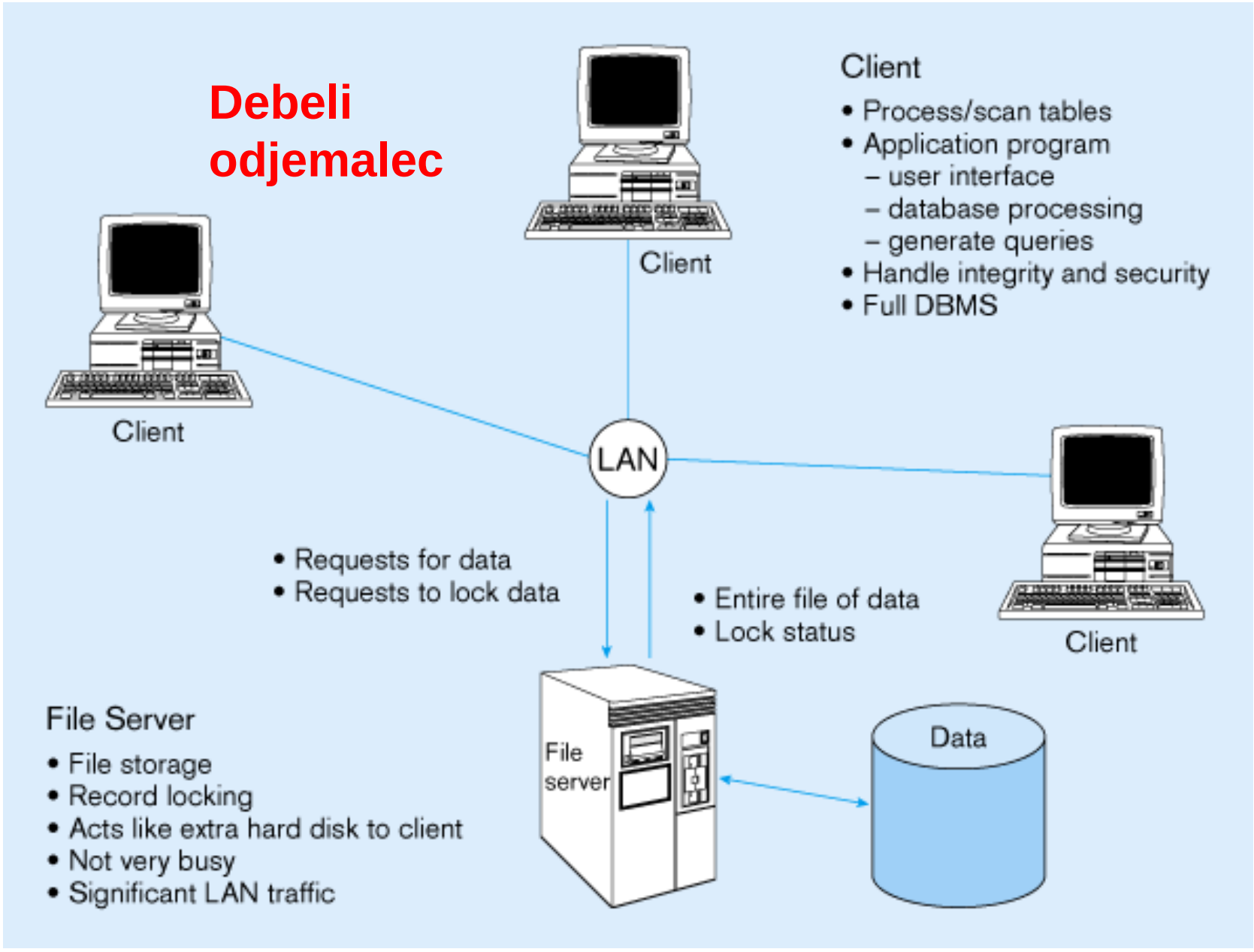
- Arhitektura datotečnega strežnika
- Arhitektura strežnika podatkovne baze
- Troslojna arhitektura

