UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA AUTOMATIKU I UPRAVLJANJE SISTEMIMA

Komunikacije

Distribuirani sistemi

Distribuirano programiranje

Uvod

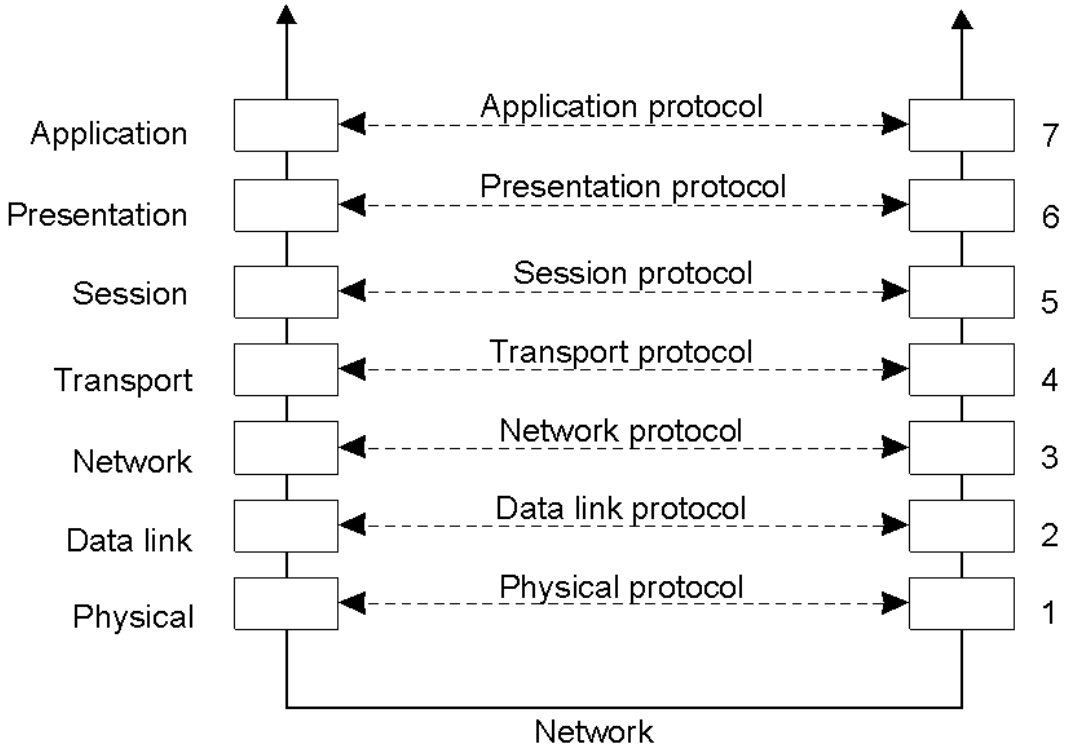
- Međuprocesna komunikacija je srce DS
- Komunikacija ja zasnovana na razmeni poruka (na niskom nivou)
- Teži se da se DS zasnuje na višem nivou komunikacija
  - Da bi se olakšao razvoj aplikacija

Slojeviti protokoli

- Kod prenosa poruke moraju postojati “dogovori”
- ISO referentni model - pojednostavljuje dogovaranje
  OSI model – *Open Systems Interconnection Reference Model*
  - Uveden radi razumevanja računarskih mreža
  - Protokoli razvijeni kao deo OSI modela nisu široko upotrebljeni
  - OSI model omogućava komunikacije u “otvorenom sistemu”
- Protokoli određuju pravila (standardna)
  - Format poruka; Sadržaj; Značenje poruka
- Promene u jednom sloju ne utiču na druge slojeve

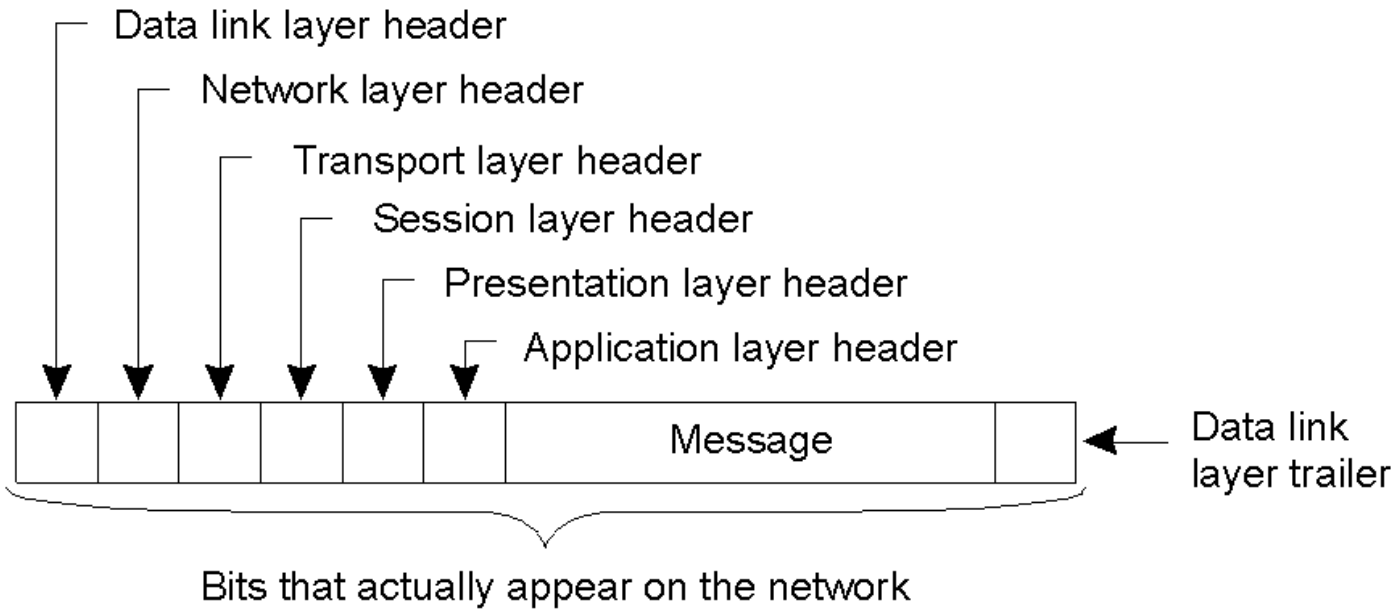
OSI model

- Ima 7 slojeva - *protocol stack*
  - svaki se odnosi na poseban aspekt
  - svaki ima interfejs ka gornjem nivou



Slanje poruke kroz slojeve

- Tokom slanja svaki sloj doda svoj *header*
- Kod prijema se uklanjaju *header*-i



