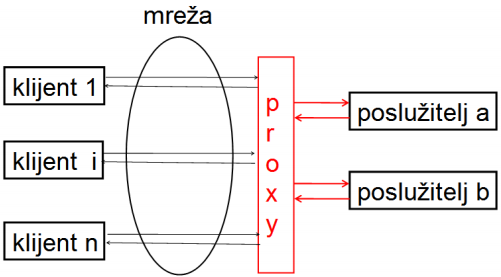
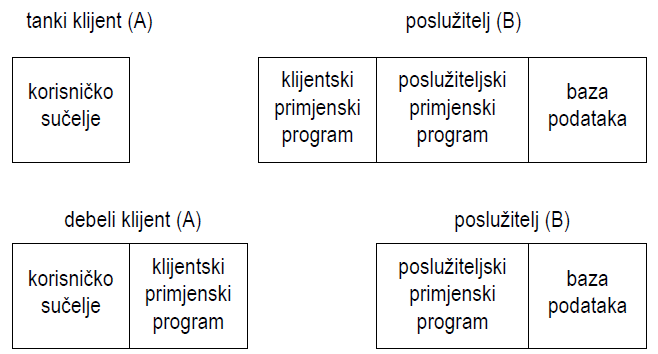
Raspodijeljeni sustavi

1. **Zahtjevi na raspodijeljene sustave:**
   1. **Otvorenost –** ako pruža usluge sukladno normiranim pravilima te definiranoj sintaksi i semantici.
   2. **Transparentnost** – ako prikriva odabrane značajke raspodijeljenog sustava
   3. **Skalabilnost** – ako posjeduje način prilagodbe povečanom broju korisnika i sredstava, njihovoj rasprostranjenosti i načinu upravljanja sustavom
   4. **Kvaliteta usluge** – nefunkcionalna obilježja (performanse, pouzdanost, raspoloživost i ukupni trošak vlasništva)
2. **Vrste transparentnosi**:
   1. **Transparentnost pristupa** – prikrivanje razlika u pristupu sredstvima i predočavanju podataka
   2. **Lokacijska transparentnost** – prikrivanje lokacije sredstva, pristup sredstvu putem imena
   3. **Migracijska transparentnost** – prikrivanje promjene lokacije sredstva na način da ta promjena ne utječe na način pristupa sredstvu
   4. **Relokacijska transparentnost** – prikrivanje premještanja / kretajnja sredstva
   5. **Replikacijska transparentnost** – prekrivanje više istovrsnih sredstava ili više preslika nekog sredstva
   6. **Konkurencijska transparentnost** – ako omogučava da više korisnika istodobno koristi isto sredstvo, a da pri tome oni sami toga nisu svjesni (prikrivanje istodobne uporabe istog sredstva od više korisnika)
   7. **Transparentnost na kvar** – rad sustava uz pojavu kvara (prikrivanje kvara)
3. **Zastupnik poslužitelja (proxy)**

****

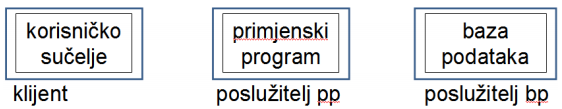
Zastupnik poslužitelja posreduje između klijenta i poslužitelja tako da od klijenta prikriva broj i lokaciju poslužitelja te način na koji su povezani (omogučava replikacijsku transparentnost). Ukoliko je vrijeme između dva zahtjeva za istim resursom relativno kratko zastupnik poslužitelja će imati kopiju tog resursa u priručnom spremištu te će ju isporučiti klijentu bez ponovnog kontaktiranja poslužitelja.

1. **Osnovni modeli raspodijeljene obrade:** 
   1. **Dvoredna arhitektura klijent-poslužitelj**

