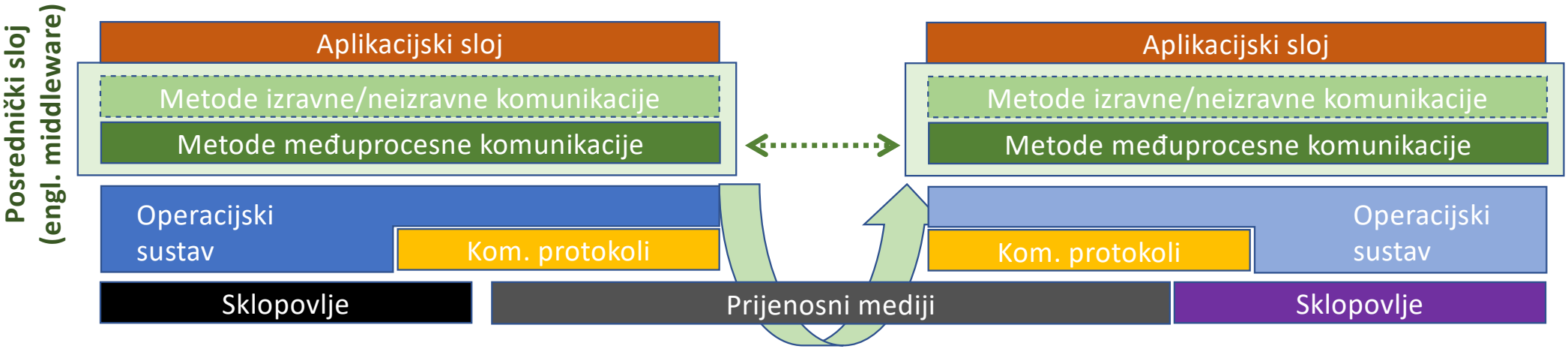
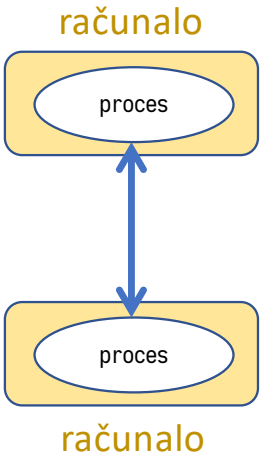
Otvoreno računarstvo

7. Raspodijeljeni sustavi

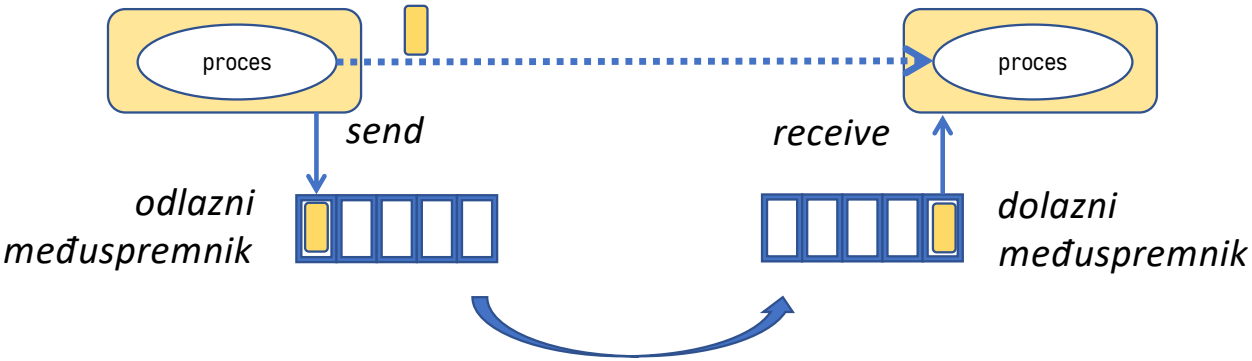
Metode međuprocesne komunikacije



- Metode međuprocesne komunikacije
  - Niska razina apstrakcije (poruke ili tokovi/cjevovodi podataka)
    - *send/receive* primitive,
    - utičnice (engl. *sockets*),
    - višesmjerno (i svesmjerno) odašiljanje (engl. *multicast, broadcast*)
    - prekrivne mreže (eng. *overlay networks*)
  - Komunikacija između procesa i računala korištenjem mehanizama i protokolima za mrežnu komunikaciju
    - Međuprocesna komunikacija: cjevovodi, dijeljena memorija, signali, …
    - Mrežna komunikacija: prevladava TCP/UDP, IP

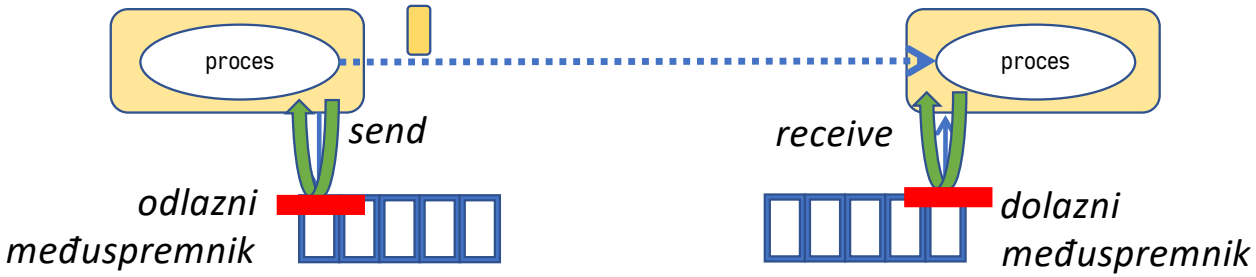


send/receive primitive (I)



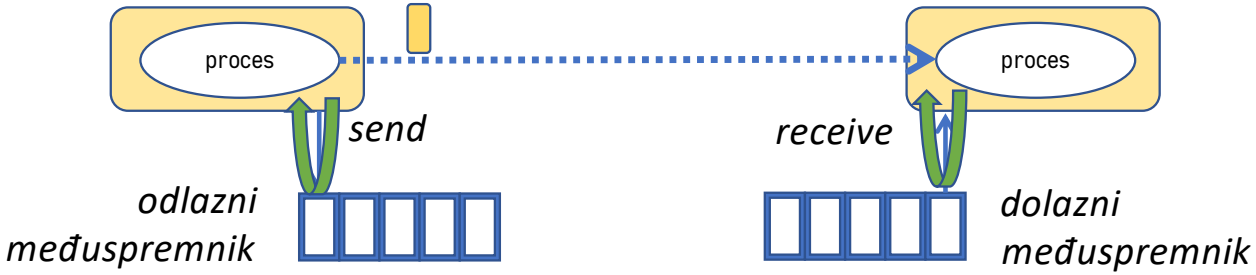
- Send: šalje poruku u odlazni međuspremnik
  - sloj operacijskog sustava i komunikacijskih protokola
  - poruka ne mora odmah biti poslana komunikacijskim kanalom
- Receive: dohvat poruke iz dolaznog međuspremnika
  - sloj operacijskog sustava i komunikacijskih protokola
  - međuspremnik može biti prazan …
- Primitive su uglavnom konceptualne naravi – implementacija nad stvarnim protokolima mrežne komunikacije transportnog sloja (npr. TCP ili UDP)

send/receive primitive – sinkrone (II)