Odjemalec mora veliko računati



Odjemalec računa bolj malo

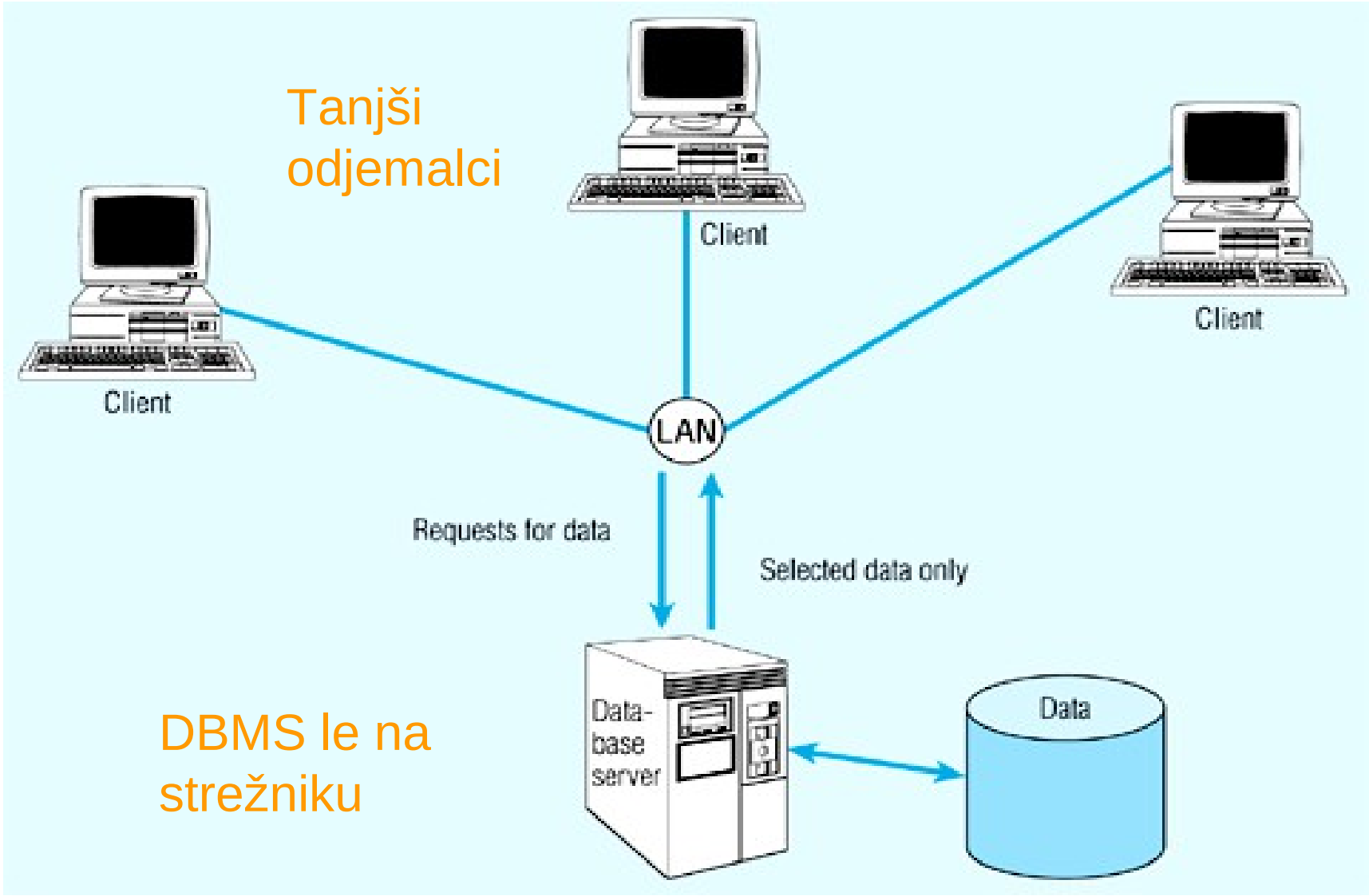
Arhitektura datotečnih strežnikov



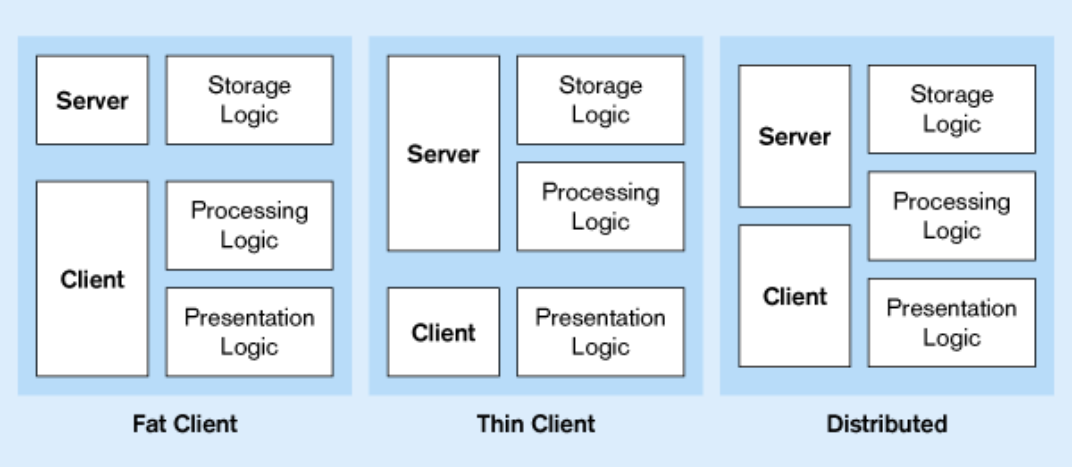
Arhitekture strežnikov podatkovnih baz

- 2-slojni pristop
- Odjemalec je odgovoren za:
  - vhodno-izhodno logiko
  - nekaj poslovne logike
- Strežnik skrbi za pomnenje podatkov in dostop do njih
  ➔ DBMS je le na strežniku
- Prednosti
  - Odjemalci so lahko bolj šibki
  - Zmanjša se promet na mreži
  - Izboljšana neokrnjenost podatkov, ker so vsi obdelani centralno
  - **Pomnjene procedure** ➔ nekatera poslovna pravila se izvajajo na strežniku