****

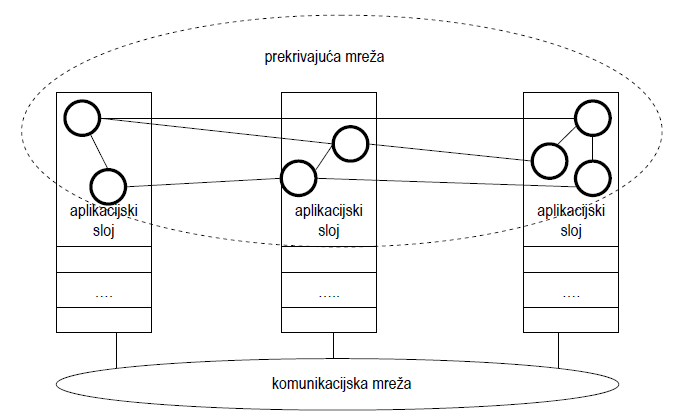
Korisničko sučelje, primjenski programi i baza podataka mogu biti na različite načine raspodijeljeni između klijenta i poslužitelja. U slučaju mršavog klijenta on sadrži samo korisničko sučelje, dok je na poslužitelju klijentski i poslužiteljski dio primjenskog programa te baza podataka. Složeniji klijent sa sučeljemi klijentskim programom naziva se debeli klijent. Odabir rješenja ovisi o namjeni sustava i raspoloživim sredstvima.

* 1. **Troredna arhitektura klijent-poslužitelj**

****

Primjer su aplikacije weba, gdje korisnički program koji se izvodi na klijentskom računalu nikad ne pristupa direktno bazi podataka, već posredno preko aplikacije weba. Klijentski program prikazuje korisničko sučelje i komunicira s aplikcaijom weba koja obavlja cijelokupnu logiku usluge i pristupa potrebnim podacima

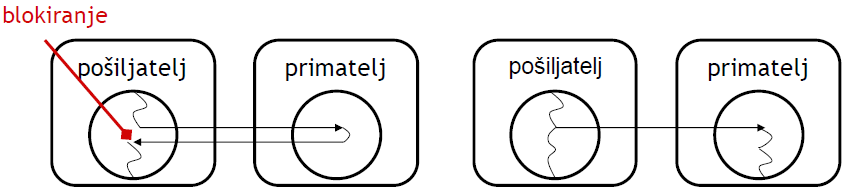
* 1. **Model ravnopravnog sudionika (P2P)**

****

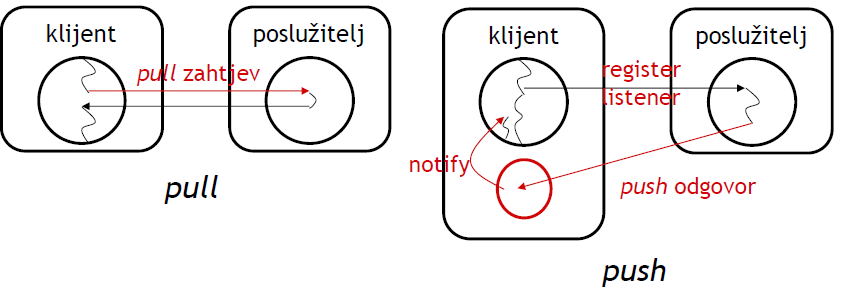
Ravnopravni sudionik je sudionik koji može djelovati i kao klijent i kao poslužitelj. Ravnopravni sudionici tvore sustav u kojem međusobno komuniciraju tako da se u aplikacijskom sloju povezuju u prekrivajuću mrežu čvorova. Ta mreža je izvedena iznad komunikacijske mreže koja povezuje računala na kojima se izvode peer procesi.

* 1. **Modeli premještanja programa i programskih agenata**
     + Kretanja procesa – klijent šalje kod poslužitelju, poslužitelj vrača odgovor sa rezultatom
     + Dostava koda na zahtjev – klijent nema izvršni kod, nego kada se javi potreba za obradom šalje zahtjev poslužitelju koji vraća traženi program
     + Aktivna poruka – poruka koja sadrži kod programa kreće se od čvora do čvora, da bi se kod izveo na zadanom čvoru
     + Programski agent – program koji obavlja neki posao za svog korisnika ili vlasnika, a raspolaže svojstvima kao što su inteligencija, samostalnost, društvenost... (pokretni ili stacionarni)

1. **Obilježja komunikacije:**
   1. **Konekcijska / bezkonekcijska** – Komunikacijski protokol je konekcijski ako procesi prvo razmjene kontrolne poruke, odnosno uspostave konekciju, i pri tome razmjene pravila i parametre komunikacije, a tek potom šalju podatke. Protokol je bezkonekcijski ako se ne razmjenjuju kontrolne poruke za uspostavu konekcije, već se prenose samo poruke s podacima.
   2. **Prezistentna / tranzijentna** – Kod perzistentne komunikacije procesa jamči se isporuka poruke ako pošiljatelj i primatelj nisu istovremeno dostupni, tako da se poruka pohranjuje u sustavu i isporučuje primatelju kada to bude moguće. Kod tranzijentne komunikacije ne jamči se isporuka poruke ako procesi nisu istovremeno dostupni.
   3. **Sinkrona / asinkrona** – Kod sinkrone komunikacije pošiljatelj je blokiran nakon slanja poruke sve do primitka potvrde o isporuci. Kod asinkrone komunikacije pošiljatelj nije blokiran te nastavlja procesiranje odmah nakon slanja.