- Sinkrone operacije – dretva/proces blokiran do okončanja operacije
  - *send*: do uspješnog čitanja poruke od strane ciljnog procesa
    - To je osnovni mehanizam sinkronizacije procesa u raspodijeljenom sustavu
  - *receive*: do dohvaćanja poruke iz međuspremnika
    - međuspremnik može biti prazan -> čekanje na dolazak poruke
    - mora postojati mehanizam isteka vremena čekanja (*engl. timeout*)

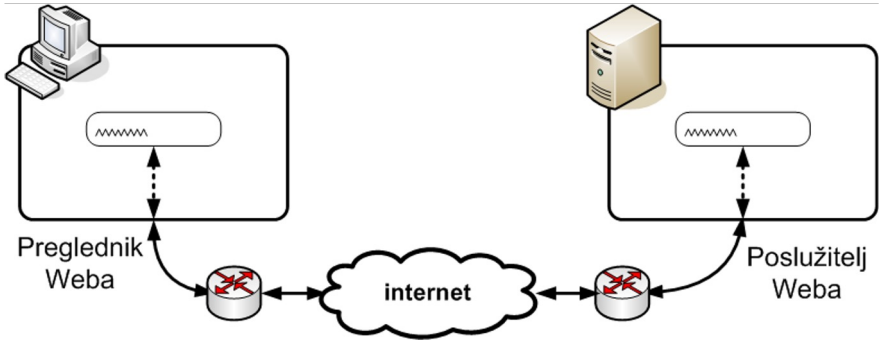
send/receive primitive - asinkrone (III)



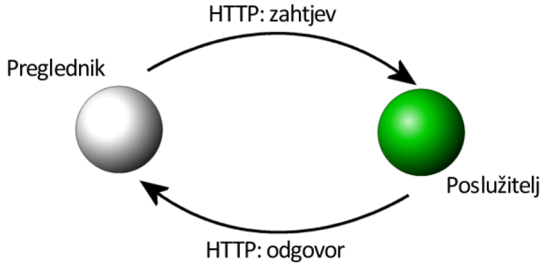
- Asinkrone operacije – dretva/proces odmah nastavlja izvođenje
  - *send*: poruka je uspješno pohranjena u međuspreminik
    - Ne podrazumijeva uspješan primitak paketa od drugog procesa !!!
  - *receive*: povratak iz rutine neovisno o postojanju poruke u međuspremniku
    - dohvaća poruku ako postoji u međuspremniku
    - registracija *callback* rutine pozivane u trenutku dolaska poruke

Komunikacija udaljenih procesa

- Preglednik Weba i poslužitelj Weba su **procesi** unutar operacijskog sustava računala klijenta i poslužitelja
- Komunikacija između procesa
  - na istom računalu
  - na udaljenim računalima

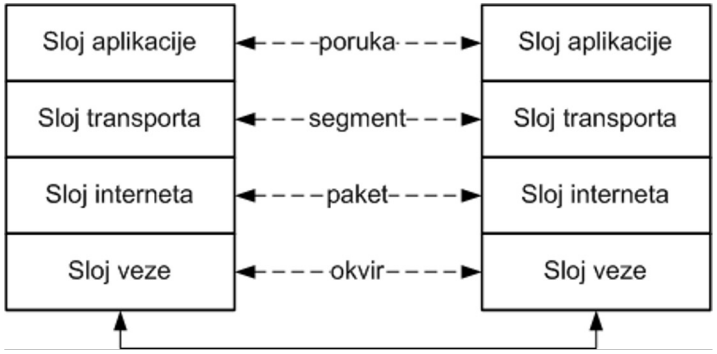


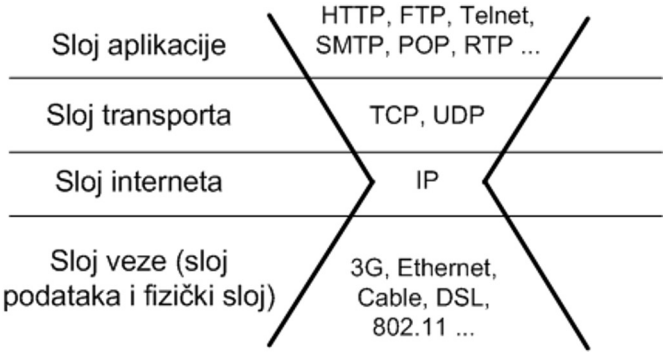
- Za uspješno izvođenje komunikacija potrebno je:
  1) locirati procese
  2) ostvariti komunikacijski kanal
  3) koristiti zajednički jezik komunikacije