Arhitektura strežnikov podatkovnih baz



Porazdelitve logike obdelovanja

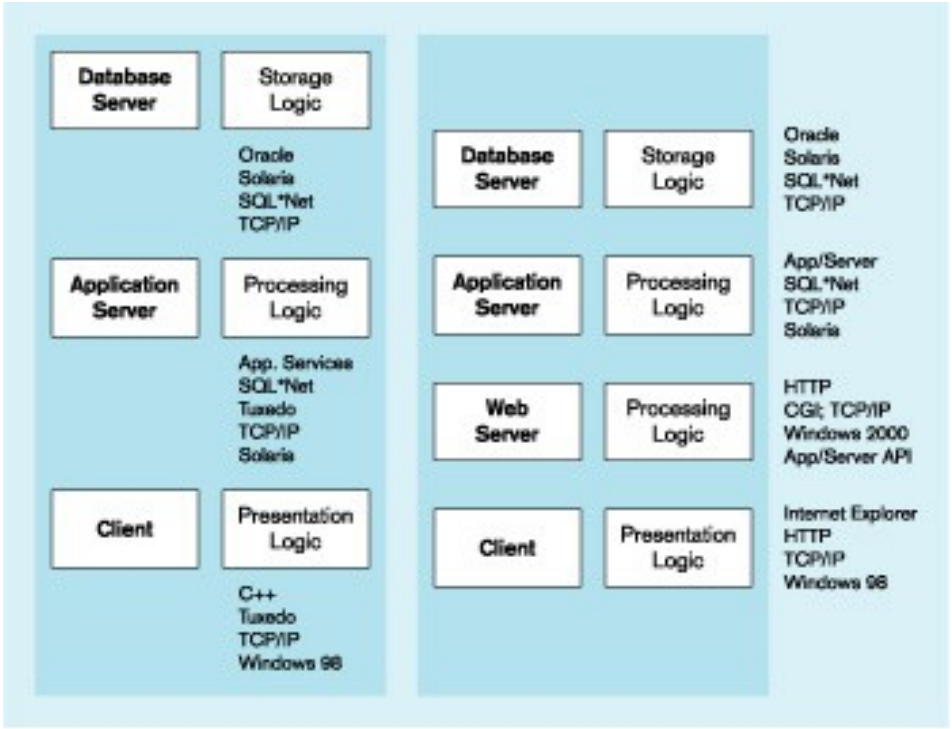


2-slojna porazdelitev

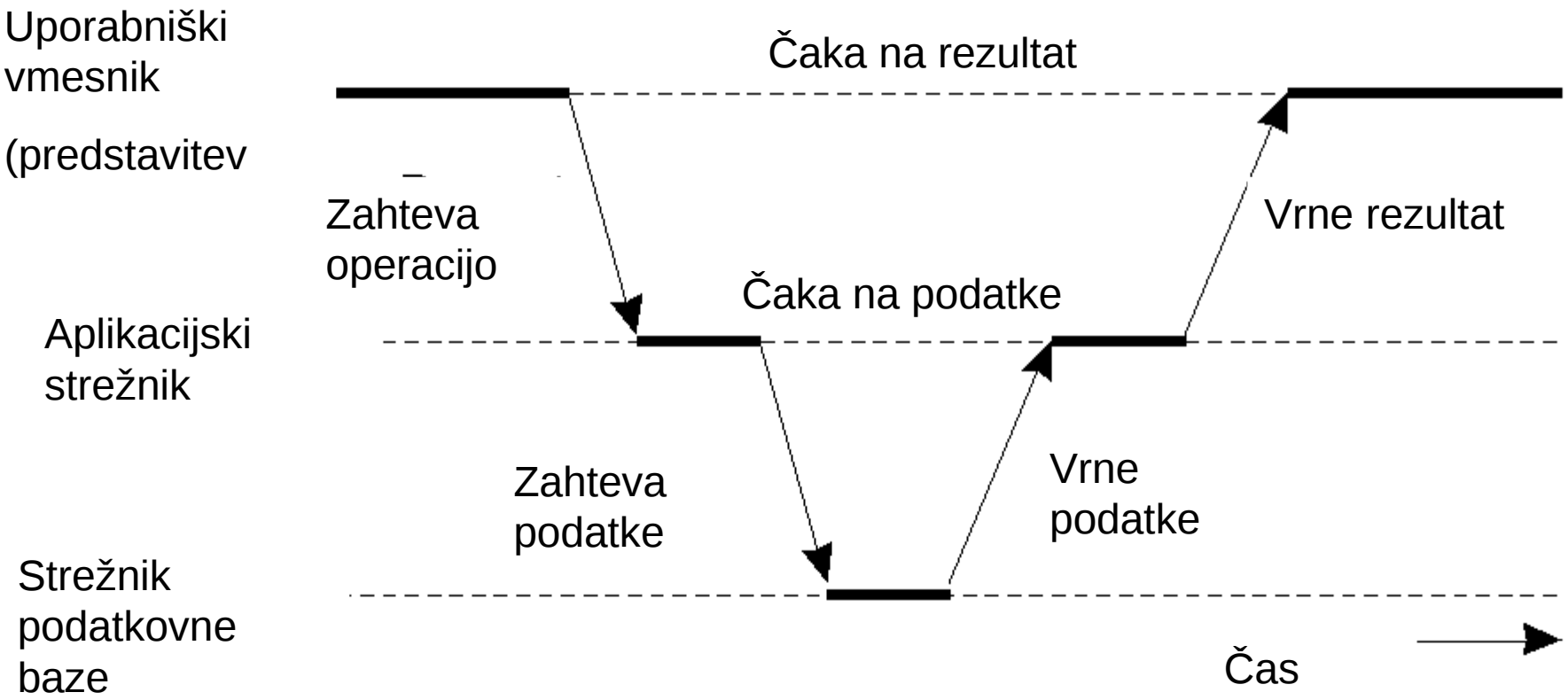
Obdelava poteka na odjemalcu, strežniku ali obeh