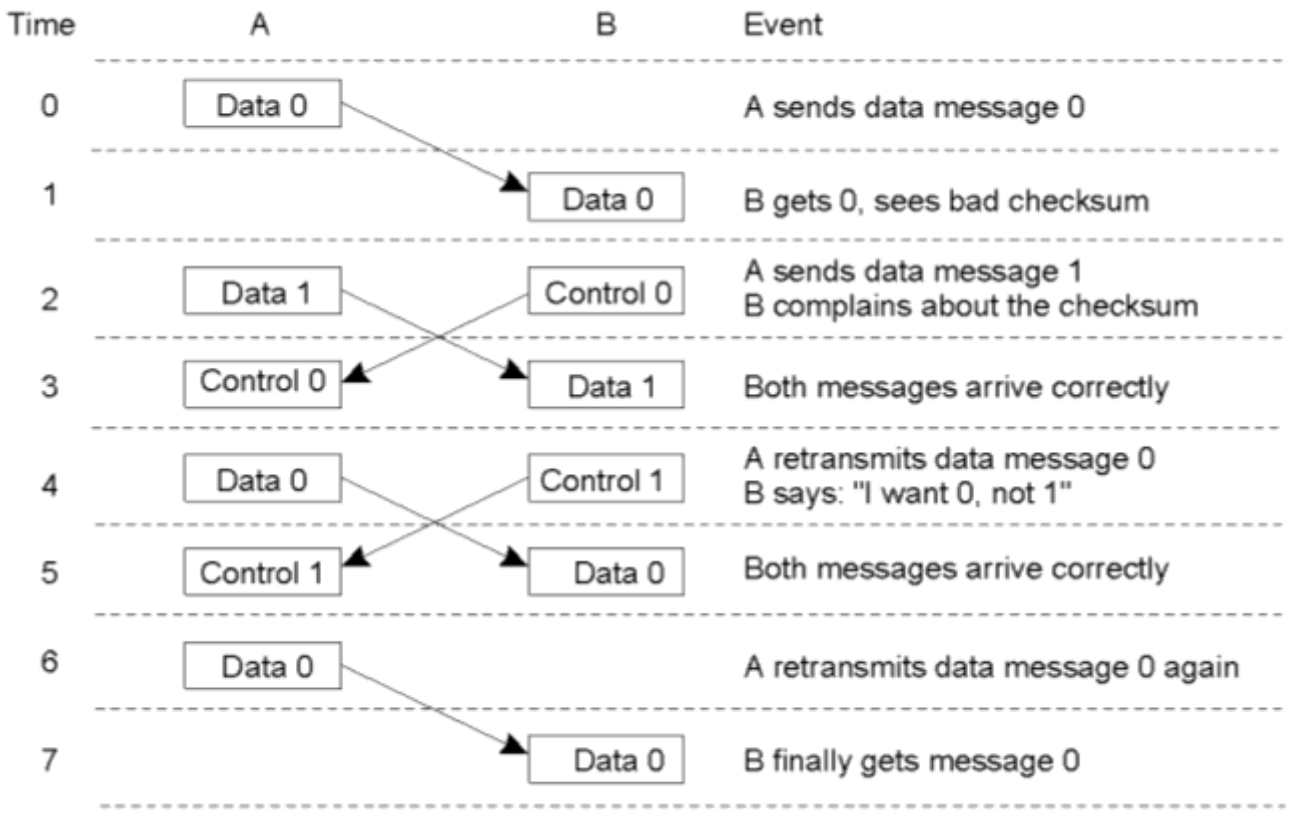
1 - Fizički sloj

- Bavi se prenosom 0-a i 1-ica – šalje bite
- Standardizuje interfejse
  - Električne
  - Mehanike
  - Signale
- Određuje
  - Napon
  - Brzinu prenosa
  - Obostranost slanja
  - Veličina i oblik konektora, broj i značenje pinova

- Primer: RS-232-C

2 - Data-link sloj

- Brine se detekcijom grešaka u prenosu i njihovom ispravkom
- Grupiše bite u *frame*-ove
  - Postavljaju se posebne oznake (*bit-pattern*) na početku i kraju
  - Dodaje je kontrolna suma (*checksum*)
  - Dodaje se redni broj poruke (u zaglavlje *frame-a*)



3 - Mrežni sloj

- U LAN-u nema potrebe da pošiljaoc poruke locira primača, ali u WAN-u ima potrebe
- Bavi se rutiranjem poruka
  - Problem: najkraći put ne znači i najbolji
  - Na put poruka utiču: kašnjenja, gustina saobraćaja, broj poruka koje su spremne za slanje
    - Promene su dinamične
- Primeri:
  - IP protokol (*Internet Protocol*)
    - Deo Internet protokol steka
    - Najšire prihvaćen
  - Virtuelni kanal ATM mreža

4 - Transportni protokoli

- Mogu ih koristiti programeri aplikacija
  - Poslednji sloj osnovnog protokol steka
- Namena: isporučiti poruku do odredišta
- Preuzima poruku od višeg sloja
  - Razbija je na delove (koji se mogu preneti)
  - Svakom delu dodaje redni broj
  - Šalje delove
- Zaglavlje sadrži informacije o
  - Koji delovi su poslati
  - Koji su primljeni
  - Koje treba ponovo poslati
  - Koliki je prihvatni prostor na prijemnoj strani
  - ...

Transportni protokoli TCP/UDP

- *Connection-oriented*
  - Prvo se uspostavlja veza i vrši dogovor oko daljih protokola
  - Primer: telefonski razgovor
  - **TCP** (*Transmission Connection Protocol*) najrasprostranjeniji predstavnik Internet protokol steka
    - Pouzdan u svakoj mreži
    - Značajan *overhead* (pogotovo u LAN)
- *Connectionless*
  - Primer: slanje pisma poštom
  - **UDP** (*Universal Datagram Protocol*)
    - Neznatno proširuje IP
    - Potrebna dodatna kontrola greške i prenosa paketa
- TCP/UDP su pogodni za klijent-server model

OSI slojevi visokog nivoa

- 5 - *Session* sloj
  - Proširena verzija transportnog sloja
  - Dodaje dijalog da se zna koja strana “priča”
  - Dodaje sinhronizaciju strana

- 6 - Prezentacioni sloj
  - Brine se o značenju bita – olakšava komunikaciju strana sa različitim prezentacijama podataka
  - Mogu se definisati slogovi sa poljima

- 7 – Sloj aplikacija
  - Mnoštvo standardnih mrežnih aplikacija
  - email, FTP, emulacija terminala, HTTP ...
  - Iz perspektive OSI modela svi DS su aplikacije

Berkeley Sockets

- Biblioteka funkcija koja omogućava slanje i prijem poruka preko Internet transportnih protokola
- *Socket* je *endpoint* (*port*) – (OS ga koristi za transportni protokol)
  - U koji aplikacija upisuje podatke
  - Iz koga se mogu čitati pristigli podaci

- *Socket* rutine za TCP/IP

  socket – kreira novi *endpoint*

  bind – poveže lokalnu adresu sa soketom

  listen – objavi spremnost prihvatanja konekcija

  accept – blokira dok se ne pojavi zahtev za konekcijom

  connect – namerava da uspostavi konekciju

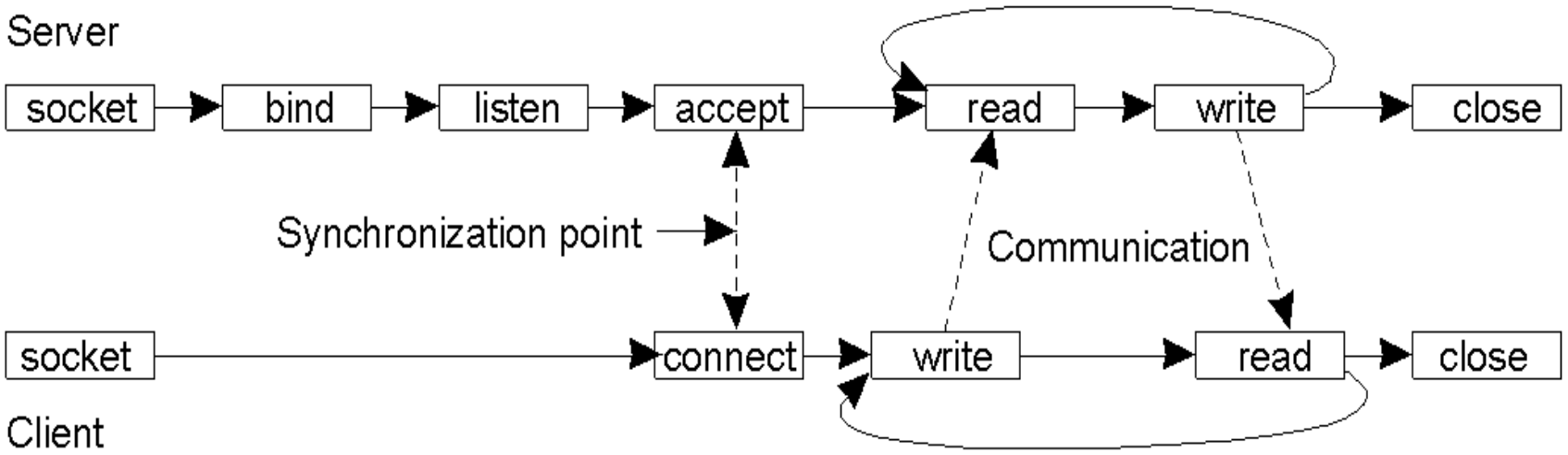
  send – šalje podatke preko konekcije

  receive – prima podatke preko konekcije

  close – oslobađa konekciju

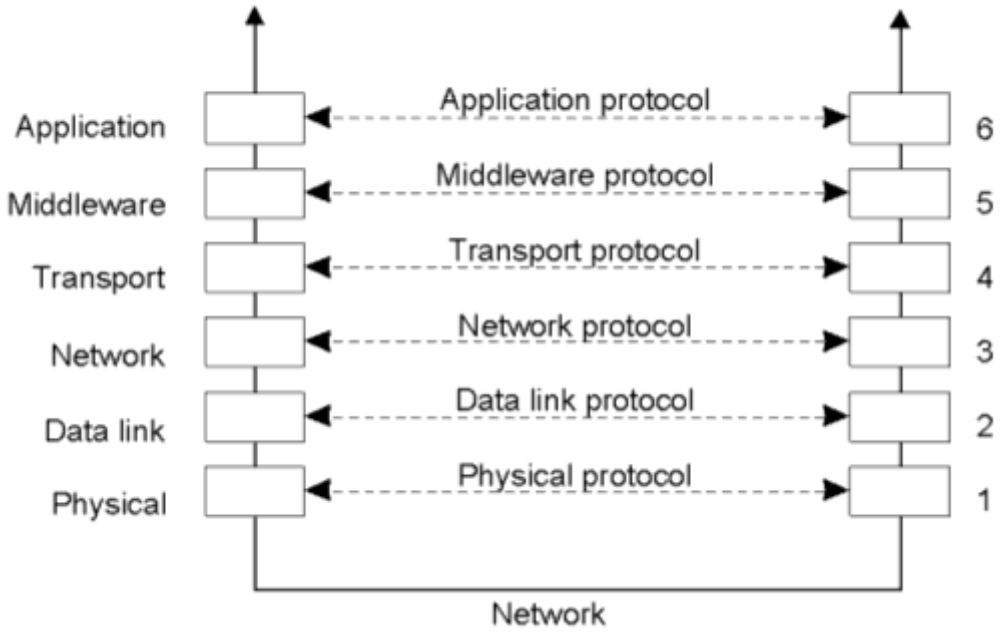
Berkeley Sockets (2)