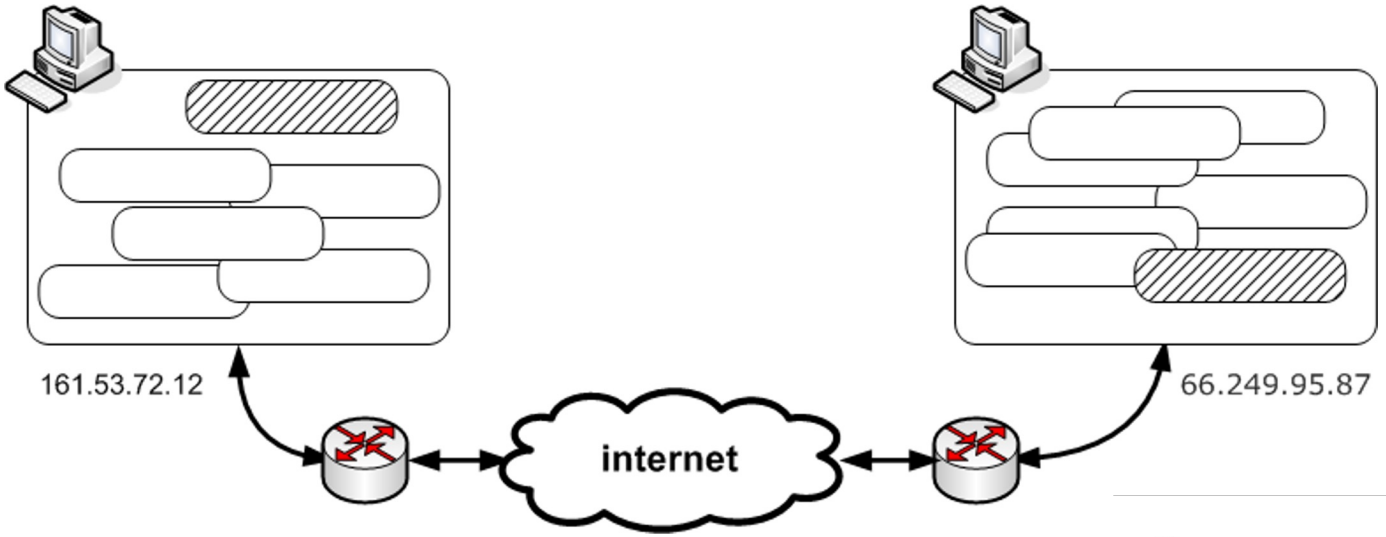
TCP/IP stog protokola

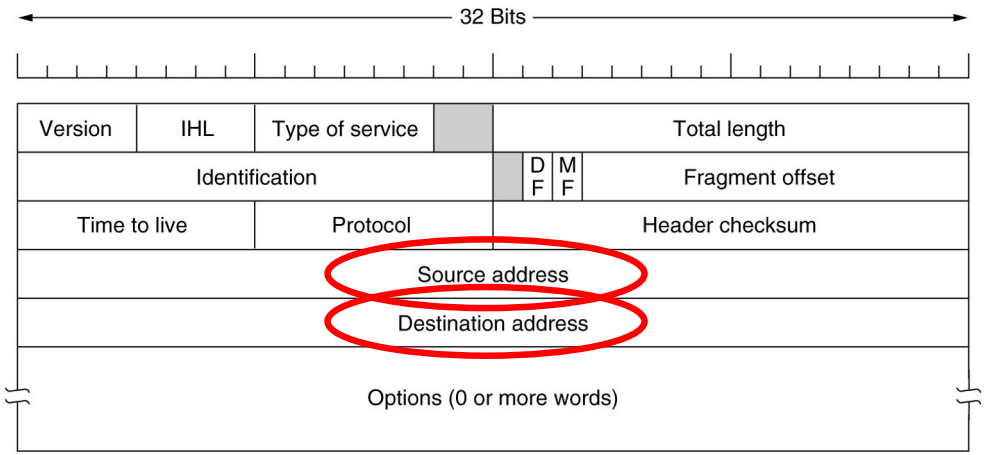
- Internet temeljen na **TCP/IP stogu** protokola
  - IP je temelj interoperabilnosti na internetu
  - neki slojevi ISO/OSI modela nisu implementirani
  - slojevi veze, interneta i transporta ugrađeni u operacijske sustave računala
  - sloj aplikacije implementiran unutar pojedinih aplikacija ili kao dodatna biblioteka funkcija



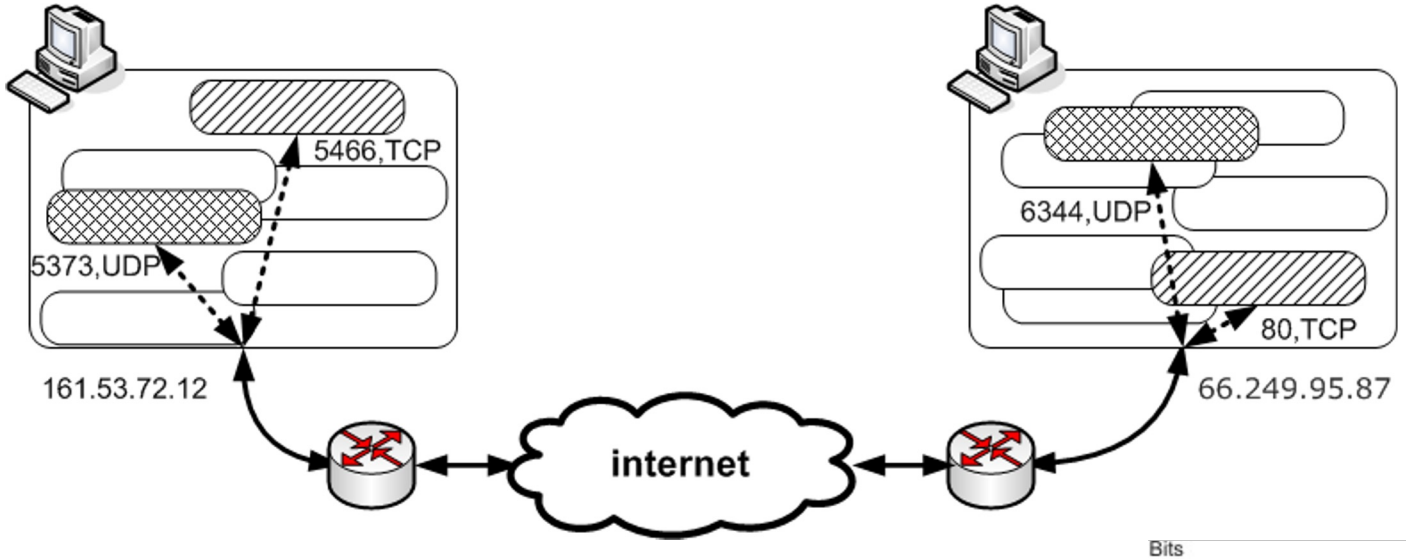


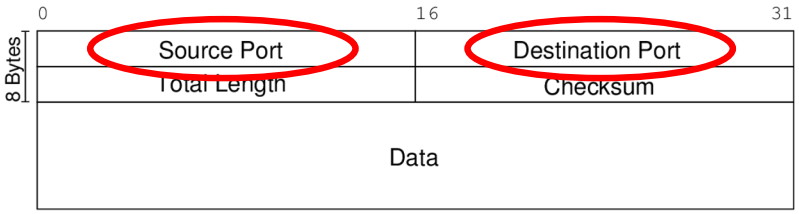
Usmjeravanje paketa između računala



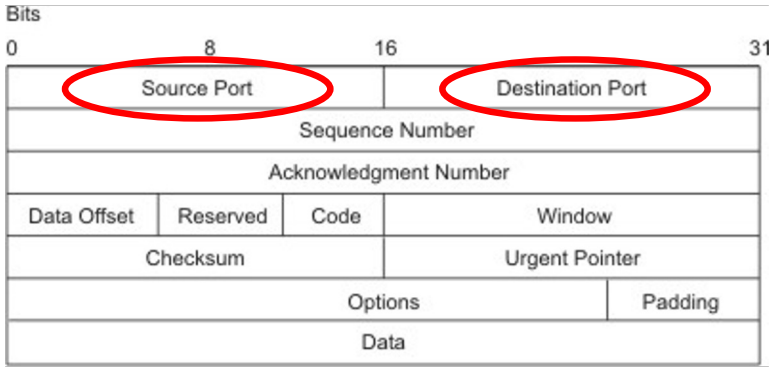


Dostavljanje paketa procesima





zaglavlje UDP paketa

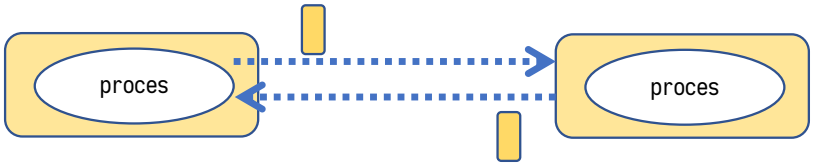


zaglavlje TCP paketa

Transportni sloj (TCP i UDP)

