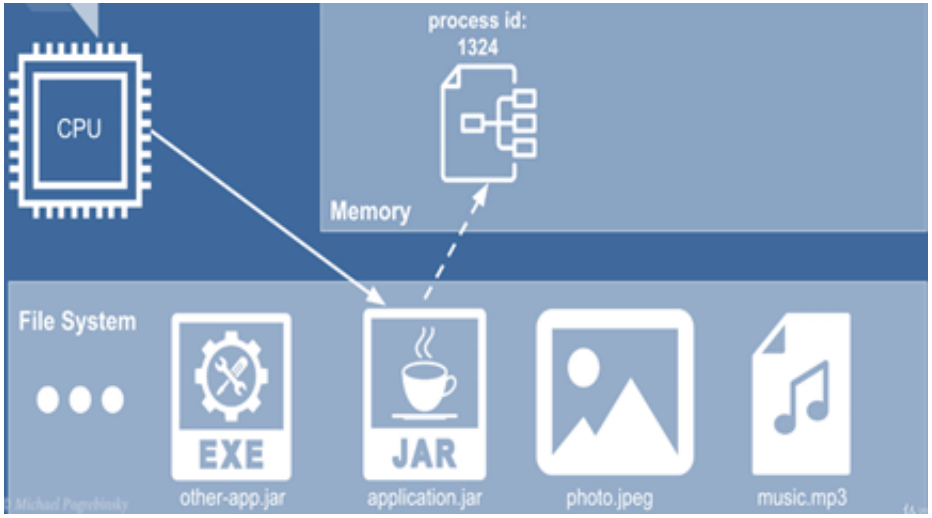
Distribuirani sistemi
Terminologija

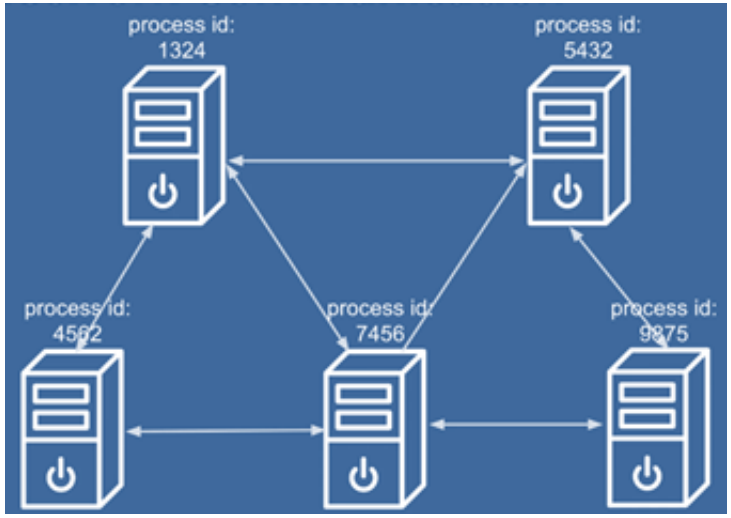
Predmet: Distribuirani sistemi

Predavač: dr Dušan Stefanović

- Distribuirani sistem
  - Sistem sastavljen od procesa
    - Instanca aplikacije koju kreira OS u memoriji (slika 1)
    - Izvršavaju se na različitim računarima,
  - Međusobno komuniciraju preko računarske mreže (slika 2)
  - Dele status ili rade zajedno