****

* 1. **Pull / push** – Pull je klasični model zahtjev – odgovor.

****

1. **Obilježja procesa**
   1. **Vremenska ovisnost / neovisnost** – Vremenski ovisni procesi moraju biti istovremeno aktivni za realizaciju komunikacije.
   2. **Ovisnost / neovisnost o referenci sugovornika** – ako moraju znati jedinstveni identifikator (adresu) procesa s kojim žele komunicirati
2. **Obilježja modela**

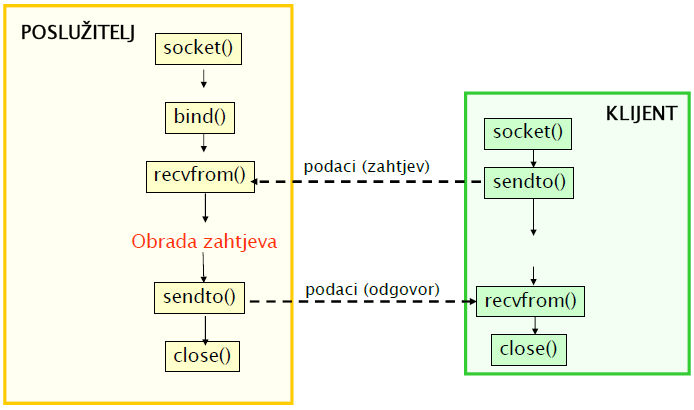
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Perzistentna | Sinkrona | Pull / push | Vremenska ovisnost | Ovisnost o referenci sugovornika |
| Socket UDP | N | N | Pull i push | D | D |
| Socket TCP | N | D | pull | D | D |
| RPC / RMI | N | D | pull | D | D |
| Razmjena poruka | D | N | Pull | N | D |
| Objavi - pretplati | D | N | push | N | N |

UDP – bezkonekcijska, nepouzdana

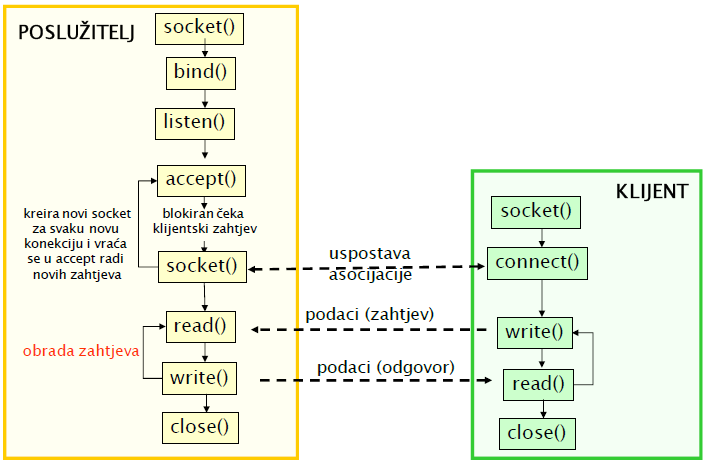
TCP – konekcijska, pouzdana

Objavi-pretplati – personalizacija primljenog sadržaja, proširivost sustava, skalabilnost.

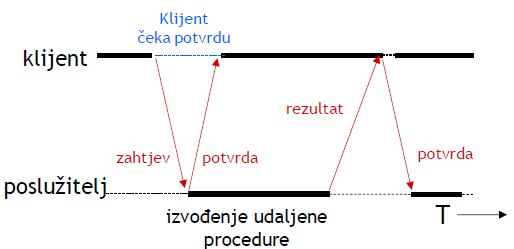
1. **Metode na klijentskoj i poslužiteljskoj strani socketa UDP:**