n-slojna porazdelitev

Obdelava poteka na aplikacijskem strežniku ali na spletnem strežniku



Večslojne arhitekture



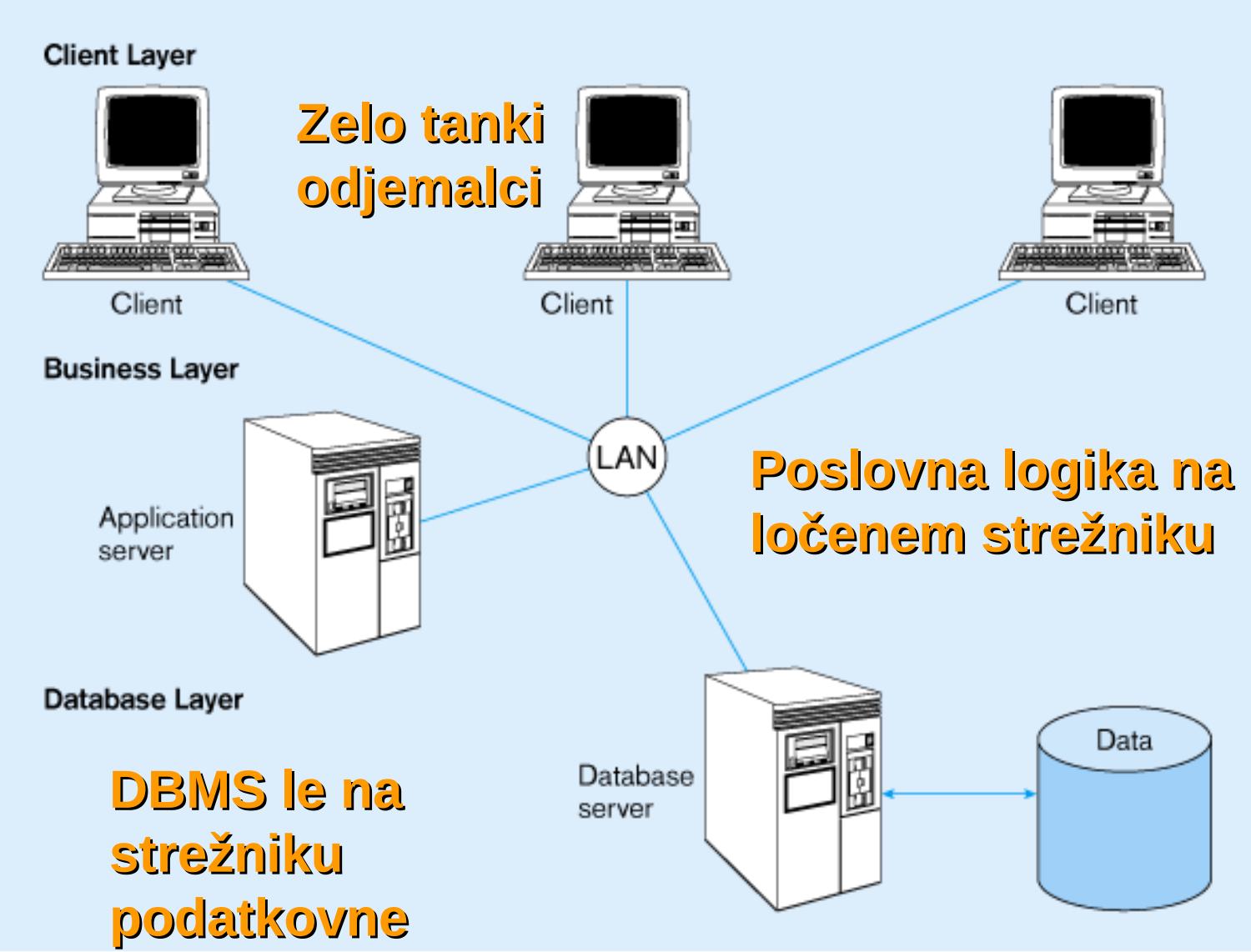
Troslojna arhitektura

- 3 plasti:

Odjemalec	Upor. vmesnik	*Spletni brskalnik*
Aplikacijski strežnik	Poslovna logika	*Spletni strežnik*
Strežnik podat. baze	Pomnilna logika	*DBMS*

- Tanek odjemalec
  - PC služi le kot uporabniški vmesnikin obdeluje bolj malo. Tudi pomni bolj malo podatkov (včasih sploh nima trdega diska)