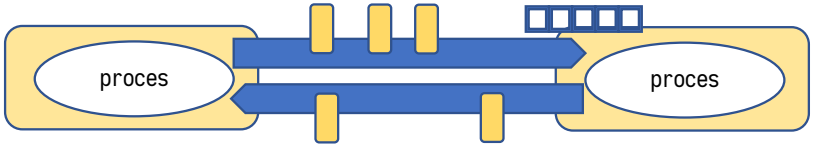
UDP: razmjena datagrama

send: neblokirajući
receive: blokirajući, neblokirajući



TCP: dvosmjerni cjevovod

send: blokirajući, neblokirajući
receive: blokirajući, neblokirajući

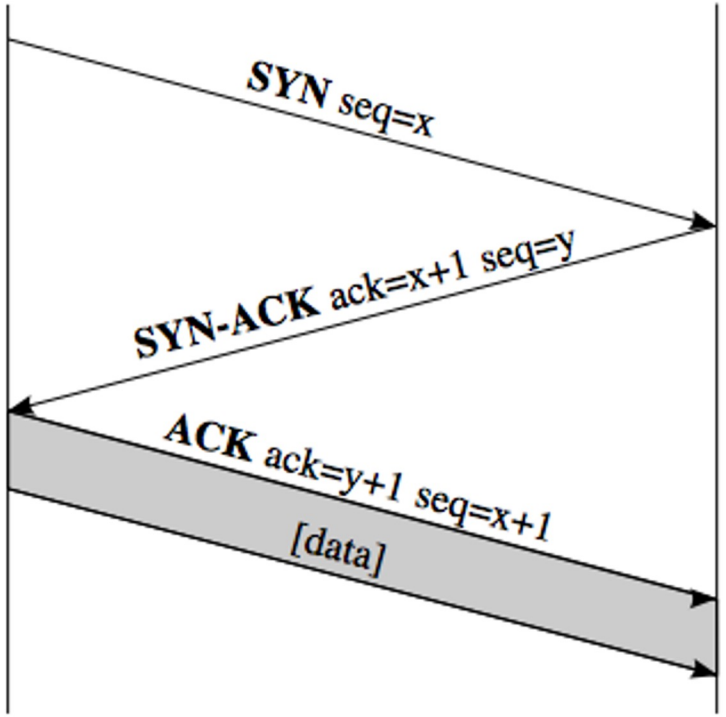


Red dolaznih zahtjeva za uspostavom veze

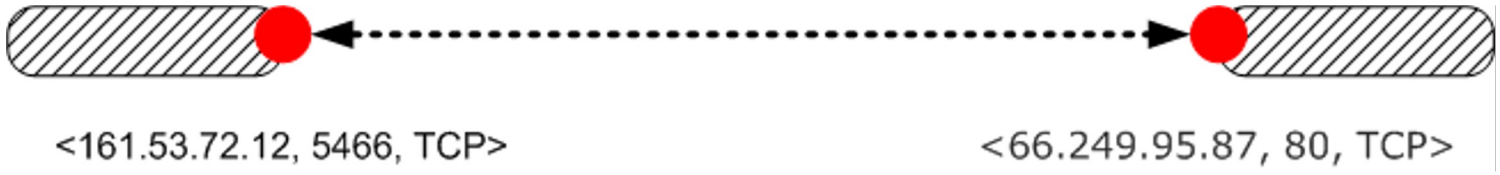
	TCP	UDP
Vrsta veze	spojna, *duplex*	bezspojna
Jedinica prijenosa	niz znakova/okteta	poruka (datagram)
Pouzdanost	okteti prenijeti bez dupliciranja, pouzdano i u redosljedu slanja	poruke mogu biti izgubljene, promijenjenog poretka, duplicirane
Veličina prenošenog sadržaja	neograničeno	ograničeno veličinom datagrama (MTU?)
Upravljanje zagušenjem	pošiljatelj se prilagođava stanju u mreži	nema upravljanja
Upravljanje tokom	pošiljatelj se prilagođava primatelju	nema upravljanja

Stvaranje TCP veze

- Razmjena tri SYN / ACK paketa za uspostavu veze
- Proces učenja pošiljatelja optimalnoj brzini slanja paketa
- Opcionalni *keep-alive* paketi tijekom neaktivne veze
- Nije poželjno učestalo stvaranje i raskidanje veze, prijenos male količine podataka tijekom trajanja veze



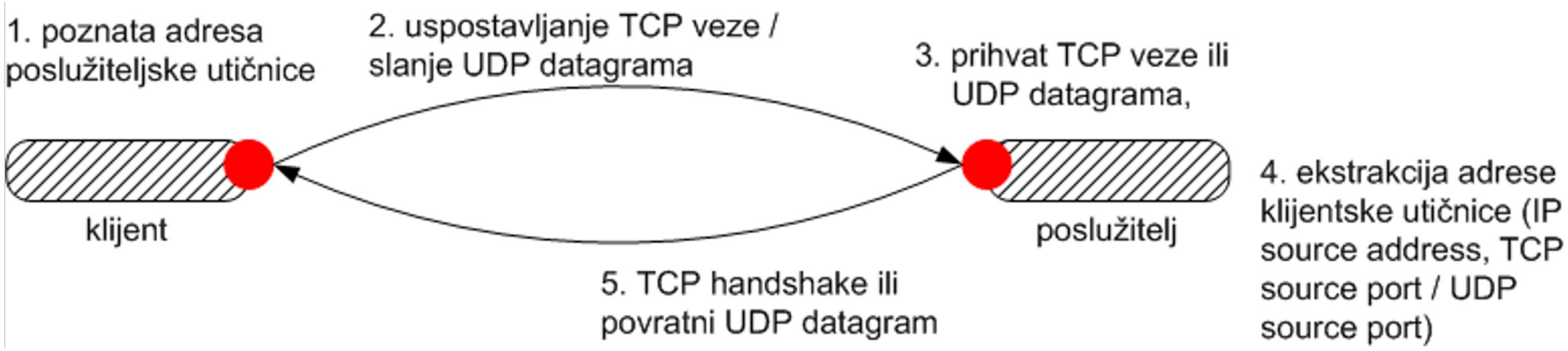
Utičnice



`<IP adresa, TCP/UDP port, protokol>`

- Utičnica (*engl. socket*) - programska apstrakcija krajnje točke komunikacije (*engl. communication endpoint*)
- Utičnica jedinstveno određena **IP adresom, portom i protokolom**
  - IP adresa = 32-bitni cijeli broj (IPv4), 128-bitni cijeli broj (IPv6)
  - port = 16-bitni cijeli broj
  - Protokol = TCP, UDP, ??? (u podlozi ne mora biti TCP/IP)

Klijent i poslužitelj



- Klijent
  - TCP: inicira vezu prema poslužitelju (*connect*), razmjenjuje nizove okteta putem cjevovoda (*read, write*)
  - UDP: izravno šalje datagrame poslužitelju, zaprima datagrame s odgovorima (*send, receive*)
  - Klijent mora prethodno poznavati adresu procesa poslužitelja (*socket*)

- Poslužitelj
  - TCP: osluškuje zahtjeve za uspostavom veze (*listen*), ostvaruje komunikacijski kanal s klijentom (*accept*) i razmjenjuje nizove okteta dvosmjernim kanalom (*read, write*)
  - UDP: zaprima datagrame zahtjeva i odgovara klijentu (*receive, send*)

URI i utičnice

`http://www.fer.unizg.hr/predmet/or`

HTTP koristi spojnu vezu - TCP

Uz pomoć usluge DNS se simbolička adresa poslužitelja pretvara u IP adresu

podrazumijevani port 80

`<161.53.72.12, 80, TCP>`

Alokacija portova

- Standardne usluge imaju alocirane portove
  - jednostavnije pronalaženje procesa usluge
- Portovi < 1024 dostupni samo procesima s posebnim ovlastima
- Portovi >= 1024 dostupni svim procesima
  - korisničkim poslužiteljskim procesima
  - operacijski sustav privremeno dodjeljuje portove klijentima