****

1. **Metode na klijentskoj i poslužiteljskoj strani socketa TCP:**

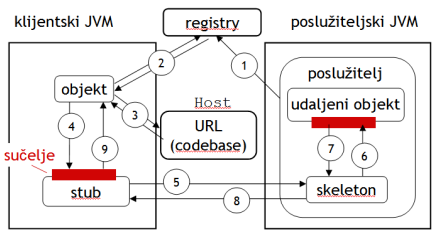
****

1. **Odgođeni sinkroni RPC (poziv udaljene procedure):**

****

Kod odgođenog sinkronog poziva udaljene procedure, klijent nije blokiran dok čeka rezultat izvođenja, već nastavlja s radom nakon uspješnog primitka potvrde. Kasnije mu poslužitelj šalje rezultat koristeči drugi asinkroni poziv udaljene procedure.

1. **Model pozivanja udaljene metode (Java RMI) kada se klasa stub učitava dinamički:**

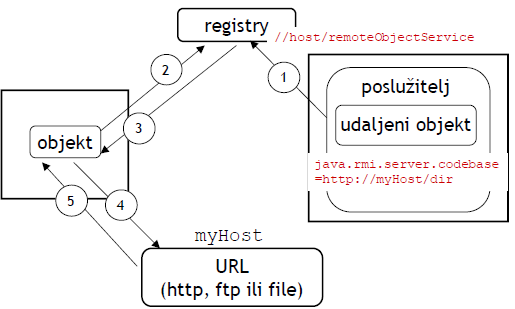
****

* 1. Poslužitelj definira codebase udaljenog objekta i registrira ga pod odabranim imenom.
  2. Klijent od regstrya traži referencu na udaljeni objekt, i dobiva je.
  3. Klijent traži i dobiva klasu stub koristeči codebase.
  4. Klijent poziva metodu stuba dostupnu na klijentskom računalu.
  5. Stub serijalizira parametre i šalje ih skeletonu.
  6. Skeleton deserijalizira parametre i poziva metodu udaljenog objekta.
  7. Udaljeni objekt vrača rezultat izvođenja metode skeletonu.
  8. Skeleton serijalizira rezultat i šalje ga stubu.
  9. Stub deserijalizira rezultat i dostavlja ga klijentu.

1. **Model pozivanja udaljene metode (Java RMI):**

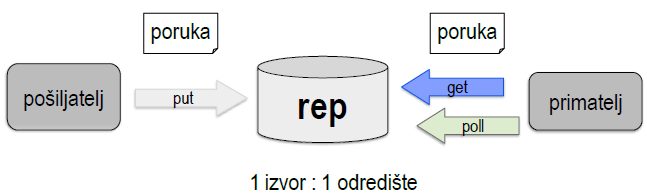
Isto kao u 12. Samo bez URL (codebase) i koraka 3.

1. **Dinamičko učitavanje stuba:**



* 1. Poslužitelj odabire codebase udaljenog objekta i registrira ga pod odabranim imenom
  2. Klijent od registrya traži referencu na udaljeni objekt koristeči registrirano ime
  3. Registry vrača referencu na klasu stuba. Ako se klasa stuba može nači na klijentskoj strani učitava se lokalna verzija klase. U suprotnom će klijent učitati klasu koristeči dobiveni codebase.
  4. Klijent traži klasu stuba koristeći codebase.
  5. Klasa stuba se dostavlja klijentu. Klijent može pozvati metode udaljenog objekta koristeči primljeni stub.

1. **Komunikacija porukama:**

****

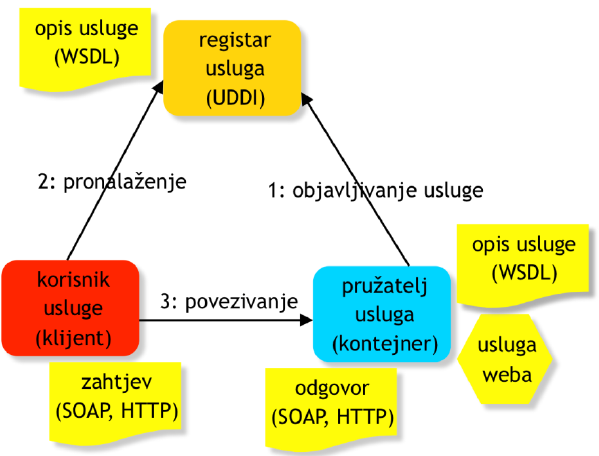
U komunikaciji između pošiljatelja i primatelja rep sudjeluje kao posrednik. Pošiljatelju se u načelu garantira isporuka poruke u primateljev rep, ali ne i isporuka poruke primatelju. Primatelj može pročitati poruku iz repa u bilo kojem budučem trenutnku.