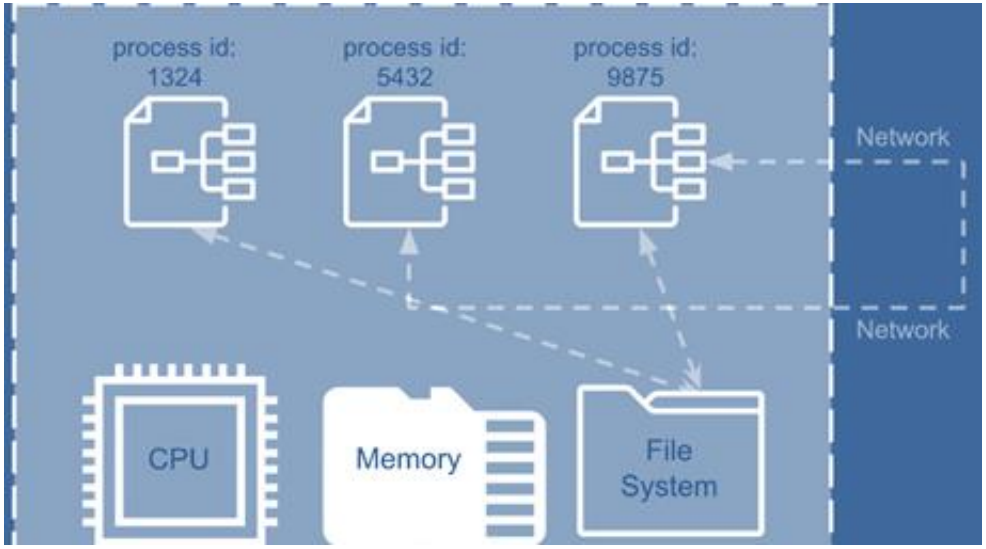
slika 1



slika 2

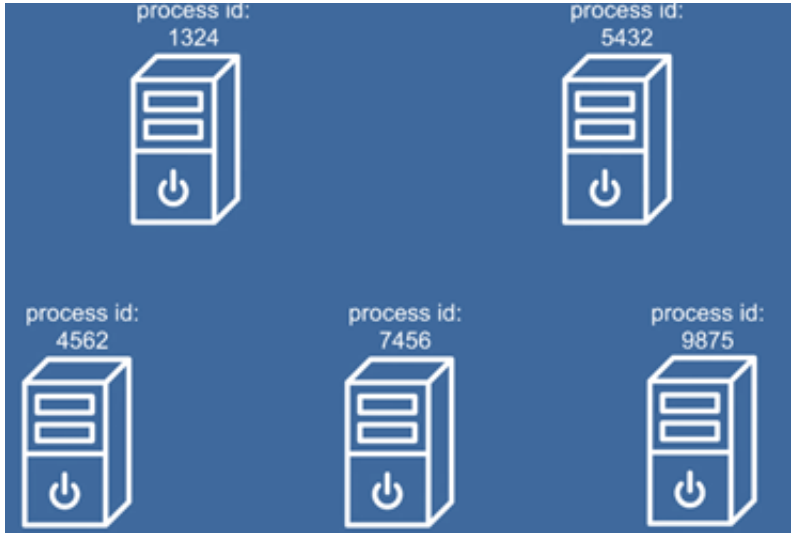
Komunikacija između procesa na istoj mašini

- Proces je izolovan od bilo kog drugog procesa koji se izvršava na istom računaru bez obzira da li se radi o procesima iste aplikacije ili druge aplikacije
- Procesi koji se izvršavaju na istoj mašini mogu da komuniciraju sa memorijom, fajl sistemom i međusobno
- Ovo nije distribuirani sistem jer se svi procesi izvršavaju na istoj mašini
  - Ne mogu da se skaliraju sa kapacitetom koji je veći od kapaciteta mašine i svi dele resurse jedne mašine



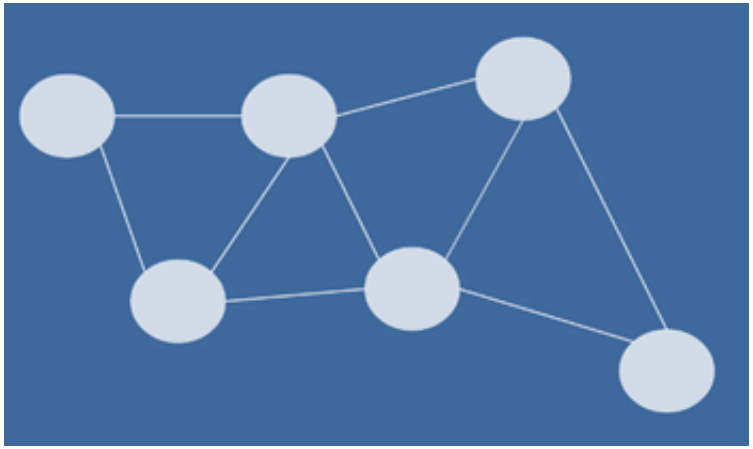
Komunikacija između procesa na različitim mašinama