- Tipična upotreba *socket-a*



Middleware protokoli

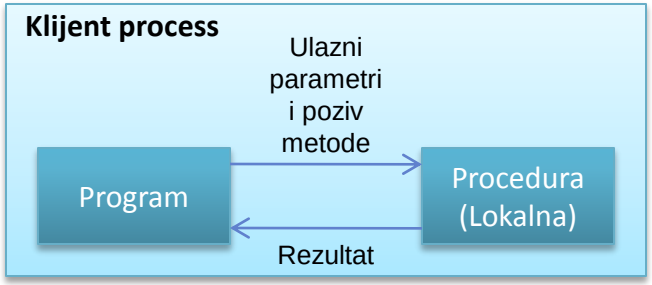
- *Middleware* je aplikacija iz OSI aplikacionog sloja
- Dodaje dodatne protokole opšte namene
  - Slojevit je
  - Primeri
    - Autentifikacija
    - Distribuiran *commit* protokol
    - …



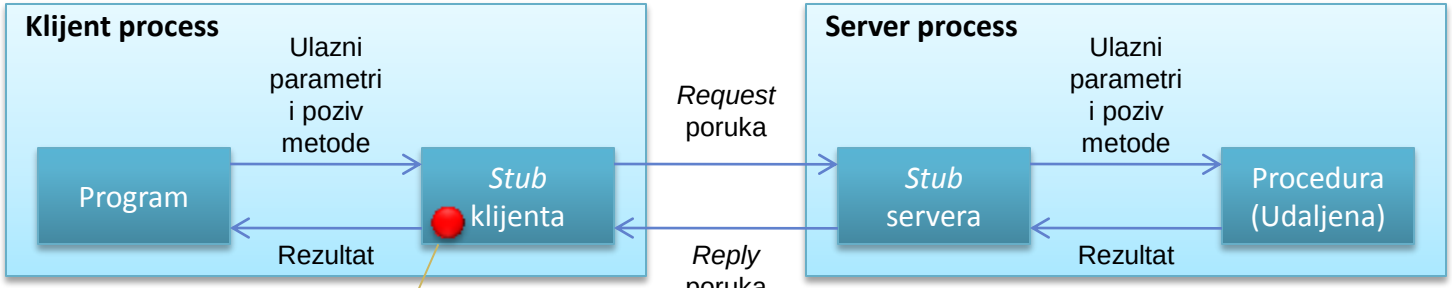
Remote Procedure Call (RPC)

- Protokol za pozivanje procedura iz udaljene mašine
- Treba da liči na poziv lokalne procedure
  - Upotreba *stub*-ova

Poziv lokalne procedure



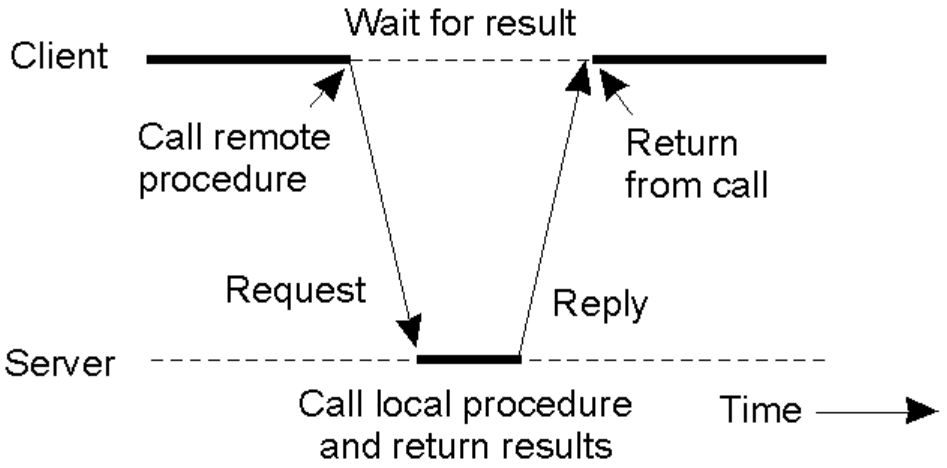
Poziv udaljene procedure



Proces je blokiran dok ne stigne *Reply* poruka

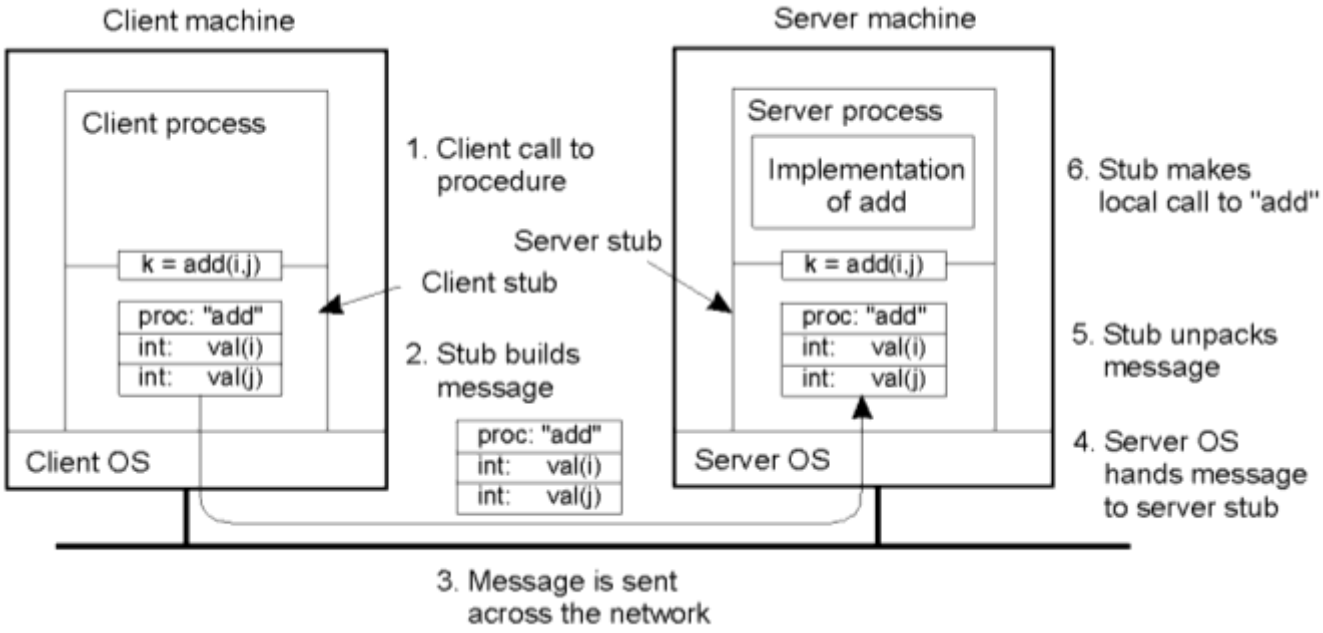
RPC (2)

- Sinhroni poziv



- Poteškoće koje rešava
  - Izvršavanje koda u dve mašine
  - Prosleđivanje ulaznih parametara i rezultata kroz mrežu
  - Predstava brojeva (*byte ordering*)
  - Prenos po referenci
  - Prenos složenih struktura

Koraci RPC-a



1. Klijent poziva klijent *stub* na normalan način
2. Klient *stub* formira poruku i poziva OS
3. Klijent OS šalje poruku u server OS
4. Server OS predaje poruku server *stub*-u
5. Server *stub* otpakuje parametre i poziva metodu servera
6. Server izvršava metodu i vraća rezultat *stub*-u
7. Server *stub* pakuje odgovor u poruku u predaj u OS
8. Server OS salje poruku u klijent OS
9. Klijent OS predaje odogovor klijent *stub*-u
10. Klijent *stub* raspakuje odgovor i predaje klijent programu