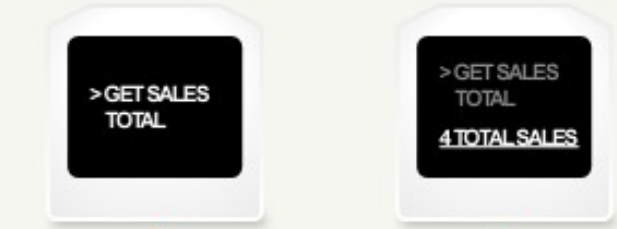
Troslojna arhitektura



Primer: poslovna aplikacija

Predstavitvena logika

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

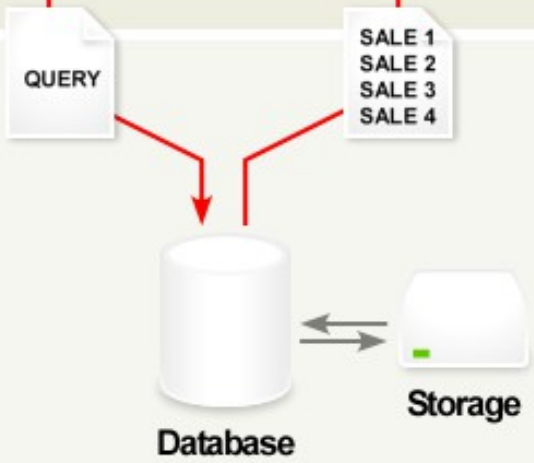


Procesna logika

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Pomnilna logika

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



Troslojna arhitektura

- Prednosti:
  - Skalabilnost
  - Tehnološka fleksibilnost
  - Dolgoročno cenejša rešitev
  - Boljše prilagajanje sistema poslovnim potrebam
  - Izboljšano servisiranje strank
  - Kompetitivne prednosti
  - Zmanjšanje tveganja
- Izzivi:
  - Visoki stroški v začetku
  - Orodja in usposabljanje
  - Izkušnje
  - Nekompatibilni standardi
  - Pomanjkanje kompatibilnih orodij za končnega uporabnika

Varnost sistemov odjemalec/strežnik

- Okolje omrežja ➔ kompleksni problemi varnosti
- Nivoji varnosti:
  - Zaščita z gesli na nivoju sistema
    - Da sploh dopustimo dostop do sistema
  - Zaščita z gesli na nivoju podatkovne baze
    - Da določimo dovoljenja za dostop do tabel; dovoljenja za branje, ažuriranje, vnašanje in brisanje
  - Varnost komunikacije odjemalec/strežnik
    - z enkripcijo

Mobilna koda in Java

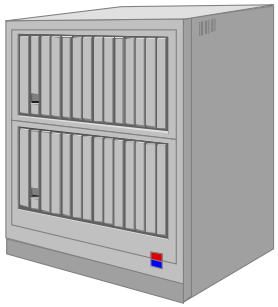
Dva ključna zahtevka:

- Prenosljivost: aplikacije tečejo vzdolž različnih platform
- Interoperabilnost: deli aplikacije morajo sodelovati (izkoriščati omrežje)