1. **Operacije koje implementira programska infrastruktura dijeljenog podatkovnog prostora:**
   1. write(t) : dodaj tuple t u raspodijeljeni podatkovni prostor
   2. read(s) -> t : vraca tuple t koji odgovara predlošku s
   3. take(s) -> t : vraća tuple t koji odgovara predlošku s i briše ga iz podatkovnog prosstora
2. **Format protokola HTTP:**
   1. Početni redak – obavezan! Sastoji se od metode koja se koristi, zatim puta do resursa i inačice protokola
   2. Polja zaglavlja – Obavezno polje zaglavlja je „Host:“ koje sadrži ime poslužitelja sa kojeg se traži resurs. Ostala polja zaglavlja su opcionalna.
   3. Prazan redak – obavezan!
   4. Tijelo poruke – U tijelu poruke se prenosi sadržaj uspješno dohvačenog resursa. Nije obavezan.
3. **Svojstva sustava sa posrednikom:**
   1. Replikacijska transparentnost – zato što korisnik nije svjestan koji poslužitelj ga je zapravo poslužio
   2. Otpornost na kvarove – jer se kvar jednog poslužitelja može prikriti od korisnika
   3. Skalabilnost – zato što ovaj sustav posjeduje sposobnost prilagodbe povečanom broju korisnika
4. **Razlika između web-aplikacija temeljenih na CGI (common gateway interface) i poslužiteljskim skriptama.**

CGI je jednostavno sučelje za pokretanje eksterinh programa iz web-poslužiteja na platformski i programski nezavisan način. Kod svakog procesa se pokreče novi proces, a podaci između poslužitelja i procesa šalju se preko varijabli okoline i tokova podataka. Nakon svake obrade proces se gasi. Nedostatak CGI-a je što se kod svakog zahtjeva pokreče novi proces i nakon obrade gasi što je zahtjevno za resurse (procesorsko vrijeme i memoriju) pa kod velikog broja zatjeva na poslužitelju to znatno utječe na performanse.

Poslužiteljske skripte dinamički generiraju HTML dokumente poput CGI-ja ali je razlika u tome što se za svaki zahtjev ne pokreće novi proces i na taj se način štede resursi.

1. **Arhitektura web-usluge:**



Kada se na pružatelju usluge instalira web-usluga onda ju on registrira u registru usluga. Pomoću protokola UDDI pružatelj usluge objavi uslugu i registru šalje opis usluge u WSDL-u. Kada klijent želi pronaći uslugu prvo registru pošalje upit za pronalaženje, a registar mu odgovara opisom usluge koja odgovara zadanim kriterijima. Nakon toga klijnt se može povezati s pružateljem usluge jer ima opis uluge. Klijent šalje pomoću protokola HTTP i SOAP zahtjev za pozivom usluge, pružatelj usluge pokreće izvođenje usluge i šalje natrag rezultat u odgovoru isto tako pomoću HTTP-a i SOAP-a.

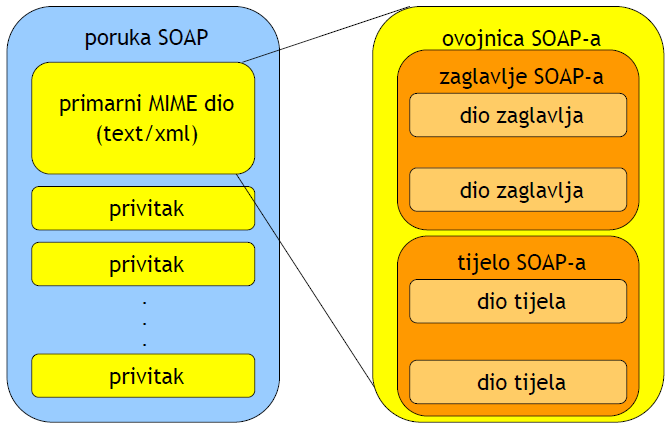
1. **Vrste web-usluga:**
   1. **Poziv udaljene procedure** – usko povezuje klijenta i poslužitelja jer je usluga usko vezana za programski jezik
   2. **Usluge temeljene na dokumentima / porukama** – ne povezuje jako klijenta i poslužitelja odnosno njihove implementacijske jezike jer je fokus na ugovorima koji su propisani WSDL-om.
   3. **Usluge temeljene na stanju resursa** - koristi protokol HTTP pa je sučelje dobro definirano, fokus je na interakciji s resursima koji imaju stanje, a ne na porukama ili operacijama.
2. **Protokol SOAP:**

Protokol SOAP omogućuje komunikaciju s web-uslugom.