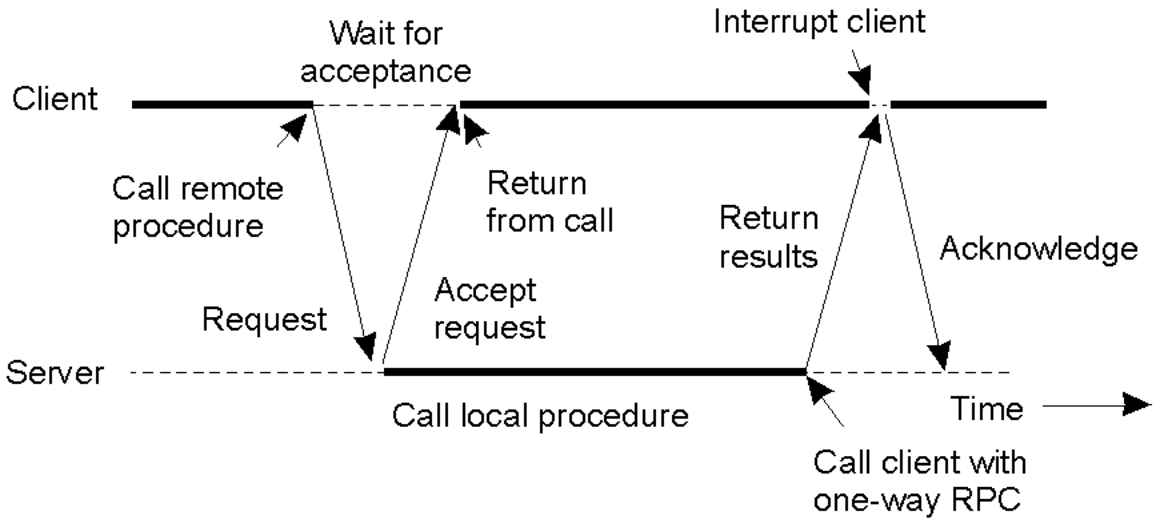
Prošireni RPC model

- Originalni RPC razmenjuje poruke između računara
- Zbog transparentnosti RPC se može upotrebiti i za lokalne pozive
  - Efikasnije je koristiti *Interprocess Communication* (IPC) umesto slanja poruka u okviru jednog računara
- Upotebom proširenog RPC modela nema razlika između lokalnih i udaljenih poziva
  - Migracija koda se lako sprovodi

Asinhroni RPC

- Poziv klijenta lokalnom *stub*-u je blokiran tokom obrade poziva
  - Ovo se događa u svim slučajevima, i kada
    - Nema povratne vrednosti pozvane metode
    - Klijentu ne treba rezultat odmah

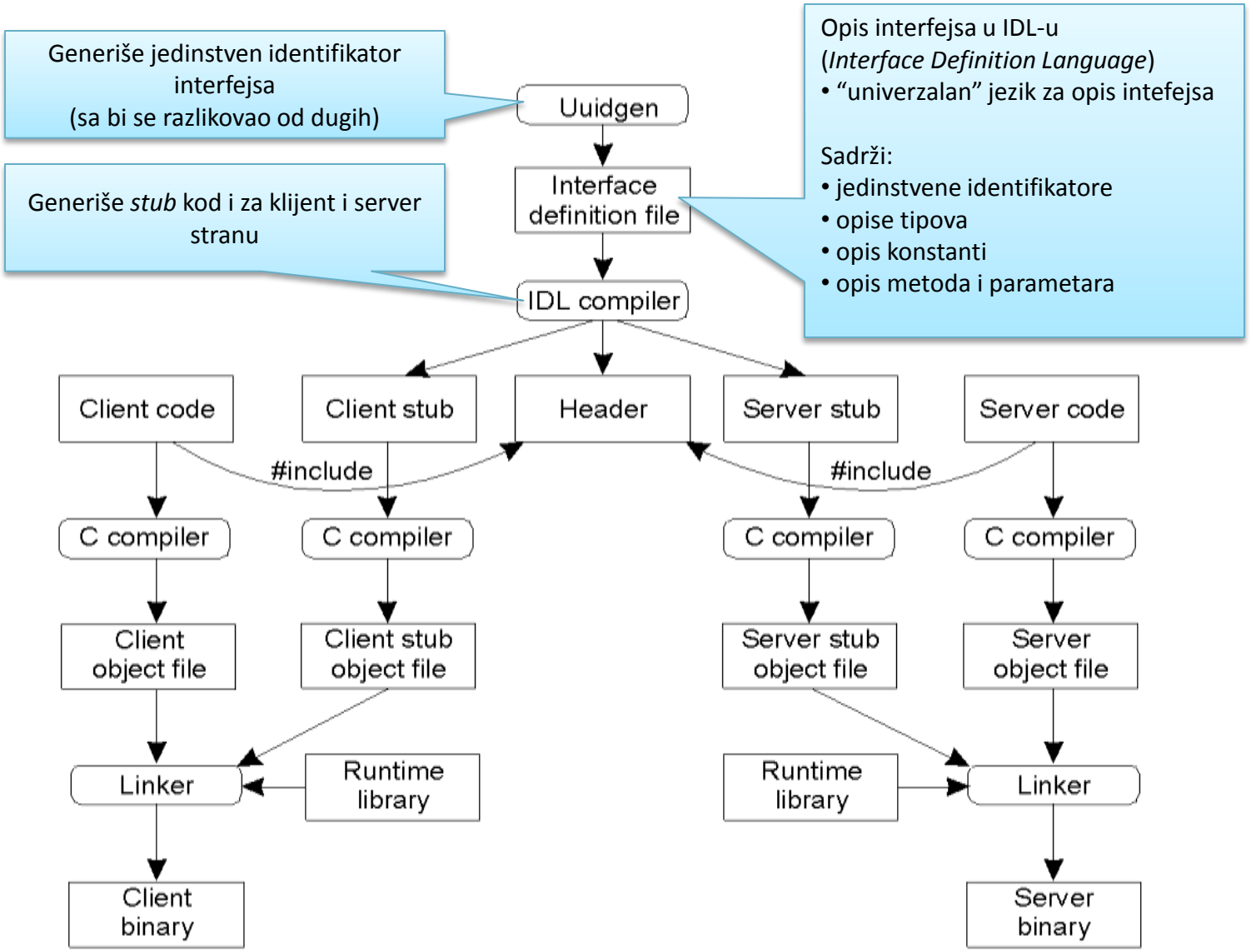
- Asinhroni RPC nakon poziva ne blokira klijenta
  - Odmah se šalje odgovor klijentu da je zahtev prihvaćen
  - Naknadno se šalje rezultat



DCE RPC

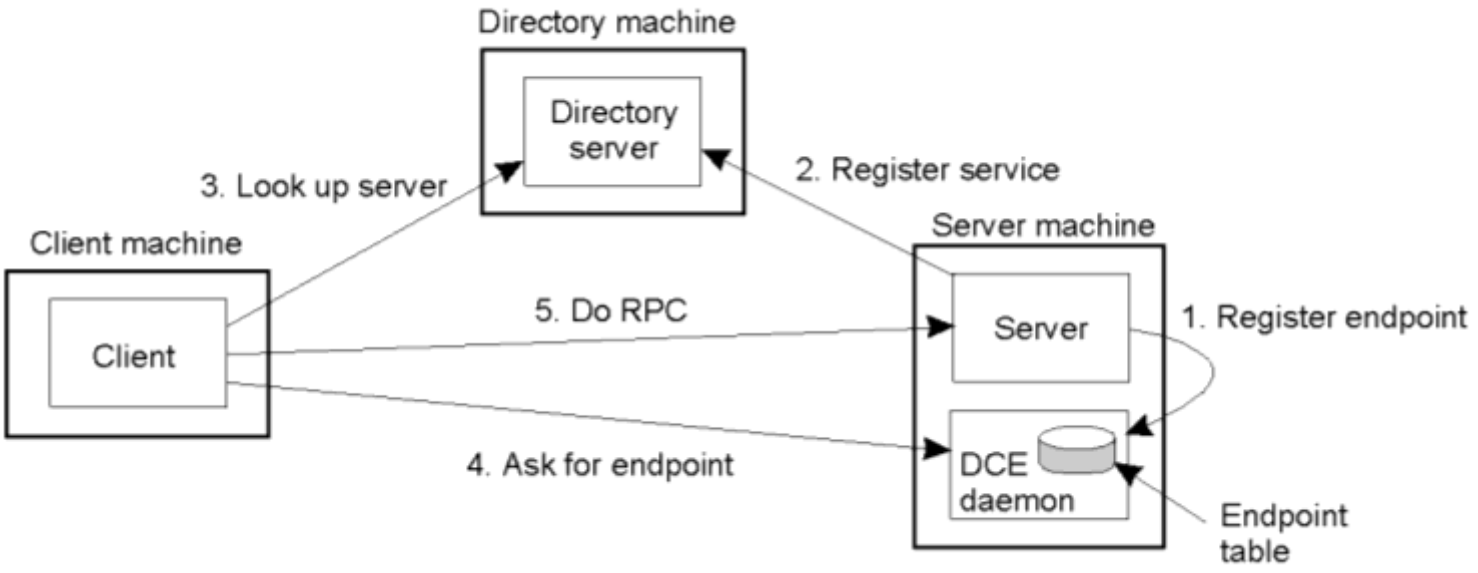
- RPC je osnova za *Distributed Computing Environment* (DCE)
  - Razvijen od strane OMG (*Object Management Group*)
  - Podržan na brojnim OS: UNIX, VAX, Windows NT
- Nudi servise
  - *Distributed file service*
  - *Directory service*
  - *Security service*
  - *Distributed time service*
- Osobine
  - Olakšava programiranje i klijent i server strane
  - Vrši konverzije tipova podataka
  - Automatski locira server (*binding*)
  - Omogućava pisanje aplikacija u raznim alatima (server napisan u C-u, klijent u Javi)