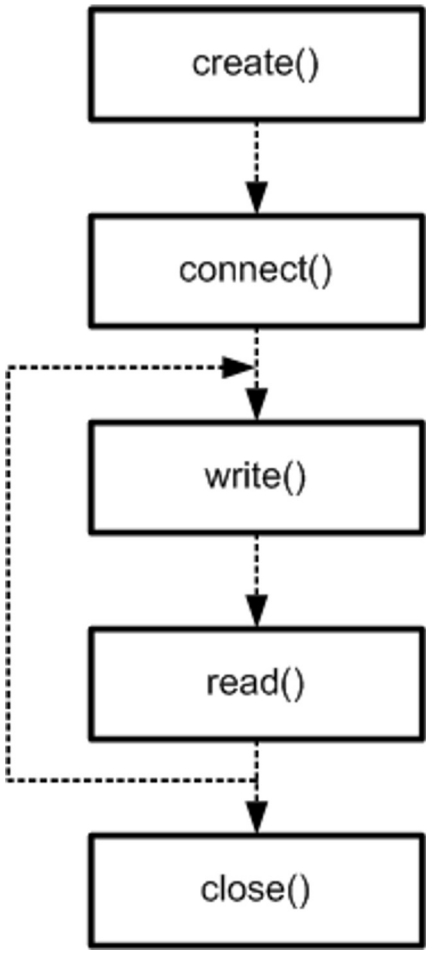
Port	Protokol	Namjena
20, 21	**FTP**	prijenos datoteka između računala
22	**SSH**	sigurna prijava na računala
25	**SMTP**	elektronička pošta
80	**HTTP**	WWW
110	**POP-3**	pristup korisnika elektroničkoj pošti
143	**IMAP**	pristup korisnika elektroničkoj pošti
443	**HTTPS**	HTTP preko sigurne veze (TLS/SSL)

Berkeley Sockets API

- Općenit mehanizam međuprocesne komunikacije
  - između procesa na istom računalu
  - između procesa na različitim računalima

- Berkeley sockets API: 4.2 BSD UNIX (1983)
  - API – apstrakcija mrežnih priključnica, jezik C
  - licencirano do 1989 – AT&T
  - *de facto* standard
  - ekvivalentni API postoje za većinu programskih jezika

Klijent – spojna veza



Stvaranje utičnice (TCP, IPv4/IPv6/lokalno)

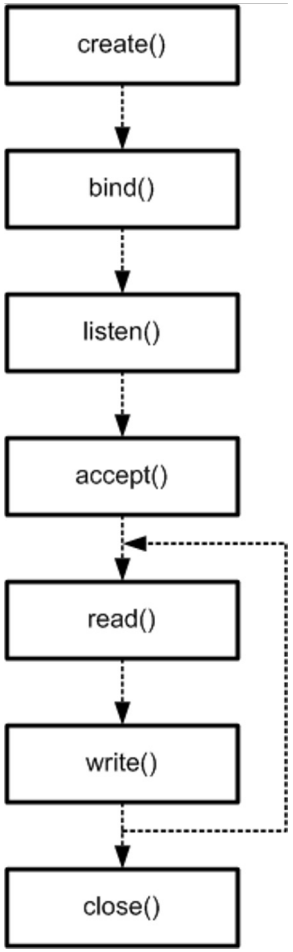
Spajanje na poslužiteljevu utičnicu (adresa utičnice)
[utičnici se dodjeljuje privremeni port]

Pisanje u izlazni međuspremnik kom. kanala

Čitanje iz ulaznog međuspremnika kom. kanala

Zatvaranje klijentske utičnice

Poslužitelj – spojna veza



Stvaranje utičnice (TCP, IPv4/IPv6/lokalno)

Vezanje utičnice na lokalnu IP adresu* i port
(*računalo može imati više mrežnih sučelja)

Postavlja utičnicu u poslužiteljski mod rada,
određuje veličinu reda dolaznih zahtjeva za vezom

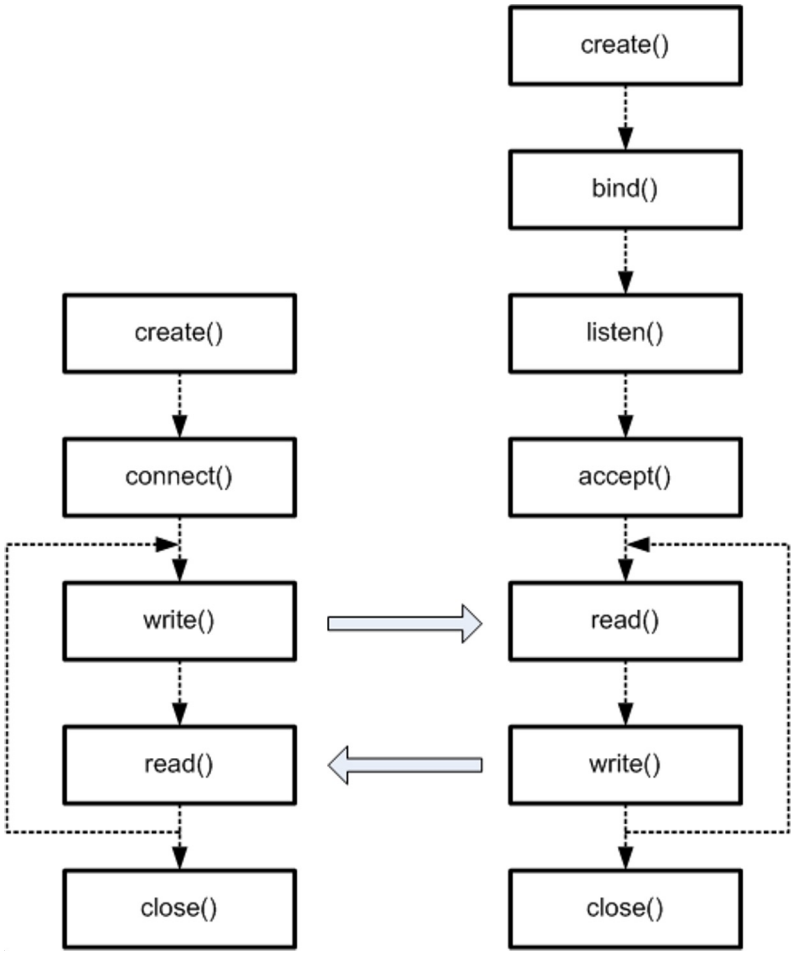
Prihvaćanje veze iz reda dolaznih zahtjeva ili
blokiranje izvršavanja procesa do dolaska zahtjeva,
prihvaćenoj vezi se **dodjeljuje privremena utičnica**