- Procesi su na različitim mašinama potpuno odvojeni
- Horizontalno se skaliraju po potrebi dodajući nove mašine proširujući memoriju i procesorsku snagu sistema
- Preko računarske mreže procesi međusobno komuniciraju
- Ukoliko ne komuniciraju ne radi se o distribuiranom sistemu.
- Potrebno je razviti algoritam koji će omogućiti da procesi međusobno komuniciraju



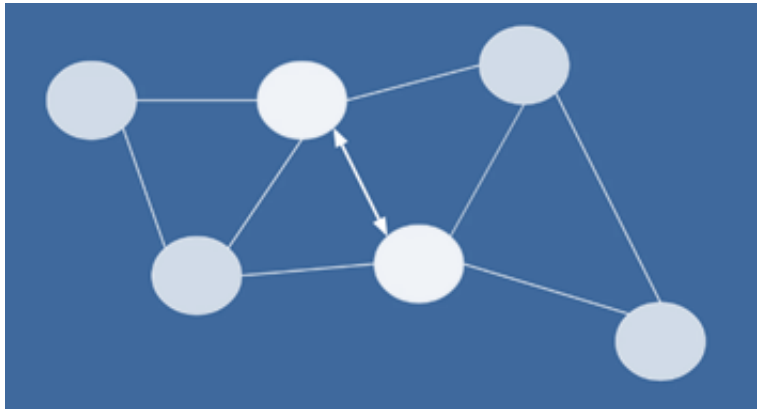
Čvor (Node)

- Proces koji se izvršava na svojoj mašini – deo distribuiranog sistema
- Termin potiče iz teorije grafova



Veza (Edge)

- Opisuje vezu između dva čvora tj. procesa
- Označava da dva procesa mogu da komuniciraju međusobno putem računarske mreže

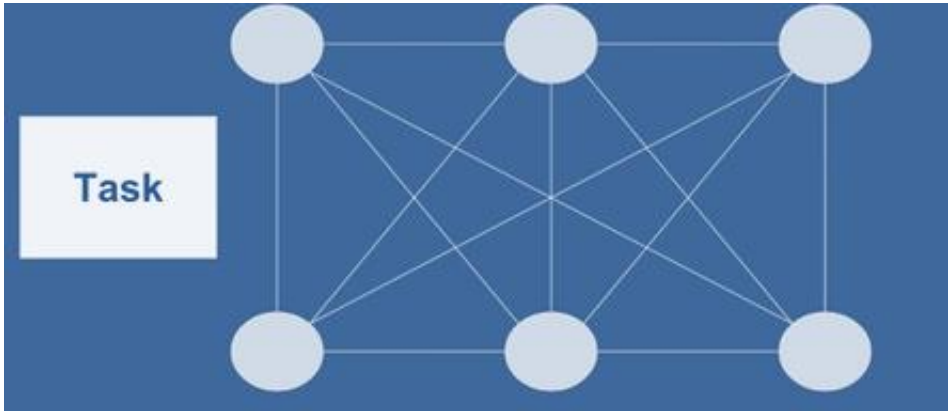


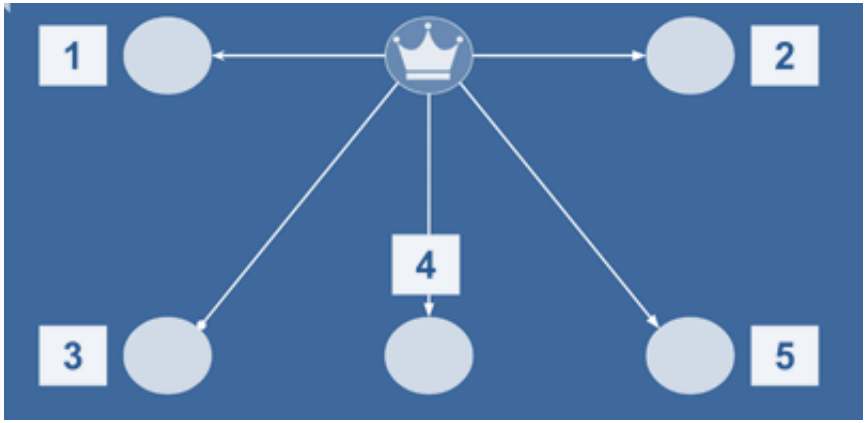
Klaster

- Kolekcija računara (čvorova) koji su međusobno povezani
- Čvorovi u klasteru rade na istom zadatku i obično izvršavaju isti kod