Dva osnovna načina rada protokola SOAP su:

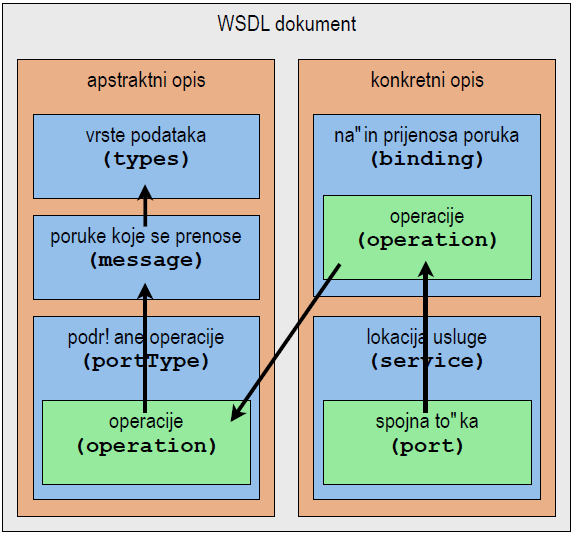
* 1. Poziv udaljene procedure – služi za prijenos serijaliziranih parametara i rezultata. Korist ovog načina rada je dobro definirano sučelje i tipovi podataka, te prilagodni kod može biti generiran automatski.
  2. Razmjena dokumenata / poruka – koriste se XML dokumenti za razmjenu poruka.

1. **Format poruke protokola SOAP:**



Poruka se sastoji od primarnog dijela (MIME – txt/xml) koji je osnovni dio svake poruke. Nakon toga mogu dolaziti različiti privici. Taj primarni dio se sastoji od ovojnice SOAP-a unutar koje su dva dijela : zaglavlje i tijelo.

1. **Format poruke protokola WSDL:**



Opis web-usluge sastoji se od apstraktnog i konkretnog dijela. Dijelovi se referenciraju i na taj način su povezani.

* 1. Types – definira vrste podataka neovisne o platformi i jeziku
  2. Message – definiraju ulazne i izlazne poruke koje se mogu koristiti kao parametri usluga
  3. Operation – predstavlja jednu operaciju/metodu/proceduru koja je definirana u usluzi, a sastoji se od definicija ulaznih, izlaznih i iznimnih poruka koje se mogu razmjenjivati
  4. Binding – definira kako je konkretna implementacija povezana s operacijama u apstraktnom opisu i definira formt u kojem će se poruke prenositi
  5. Service – definirani URI preko kojeg se usluga može pozvati

1. **Vremenska složenost:**

Mjeri se brojem izvedenih koraka koji dovode do završnog stanja algoritma, tj do stanja u kojem su svi procesi zaustavljeni ili kada se više ne proizvodi novi ulaz.

1. **Komunikacijska složenost:**

Mjeri se broj kreiranih i poslanih poruka.

1. **Za koje je svojstvo raspodijeljenih sustava značajna komunikacijska složenost?**

Za skalabilnost sustava. Na temelju komunikacijske složenosti možemo zaključiti kako raste generirani promet raspodijeljenog sustava s rastom tog sustava.

1. **Uzročna slijednost (CO):**

Uzročna slijednost osigurava da uzročno povezani događaji slanja dviju poruka istom primatelju rezultiraju primanjem u slijedu kojim su poslani.

1. **Sinkrona slijednost:**

Osigurava da se slanje i primanje poruke događa istovremeno.

1. **FIFO i non-FIFO:**

FIFO kanal čuva redosljed poruka, odnosno da se ponaša kao rep. Non-FIFO kanal ne čuva redosljed poruka, odnosno ponaša se kao skup.

1. **Događaji su uzročno povezani ako:**
   1. Su slijedni događaji na istom procesu
   2. Postoji slanje i primanje poruke između njih
   3. Postoji tranzitivna uzročnost