Čitanje iz ulaznog međuspremnika kom. kanala

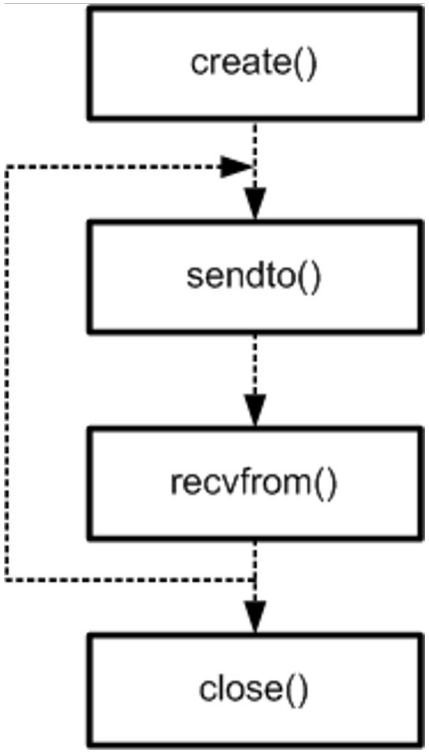
Pisanje u izlazni međuspremnik kom. kanala

Zatvaranje privremene utičnice / zatvaranje poslužiteljske utičnice

Interakcija klijenta i poslužitelja



Klijent – bezspojna veza



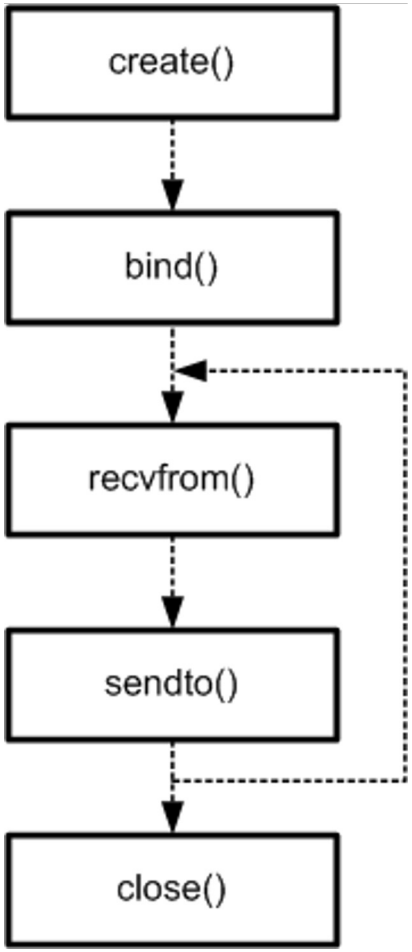
Stvaranje utičnice (UDP, IPv4/IPv6/lokalno)

Slanje datagrama prema nespojnoj utičnici

Prihvaćanje datagrama s nespojne utičnice

Zatvaranje utičnice

Poslužitelj – bezspojna veza



Stvaranje utičnice (UDP, IPv4/IPv6/lokalno)

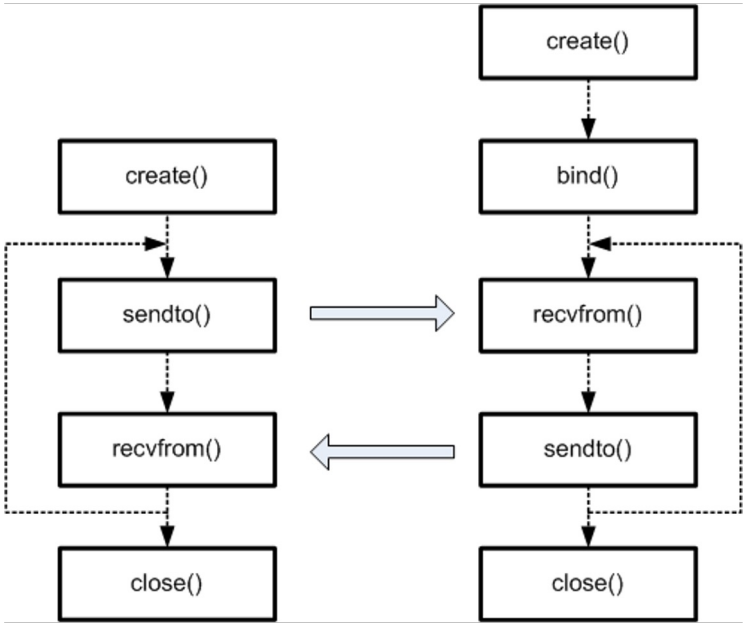
Vezanje utičnice na lokalnu IP adresu* i port
(*računalo može imati više mrežnih sučelja)

Prihvaćanje datagrama s nespojne utičnice

Slanje datagrama prema nespojnoj utičnici

Zatvaranje utičnice

Interakcija klijenta i poslužitelja



Procesni modeli na strani poslužitelja

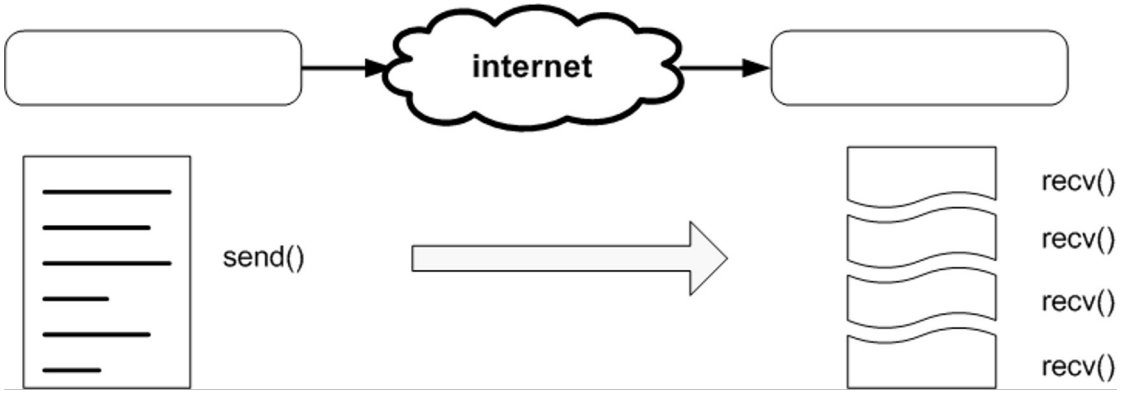
- Spojna veza:
  - Istovremeno samo jedan klijent
  - Proces ili dretva po vezi
  - Proces ili dretva po klijentu
  - ...

- Nema trajnih veza s klijentskim procesima
  - svaki prispjeli datagram je neovisan, može biti iz različitog procesa
  - datagrami se mogu obrađivati u zasebnim dretvama ili procesima ...

Čitanje podataka iz spojne veze

- Klijent šalje sadržaj u komunikacijski kanal (tj. međuspremnik)

- Na strani poslužitelja naredba `recv()` (*read*) čita sadržaj **dolaznog međuspremnika**
  - može biti potrebno više naredbi čitanja za dohvat čitavog sadržaja upita / odgovora
  - mogući uzroci
    - kašnjenja unutar mreže / različiti putovi paketa
    - fragmentacija paketa
    - gubitak/retransmisija paketa
    - ...tko zna?



Uokvirenje poruka

- Bezspojna veza – poruka je sadržana u datagramu
- Spojna veza – poruka je niz okteta
  - Kako detektirati kraj poruke?
- Detekcija kraja jednorednih poruka je jednostavna …
  - npr. REQUEST index.html**<CR><LF>**
- Problem detekcije kraja **duljih** poruka
  - npr. prenošene binarne datoteke ili duljeg teksta
- Tri osnovne metode uokvirenja poruka:
  - **umetanjem** okteta (*engl. octet stuffing*)
  - **brojanjem** okteta (*engl. octet counting*)
  - **uništavanjem veze** (*engl. 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 prijamu 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 pošiljatelj 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

Mrežno programiranje

- Primjer Socket IPC za platformu Java
  - http://www.cdk5.net/ipc
- Službene upute za Java socket API:
  - https://docs.oracle.com/javase/tutorial/networking/sockets/index.html
- Socket API za node.js:
  - https://nodejs.org/api/net.html

Otvoreno računarstvo

7. Raspodijeljeni sustavi

Zašto vanjska reprezentacija podataka (XDR)?