Podela zadataka u klasteru

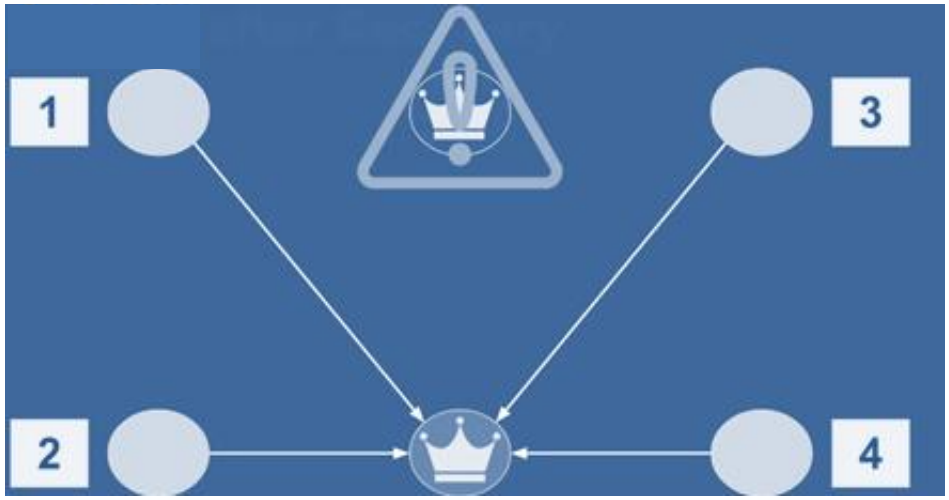
- Kompleksna izračunavanja i analiza velike količine podataka se dodeljuju klasteru
- Podela zadatka na čvorove
- Paralelni rad i distribucija zadataka na čvorove
- Izbor čvora koji će da bude zadužen za distibuciju zadataka i sakupljane rezultata (master node)
  - Otkaz čvora (master) je problem u ovakvoj topologiji





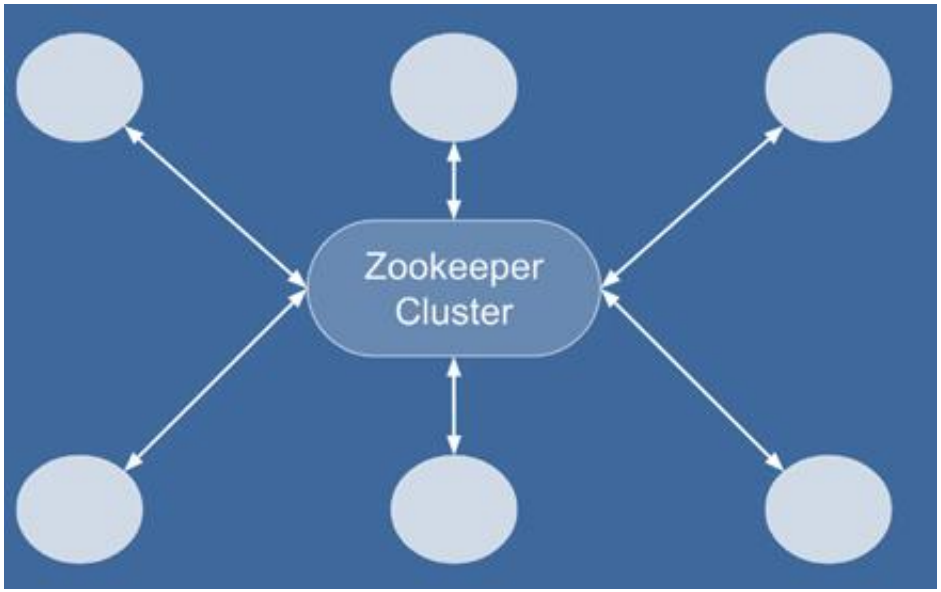
Algoritam za biranje master čvora

- Izbor novog master čvora u slučaju da trenutni nije dostupan
- Nakon oporavka postaje regularan čvor
- Kriterijum za izbor master čvora
  - Podrazumevano čvorovi nemaju informaciju o ostalim čvorovima u klasteru
  - Potreban je servis ta registraciju čvorova
  - Potreban je mehanizam za otkrivanje otkaza master čvora i automatizacija izbora novog master čvora