Pisanje klijenta i servera



Povezivanje klijenta i servera

- Server se mora registrovati
- Klijent locira:
  - Računar – treba mu IP adresa
  - Proces u okviru mašine - treba mu *endpoint* (port)



Objekat

- Osnovna ideja objekno-orijentisanog programiranja je da svaki objekat sakriva svoju unutrašnjost

- Objekat
  - Poseduje dobro definisan interfejs
    - Sva komunikacija sa objektom se obavlja preko interfejsa
    - Interfejs čine samo metode
  - Sadrži njemu bitne podatke (stanje objekta), ali sakrivene
    - Stanje se može menjati samo pozivanjem metoda

- Ako imamo dva objekta sa istim interfejsom onda se jedan može koristiti umesto drugog (*polimorfizam*)
  - Iako im je unutrašnjost (implementacija metoda i/ili podaci koje sakrivaju) drugačija
  - Postojeći objekat se može zameniti novim sa izmenjenim (proširenim) ponašanjem, a da to ne utiče na ostali deo sistema

Stanje objekta

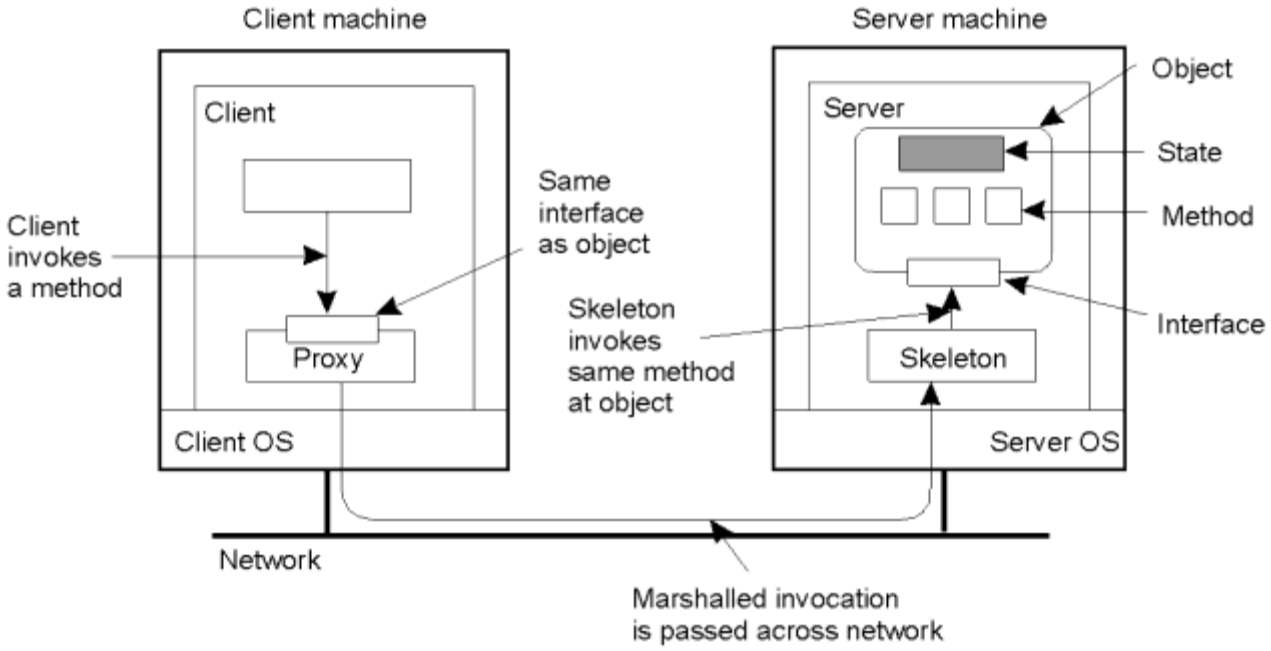
- Stanje objekta sačinjavaju njegovi podaci
  - Tipično životni vek objekta zavisi od tih podataka

- U zavisnosti od mesta skladištenja objekat ima:
  - Lokalno stanje (skladišteno na jednom računaru)
  - Distribuirano stanje (skladišteno na više računara) – *distributed object*

- *Persitent* i *transient* objekti
  - **Perzistentan** objekat ima stanje koje nadživi server
  - **Tranzientan** objekat nestaje pre nego se ugasi server

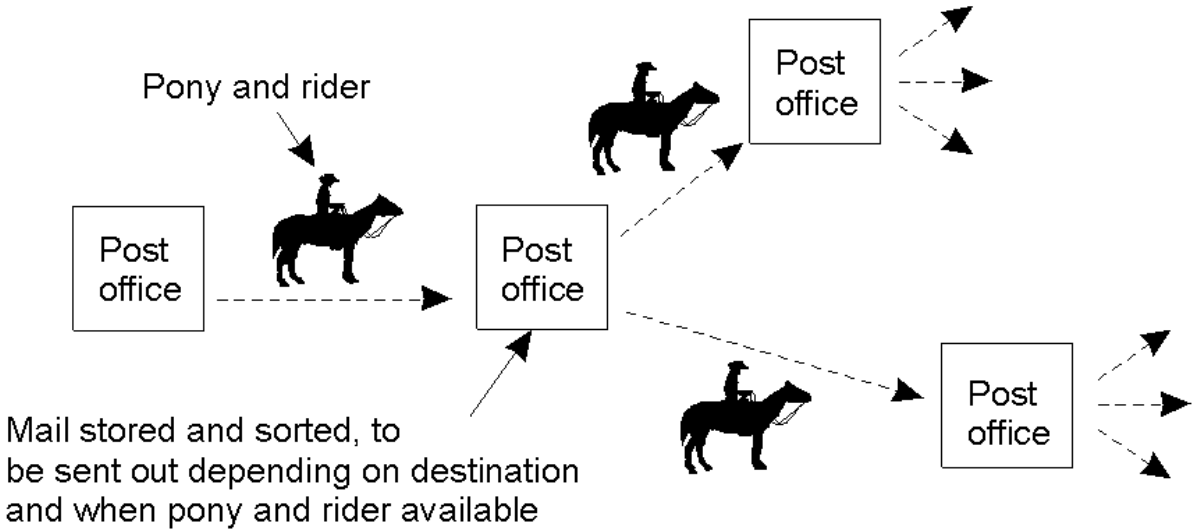
Udaljen objekat

- *Remote Method Invocation* (**RMI**) je naziv za pozivanje metoda udaljenog objekta (zasnovan na RPC)
  - Postoje *stub*-ovi nazvani: *Proxy* i *Skeleton*
- Povezivanjem na udaljeni objekat se kreira **referenca na objekat**
  - Preko reference se pozivaju metode objekta
  - Reference se mogu prenositi između procesa na različitim mašinama