- Komunikacija dvaju procesa – razmjena informacija – u kojem formatu??
- Različitosti zapisa podataka izviru iz:
  - Sloja platforme: procesor i operacijski sustav
    - big/little endian
    - širina riječi
  - Sloja aplikacije – korištenog programskog jezika:
    - Zapis niza znakova (\0 ili zapis duljine)
    - Korištenog kodiranja za zapis niza znakova
    - Podržanih jednostavnih tipova podataka (short, unsigned, double, void, boolean …)
    - Podrazumijevane preciznosti zapisa jednostavnih tipova podataka
    - Podržanih složenih tipova podataka (map, array, tuple, dictionary, objekti …)
    - …

Enkodiranje i dekodiranje zapisa

- *Encoding/marshalling/serijalizacija/flattening* – pretvorba strukture podataka (jednostavnih, složenih tipova podataka, objekata …) u binarni ili tekstni zapis
- *Decoding/unmarshalling/deserijalizacija/rebuilding* – rekonstrukcija (ekvivalentne) strukture podataka iz binarnog ili tekstnog zapisa

Pristupi prenosivosti zapisa strukture podataka

1. „as-is”, „kaj to treba?”, „nisam razmišljao o tome!” i slično
   - Pošiljatelj i primatelj dijele istu platformu i jezik implementacije, koristi se uobičajeni (podrazumijevani) format zapisa (npr. Java serijalizacija, json.stringify() …)

2. **Normirani format vanjske reprezentacija podataka (XDR)**
   - Sun XDR, CORBA CDR, XML-RPX, JSON-RPC …
   - Poznavanje ograničenja formata i prikladnost za pojedine platforme i jezike
   - Uvijek se vrši konverzija u i iz formata zapisa

3. U formatu pošiljatelja s dodatkom meta-podataka
   - Primatelj se brine za moguće konverzije ukoliko je to potrebno
   - Npr. zapisi originalno u big-endian formatu (informacija uključena u meta podatake), konverzija se vrši samo ako primatelj radi s formatom little-endian

Svojstva XDR, klasifikacija

Sustav	Format	Norma	Podržane platforme / jezici	Binarni / tekstni	Meta-podaci uključeni
Sun RPC	XDR	da	višeplatforman, višejezičan	binarni	ne
CORBA	CDR	da	višeplatforman, višejezičan	binarni	ne
Java	Java object serialization	da	samo Java	binarni	da
XML-RPC	XML	da	višeplatforman, višejezičan	tekstni	da
JSON-RPC	JSON	da	višeplatforman, višejezičan	tekstni	ne
Web usluge	SOAP (XML)	da	višeplatforman, višejezičan	tekstni	da
YAML	YAML	ne	višeplatforman, višejezičan	tekstni	(ne)
Web	???	???	???	???	???

Primjer: XDR, CDR

- Tipovi podataka:
  - Jednostavni tipovi, složeniji tipovi (ograničene složenosti: polja, strukture, unije …)
  - Propisan konačan skup tipova i njihova definicija (integer je 16 bitni predznačen …)
  - Big/little endian:
    - XDR: big-endian format (network byte order)
    - CDR: indikacija formatu zapisa, podaci se šalju u formatu pošiljatelja
  - Tipovi podataka i parametri funkcija/metoda se definiraju u opisu sučelja (RPC IDL) / objekata (CORBA IDL):
    - Parametri se kodiraju u rezultirajući niz po redosljedu navođenja u opisu, nije potrebno uključivati meta-podatke -> učinkovit i sažet binarni zapis
    - Cjelobrojni brojevi – definrana preciznost neovisno o bilo kojoj plaformi (16 bita …)
    - Brojevi u pomičnom zarezu – slijedi se IEEE format

Primjer: Java serijalizacija

- Specifično za Javu (prenosivost podataka omogućena prenosivošću platforme):
  - Serijalizacija i rekonstrukcija strukture podataka bilo koje složenosti
  - Uključeni meta-podaci, primarno o razredima zapisanih objekata:
    - Ime razreda + sažetak -> informacija uključuje i verziju definicije razreda
  - Serijalizacija stvara „duboki” zapis strukture podataka
    - Uključena i stanja objekata referencirana od strane drugih objekata u strukturi
    - Za pojedini objekt samo jedan zapis stanja (višestruka pojavljivanja istog objekta u zapisu su naznačena referencama na prvi-originalni zapis)
    - Tranzijentne varijable se ne serijaliziraju
  - API:
    - java.io.Serializable
    - Java.io.ObjectInputStream
    - Java.io.objectOutputStream

Otvoreno računarstvo

7. Raspodijeljeni sustavi

7.1 Uvod

7.2 Modeli

7.3 Međuprocesna komunikacija

7.4 Vanjske reprezentacija podataka

7.5 Izravna komunikacija

7.6 Neizravna komunikacija

Paradigma udaljenih poziva (I)

- Razina apstrakcije viša od paradigme međuprocesne komunikacije
- **Usluga** (*service*) – upravlja skupom povezanih resursa, omogućava dostup resursima korisnicima i aplikacijama
  - usluga ispisa
  - usluga pristupa udaljenom datotečnom sustavu
  - usluga dohvata stranica Weba
  - aplikacijski specifična usluga...
- Pristup funkcionalnosti usluge putem **sučelja** (*interface*) – skupa čvrsto definiranih operacija
  - read, write
  - get, put, post, delete, head, ...

Paradigma udaljenih poziva (II)

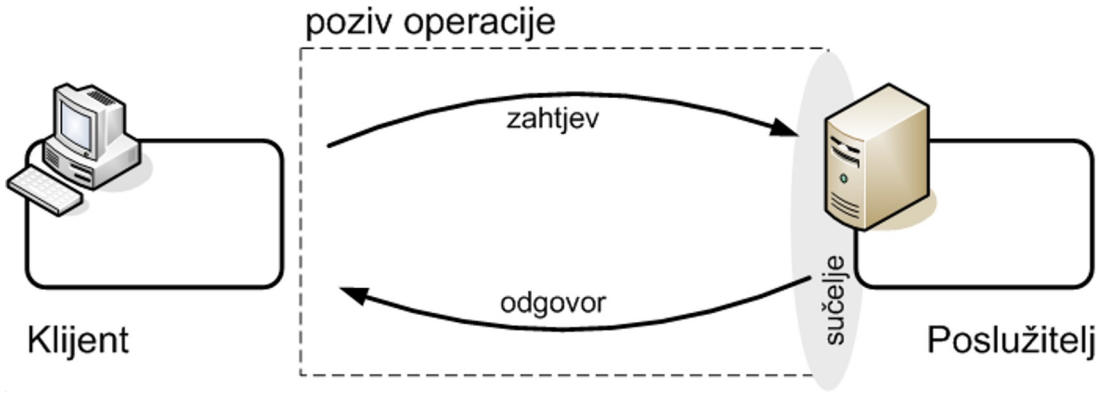
- Udaljeni poziv (*engl. remote invocation / call*) –
  poziv operacije **nad resursom**, sukladno definiranom sučelju