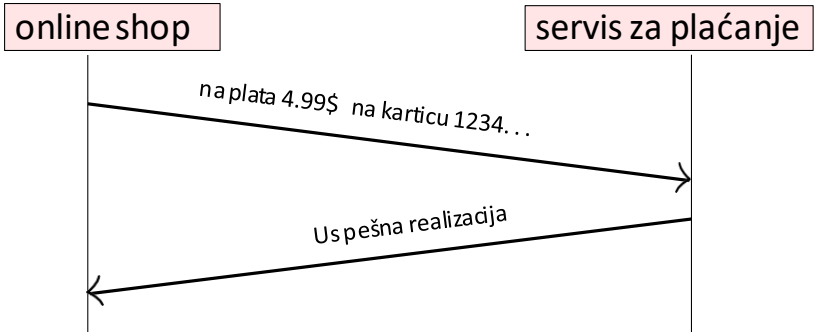
Apache Zookeper

- Servis visokih performansi za kordinaciju u distribuiranom sistemu
- Servis otvorenog koda
- Dizajniran isključivo za distribuirane sisteme
- Primenjuje se na klaster koji se obično sastoji od neparnog broja čvorova.
- Oslanja se na redudatnost da bi i usled otkaza obezbedio rad sistema



Pozivanje udaljene procedure (RPC)

- Primer distribuiranog sistema je kupovina online proizvoda koji se plaćanje credit/debit karticom
- Web shop aplikacija koristi servis koji je specijalizovan za obradu plaćanja preko Interneta
  - Kada klijent pokrene kupovinu, parametri sa njegove kartice se šalju servisu koji je nezavisan od web shop aplikacije na obradu.
  - Servis za plaćanje (payment servis) ustvari komunicira sa infrastrukturom kartice (Visa, MasterCard,...) koja komunicira sa bankog gde je izdata kartica da bi se izvršilo plaćanje.



Primer koda za pozivanje udaljene procedure

- Kod za obradu plaćanja na strani online shop web aplikacije

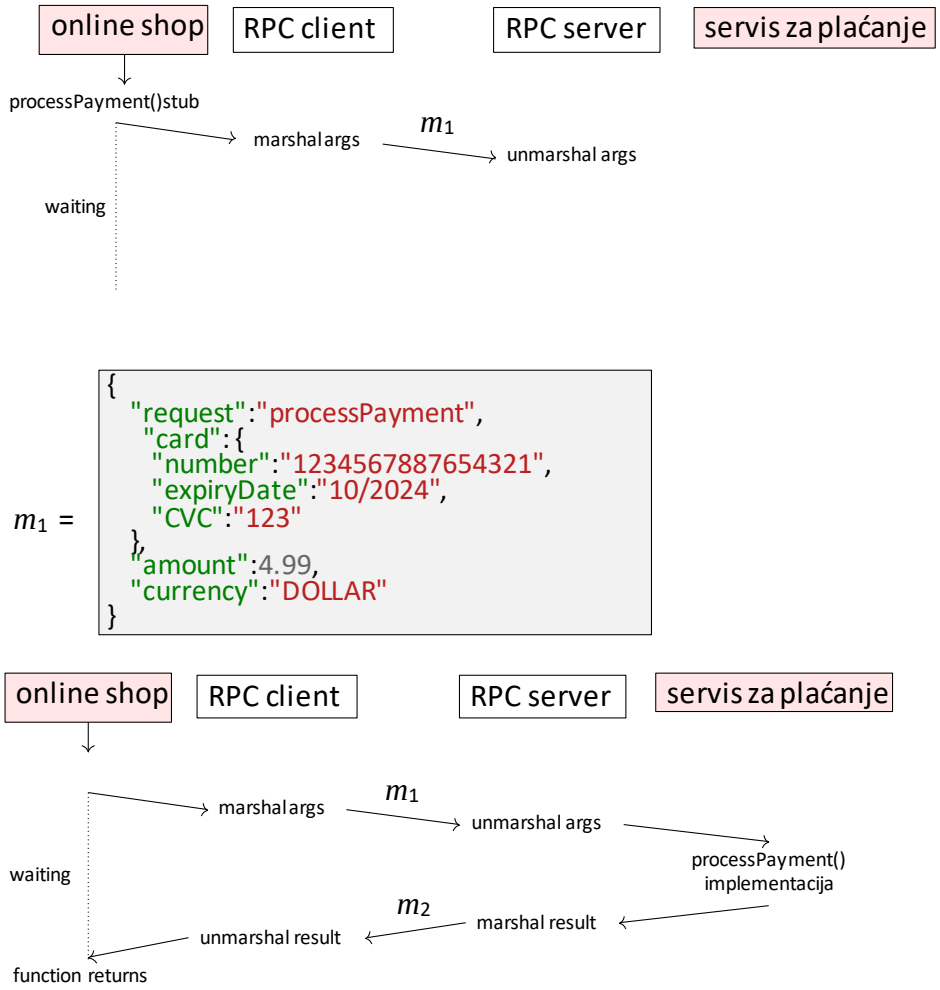
```
// Online shop - upravljanje detaljima kartice
Card card=new Card();  card.setCardNumber("1234 5678 8765 4321");
card.setExpiryDate("10/2024");  card.setCVC("123");
Result result=paymentsService.processPayment(card,  4.99, Currency.dollar);
if(result.isSuccess())
{
  fulfilOrder();
}
```