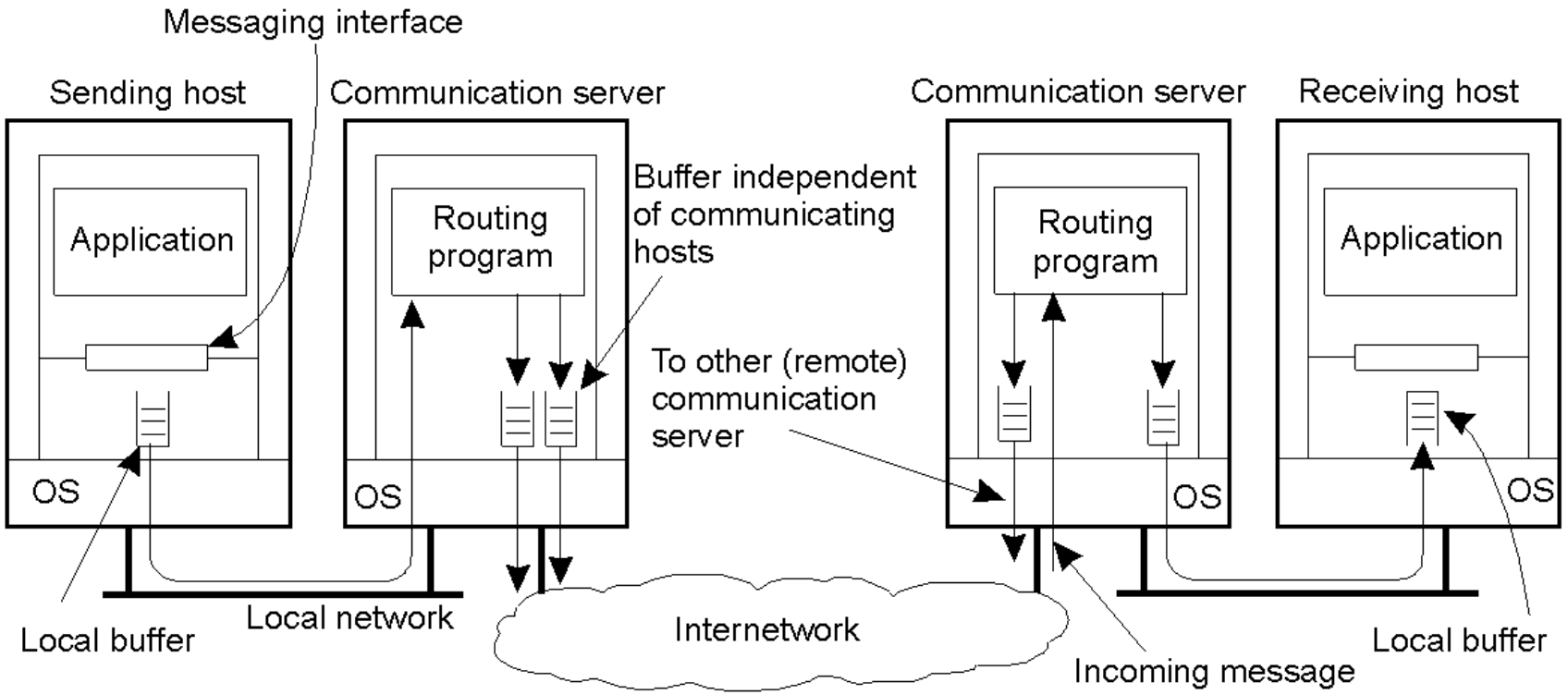
Komunikacija zasnovana na porukama

- RPC i RMI
  - Su jedan način komunikacije u DS
  - Nisu uvek pogodni
    - Šta ako server ne radi prilikom poziva?
    - Blokiranje klijenta dok poziv traje
- *Messaging* je drugi način
  - Komunkacija zasnovana na porukama
  - Primer:
    Pony Express



Perzistencija i sinhronizam u komunikaciji

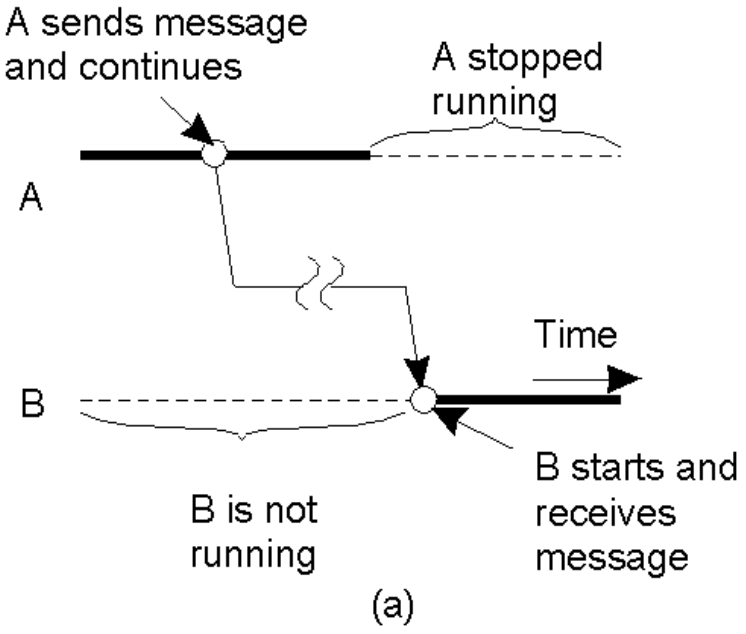
- Hostovi (pošiljaoc i primaoc) su povezani mrežom

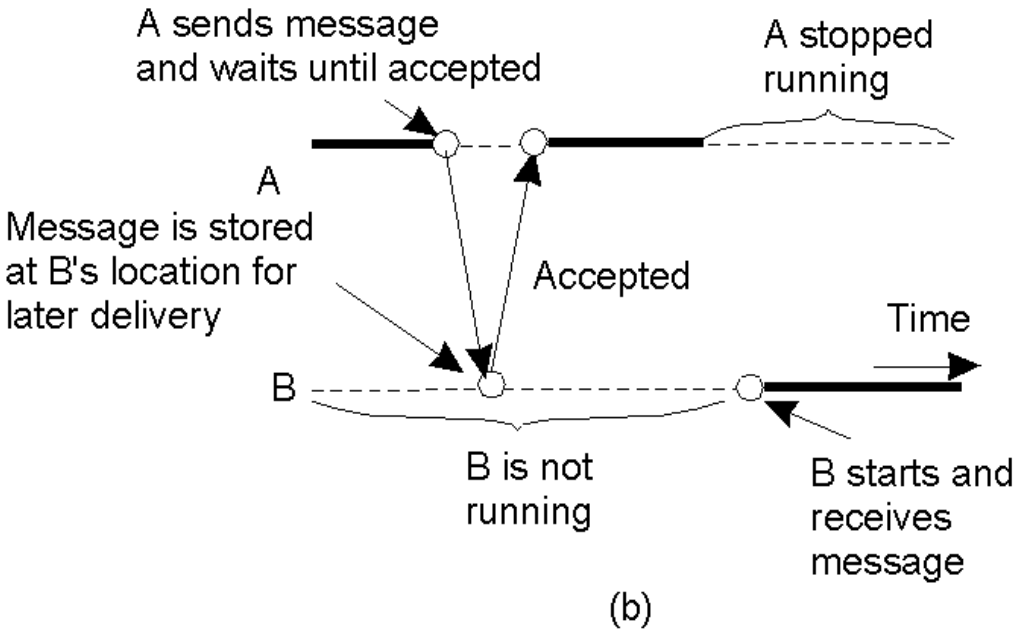


Perzistencija i sinhronizam u komunikaciji (2)

a) *Persistent asynchronous communication*

b) *Persistent synchronous communication*

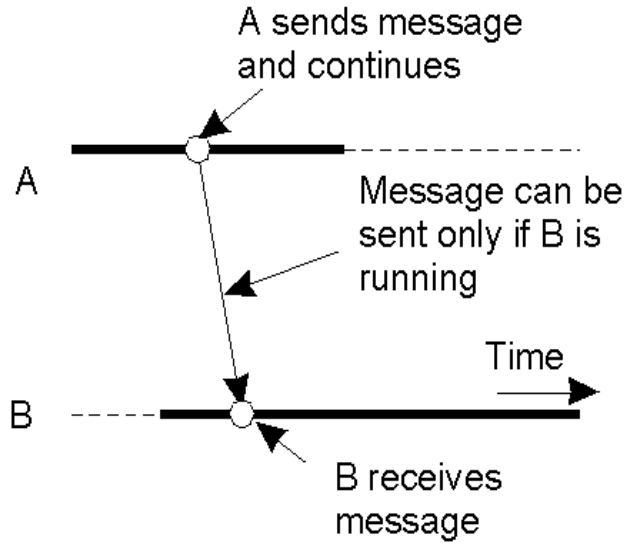




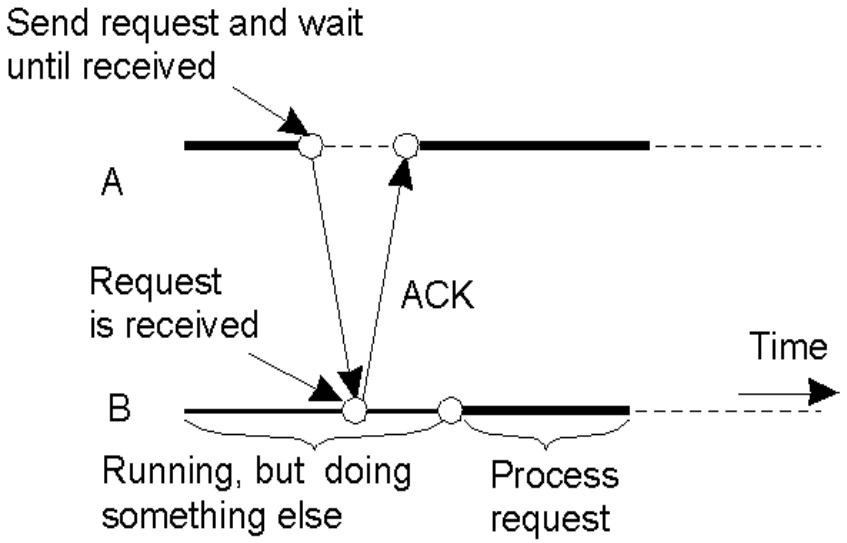
Perzistencija i sinhronizam u komunikaciji (3)

c) *Transient asynchronous communication*

d) *Receipt-based transient synchronous communication*



(c)