Dinamična prenosljivost: mobilna koda

Pošlji kodo (kot obvestilo) gostitelju

Mobilna koda

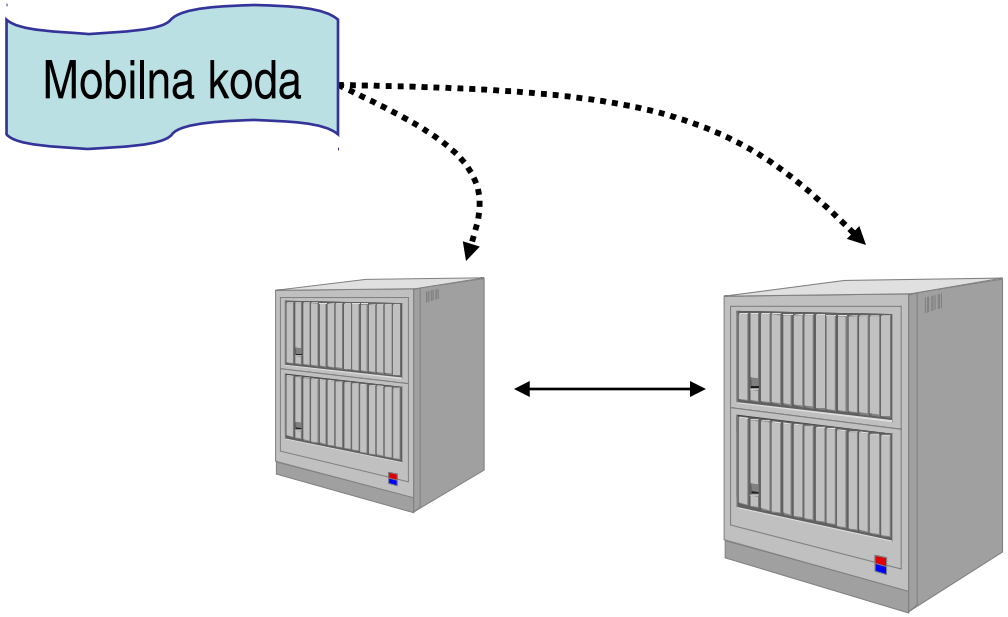


Izvajanje programa,
ki ga predstavlja koda

Mobilna koda:
Koda, ki predstavlja program, ki ga lahko premikamo na heterogene pltforme in tam izvajamo

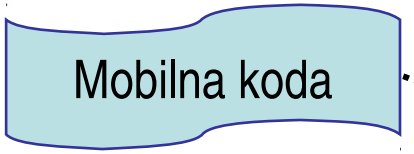
Prenosljivost lahko omogoča interoperabilnost