Implementacija Payment funkcije je na drugom čvoru

- Pozivanje funkcije za plaćanje iza scene šalje zahtev ka servisu za plaćanje, čeka odgovor i vraća odgovor
- Implementacija procesa samog plaćanja i sam kod se ne nalazi u web aplikaciji
  - To je deo servisa za plaćanje - drugi program koji se izvršava **na drugom čvoru** i pripada drugoj kompaniji.
- Vrsta interakcije gde kod na jednom čvoru poziva funkciju na drugom čvoru je Remote Procedure Call
- Software koji implementira RPC se zove **RPC framework ili middleware**
  - RPC framework **argumente funkcije enkodira** u poruku i šalje ih servisu za plaćanje

Pozivanje udaljene procedure (RPC)

- Proces enkodiranja argumenata funkcije se zove ***marshalling*** *i najčešće korišćeni format je JSON*
- *Slanje poruke sa RPC klijenta na RPC server obično je preko HTTP protokola (web servis)*
- *Na serverskoj strani RPC framework dekodira poruku i poziva željenu funkciju sa poslatim argumentima*
- *Kada funkcija vrati rezultat, on se enkoduje šalje u dogovorenom formatu klijentu*
- *Pozivaocu funkcije izgleda da se funkcija izršila lokalno*



Pozivanje udaljene procedure (RPC)

- RPC je onogućio da poziv ka udaljenoj funkciji bude isti kao poziv lokalne funkcije
- Sistem krije gde su locirani resursi

“Lokalna transparetnost”:

U praksi .....

- Šta ukoliko servis otkaže tokom poziva funkcije ?
- Šta ukoliko se poruka izgubi?
- Šta ukoliko poruka zakasni?

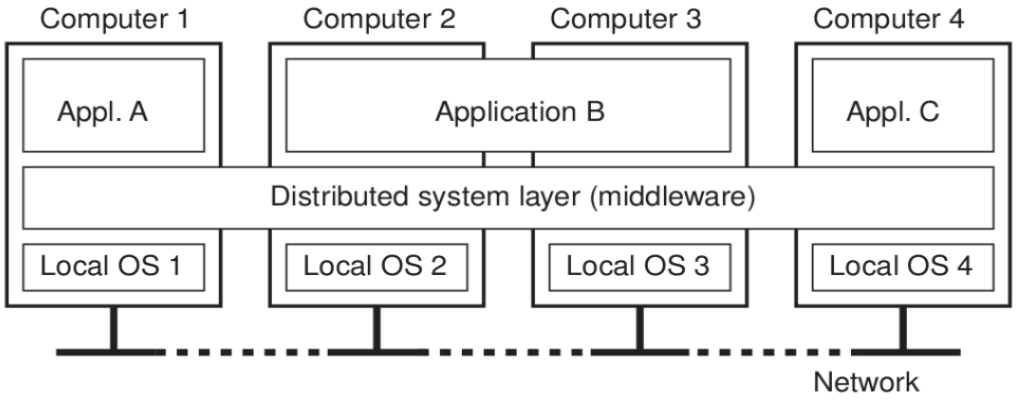
Izazovi o kojima mora da se vodi računa u odnsu kada se resurs nalazi na lokalnom računaru