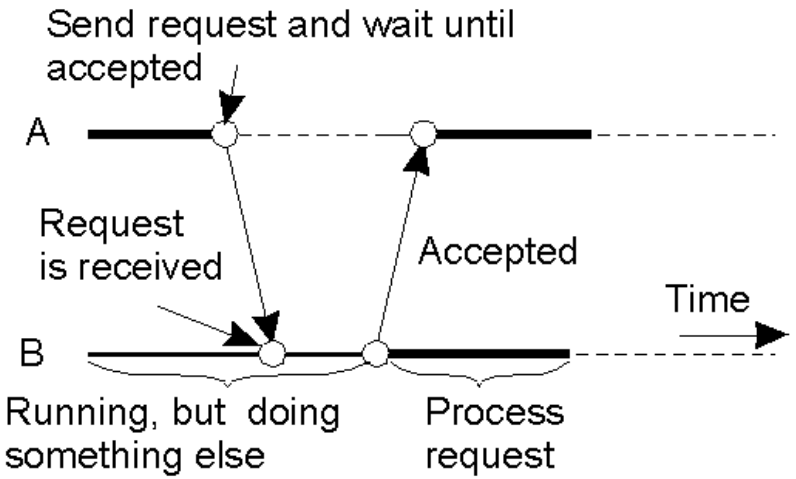


(d)

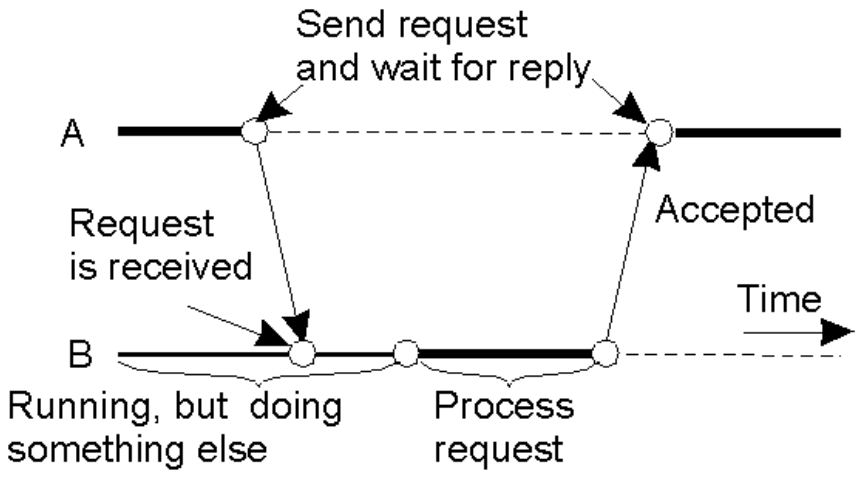
Perzistencija i sinhronizam u komunikaciji (4)

e) Delivery-based transient synchronous communication

f) Response-based transient synchronous communication



(e)



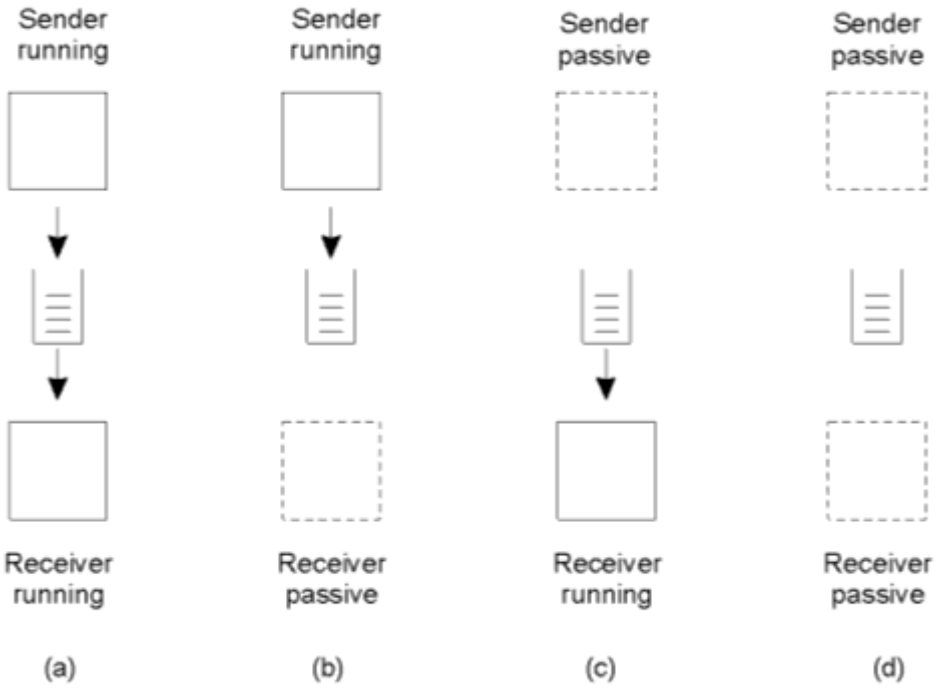
(f)

Message-Oriented Middleware (MOM)

- Naziva se i: *message-queuing system*
- Koristi *message*-orijentisanu perzistentnu asinhronu komunikaciju
  - Postoji skladištenje poruka
  - Ne zahteva se da pošiljaoc ili primaoc poruka bude aktivan
- Osnovna ideja:
  - Kod slanja poruka se upisuju u red poruka
  - Poruka se prenosi do odredišta preko niza servera
  - Isporuka u odredište čeka da ono bude raspoloživo (spremno)
- Posledice
  - poruka će biti isporučena, ali se ne zna kada
  - ne garantuje se da će biti pročitana (ili obrađena)
- Tipično svaka aplikacija ima svoj red za poruke koji čita, a druge aplikacije u njega šalju poruke
  - Više aplikacija može čitati isti red poruka

MOM (2)

- Dobra strana: slaba zavisnost pošiljaoca i primaoca
  Loosely-coupled communication
  - Primaoc ne mora da radi kada se poruka šalje
  - Pošiljaoc ne mora da radi kada se poruka obrađuje
  - Pošiljaoc i primaoc rade nezavisno



MOM (3)

- Adresa odredišta treba da je poznata = adresa reda poruka
- Postoji ograničenje na veličinu poruka (tipično)
- Osnovne rutine

  Put – dodaj poruku u red

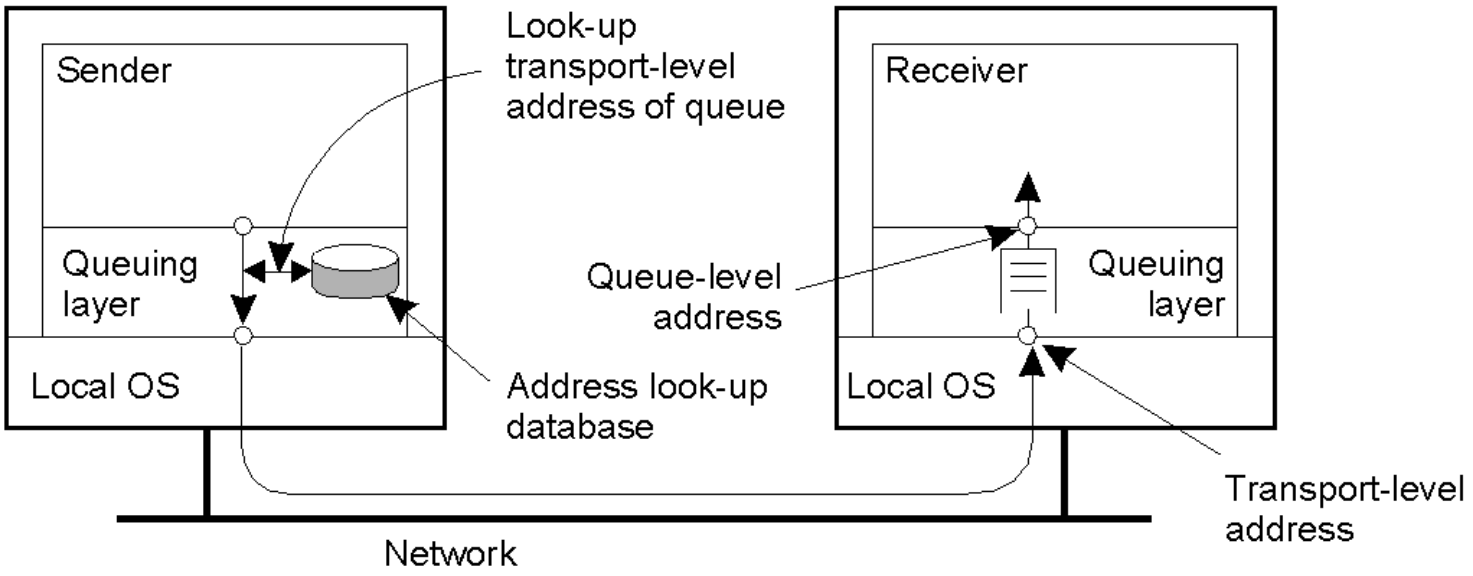
  Get – blokiraj ako je red prazan, inače izvadi prvu poruku