Mobilna koda, ki izvira iz skupnega vira, poveča interoperabilnost



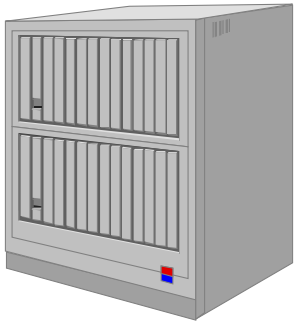
Mobilna koda in mobilni agenti

Mobilna koda:
Koda predstavlja program



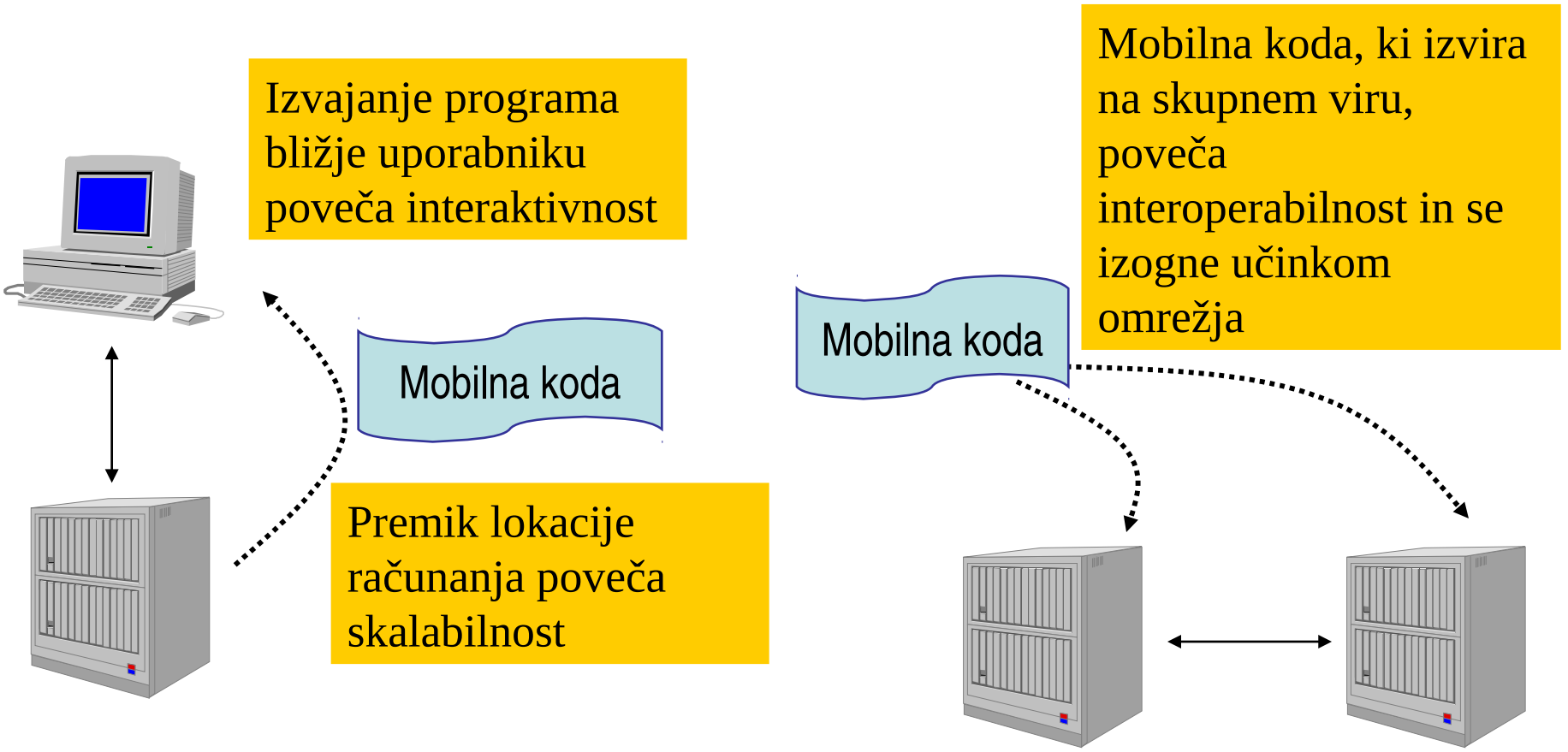
Pošlji obvestilo gostitelju

Mobilni agent:
Koda in podatki predstavljajo objekt ali komponento



Izvajanje programa

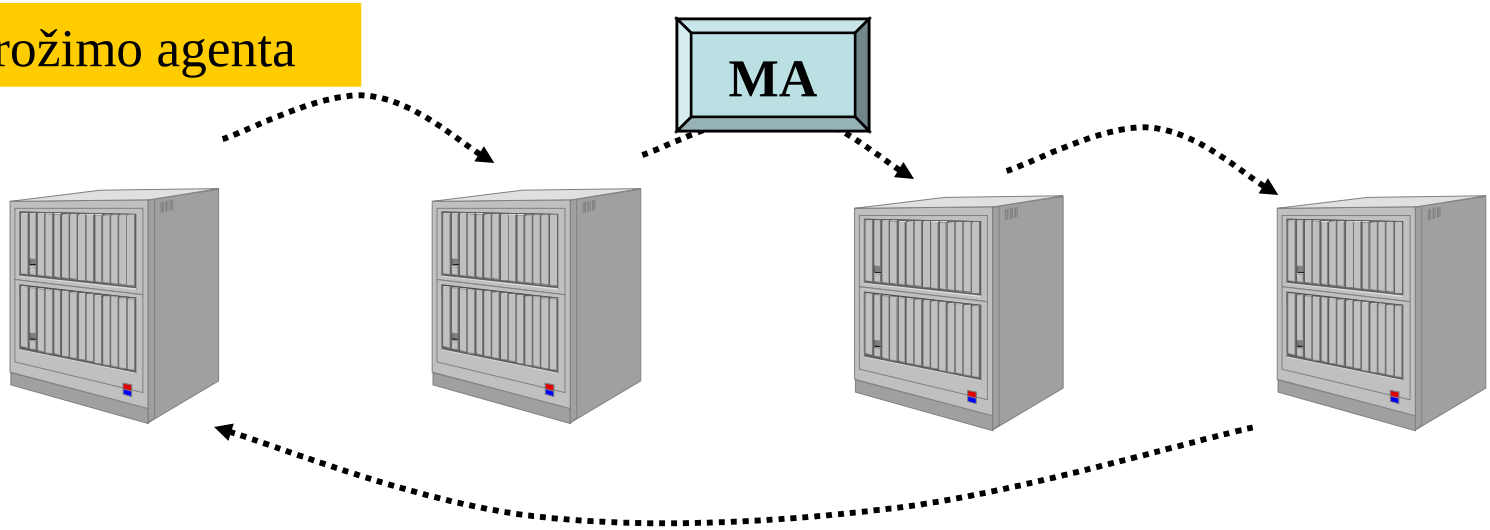
Nekaj prednosti mobilne kode



Zamisel o mobilnih agentih

Agent teče na gostiteljih in spreminja svoje stanje

Sprožimo agenta



Agent vrača