Pozivanje udaljene procedure (RPC)

- Tokom poslednje dekade razvijeni su različiti RPC sa ciljem da olakšaju programiranje distribuiranih sistema
  - Objektno orjentisani middleware kao što je COBRA (1990)
  - Izazovi u nižim sljevima distribuiranog sistema su i dalje ostali isti

Istorija razvoja RPC servisa

- SunRPC/ONC RPC (1980)
- CORBA: object-oriented middleware, 1990
- Microsoft's DCOM and Java RMI (slično - CORBA)
- SOAP/XML-RPC: RPC primenom XML i HTTP (1998)
- Thrift (Facebook, 2007)
- gRPC (Google, 2015)
- REST (najčešće je JSON)
- Ajax u web browsers



REST API

- RPC se danas najčešće implementira koristeći slanje podataka u JSON formatu preko HTTP protokola
- Popularni skup principa u dizajnu za takve API-je zasnovan na HTTP-u poznat je kao REST(*representational state transfer*) a API-ji koji se pridržavaju ovih principa nazivaju se RESTful.
  - Komunikacija je bezkonekciona (stateless), svaki zahtev je samostalan i nezavisan od ostalih
  - Resursi (objekti nad kojima se manipuliše) se predstavljaju URL adresama
  - Stanje objekta se ažurira preko HTTP zahteva standardnom metodom POST ili PUT do odgovarajuće URL adrese
- Popularnost REST API leži u korišćenju JS koda koji se danas izvršava u svim Web čitačima a koji olakšava kreiranje HTTP zahteva ka serveru.

REST API

- Kod uzima argumente, prevodi u JSON format koristeći funkciju JSON.stringify() i šalje na URL adresu https://example.com/payments preko HTTP POST zahteva

```
Let args={amount:4.99, currency:'DOLLAR',                    /*...*/};
let request={
    method:'POST',
    body:JSON.stringify(args),
    headers:{'Content-Type':'application/json'}
};

fetch('https://example.com/payments', request)
    .then((response) => {
        if(response.ok) success(response.json());
        Else failure(response.status);                  // server error
    })
    .catch((error) => {
        failure(error); // network error
    });
```

- Dva moguća scenarija
  - Server može da vrati status da je zahtev uspešno obrađen (response.json() metode čita primljeni json fajl) pozivajući success funkciju
  - Server može da vrati status da je zahtev nije uspešno obrađen ili da zahtev ne može da se isporuči do servera
- RESTful API i RPC zasnovan na HTTP su na webu i koriste se često ne samo u slučaju kada je JS klijent već u server server komunikaciji i mobilnim aplikacijama

RPC u distribuiranim sistemima

Server to Server RPC se primenjuje u distribuiranim sistemima gde je softver isuviše složen i veliki da se izvršava na jednoj mašini

"Service-oriented architecture" (SOA) / "microservisi":

- Razdvajanje složenog softvera u više servisa
- Na više čvorova koji komuniciraju preko RPC-a.
- Različiti timovi mogu da održavaju takav softver i da bude napisan na različitim programskim jezicima

Različiti servisi su pisani u različitim tehnologijama

- **Interoperabilnost** – konverzija tipova podataka da se poslati argumenti kompatibilni sa funkcijom koja se poziva na drugom čvoru i da su vraćeni podai mogu da se pročitaju
- **Interfejs za desinisanje jezi**ka (IDL) – jezik nezavisan od API specifikacije

Primer Interfejsa za definisanje jezika (IDL) gRPC

```
messagePaymentRequest{  messageCard{
      required string cardNumber=1;
      optional int32 expiryMonth=2;
      optional int32 expiryYear=3;
      optional int32 CVC=4;
   }
   Enum Currency {EUR=1;USD=2;  }

    required Card card=1;
    required int64 amount=2;
    required Currency currency=3;
}

messagePaymentStatus{
Required bool success=1;
Optional string errorMessage=2;
}

Service PaymentService {  rpc ProcessPayment(PaymentRequest) returns (PaymentStatus) {}
}
```