  Poll – proveri da li ima poruka i ako ima izvadi prvu (nema blokiranja)

  Notify – postavi metodu koja će biti pozvana kada pristigne poruka

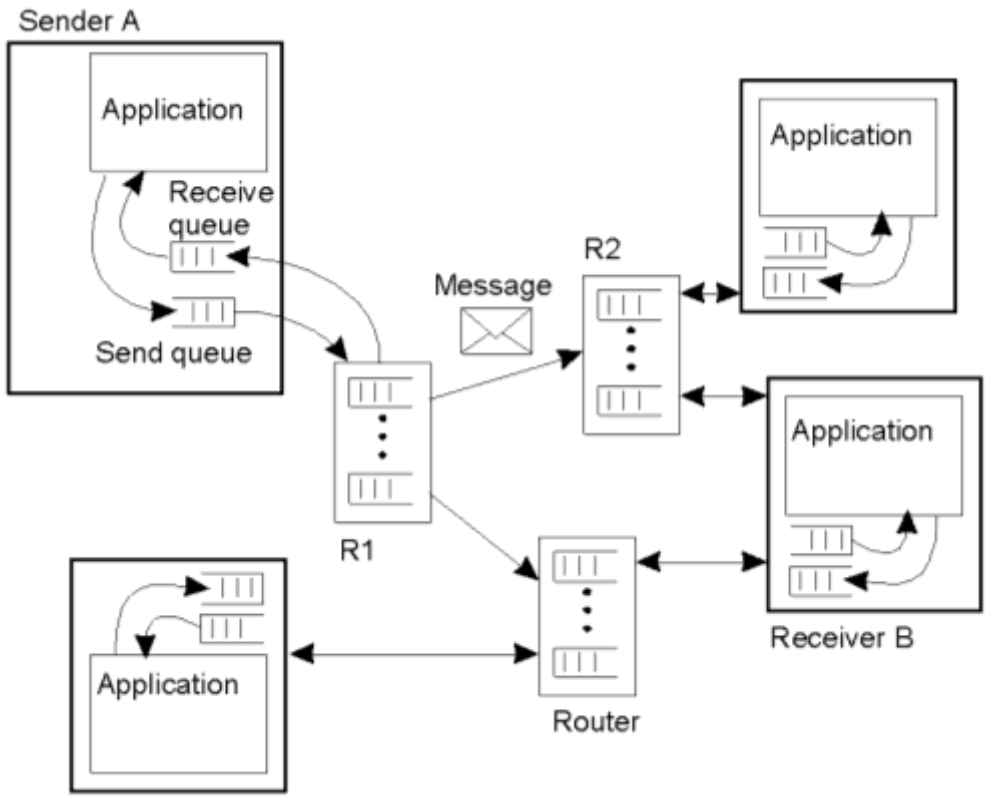
Generlana arhitektura MOM

- Adresa reda poruka (*queue-level address*) se mapira na adresu mašine (*transport-level address*)



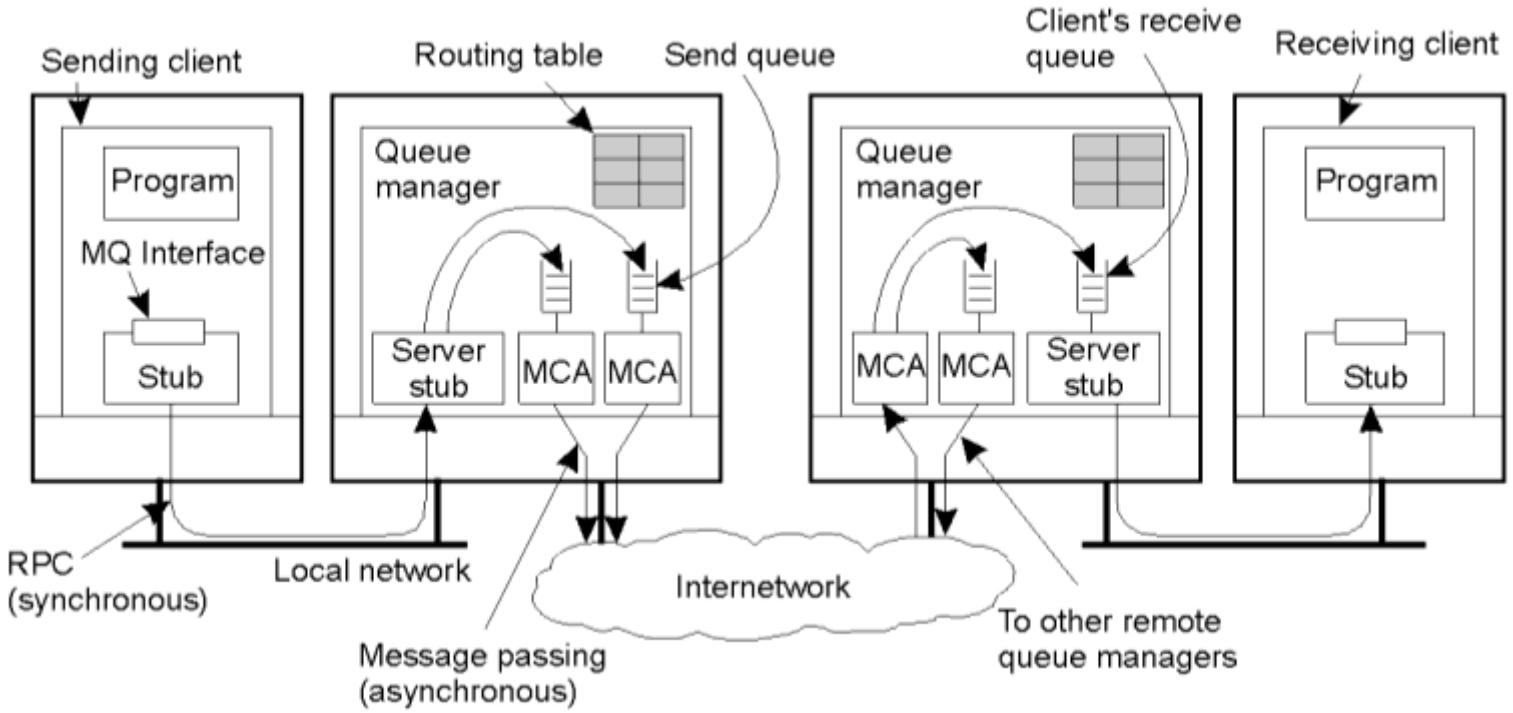
Upotreba rutera u MOM

- Rutiranje poruka omogućava skalabilan MOM
  - Preko adaptivne šeme mapiranja u ruterima (ili relejima)
- Omogućavaju
  - Sekundarno procesiranje (razlozi: *fault tolerance, security*)
  - Transformisanje sadržaja poruke
  - *Multicast* poruke



IBM MQSeries

- *Queue manager* (QM) upravlja redom (prazni red i šalje poruke u drugi red komunicirajući sa drugim QM)
- QM su uparerni preko *queue channel*-a (QC)
  - Jednosmeran, pouzdan prenos poruka
- Message channel agent (MCA) upravlja krajem kanala



IBM MQSeries MQI

- Programski interfejs - *Message Queue Interface* (MQI)

  MQopen – otvori (udaljeni) red poruka

  MQclose – zatvori red poruka

  MQput – postavi poruku u otvoren red poruka

  MQget – preuzmi poruku iz (lokalnog) reda poruka

  – Poruke se vade iz reda
    - na osnovu prioriteta
    - Po FIFO principu
    - Zahtevom za posebnu poruku
  – MQSeries omogućava slanje signala aplikaciji kada poruka dospe u red

Stream-orijentisana komunikacija

- Do sada su razmatrane komunikacije gde se razmenjuju kompletne informacije (i manje-više nezavisne)
  - Prenos se može obaviti brže ili sporije – vremensko kašnjenje nema uticaj na ispravnost (korektnost) podataka
- Postoji potreba za komunikacijama gde pravovremena isporuka ima suštinski značaj (vremenski zavisna informacija)
  - Prenos audio/video sadržaja
  - Postoje protokoli za *stream*-orijentisanu komunikaciju