- Postoji **vremenska i prostorna sprega** klijenta i poslužitelja

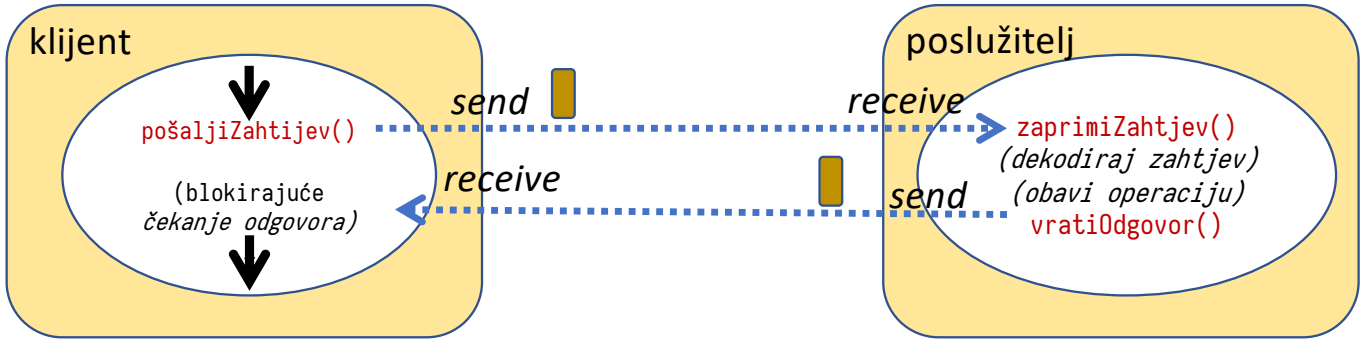
- Vrste udaljenih poziva:
  1) zahtjev-odgovor (engl. request – response)
  2) poziv udaljenih procedura (*engl. remote procedure call - RPC*)
  3) pozivi udaljenih metoda (*engl. remote method invocation - RMI*)

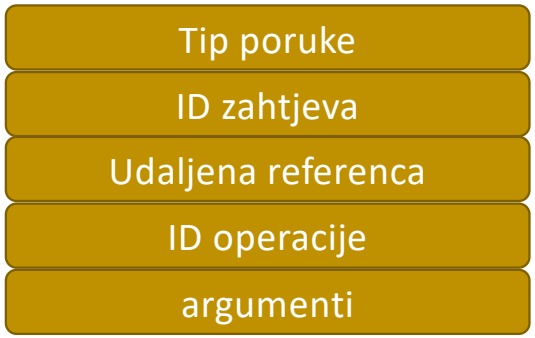
Zahtjev - odgovor

- protokol **zahtjev – odgovor**
  - izveden korištenjem metoda međuprocesne komunikacije (utičnice - TCP ili UDP)
  - jednostavan, učinkovit, ali primitivan mehanizam
  - **zahtjev** sadrži pozivanu operaciju i argumente (opcionalno i sadržaj resursa)
  - **odgovor** sadrži rezultate operacije nad resursom (opcionalno i sadržaj resursa)



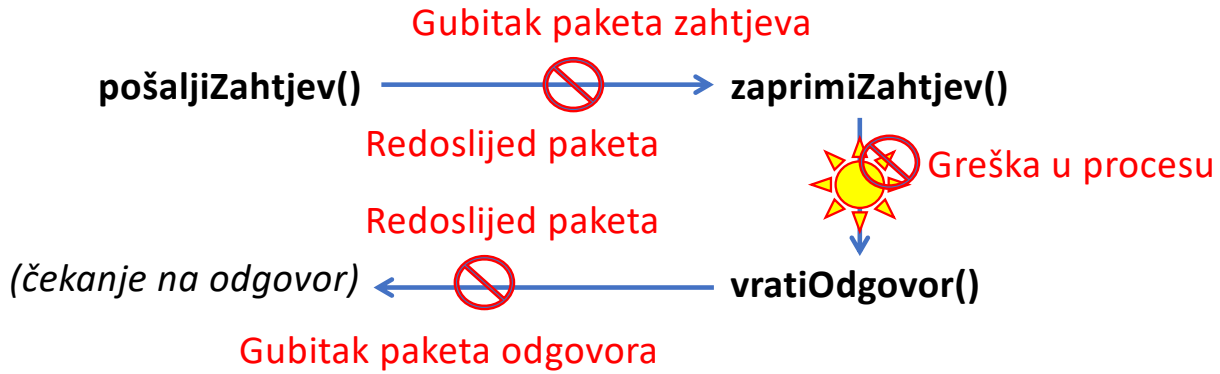
Zahtjev-odgovor korištenjem protokola UDP





- Elementi poruke
  - tip - zahtjev ili odgovor
  - ID zahtjeva – klijent generira jedinstveni broj zahtjeva
  - udaljena referenca – označava udaljeni resurs
  - ID operacije – određuje zahtjevnu operaciju nad resursom
  - argumenti – argumenti operacije ili rezultat operacije
- Identifikator zahtjeva na strani poslužitelja
  - ID zahtjeva + identifikator procesa klijenta (najčešće IP adresa+port)

Model grešaka kod protokola zahtjev-odgovor (UDP) (I)



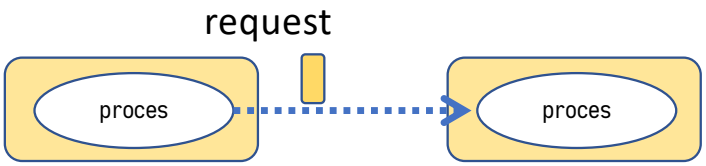
- Četiri moguća pogreške: gubitak paketa zahtjeva ili odgovora, greška u procesu obrade zahtjeva, redoslijed dostave poruka
- *pošaljiZahtjev()* je blokirajuća operacija s ograničenim periodom čekanja na poruku odgovora (timeout)
- Moguće reakcije na istek perioda čekanja:
  - Operacija dojavljuje grešku, program nastoji izvesti oporavak od greške
  - Operacija automatski pokušava retransmisiju zahtjeva (razuman broj puta)

Model grešaka kod protokola zahtjev-odgovor (UDP) (II)

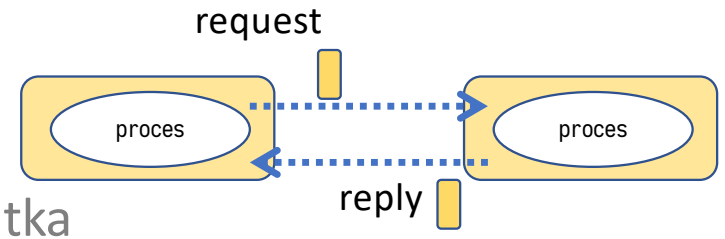
- U slučaju dolaska retransmisije zahtjeva na poslužitelj:
  - Ako poslužitelj još nije poslao odgovor na prvotni zahtjev (obrada traje dulje od perioda čekanja klijenta) – *odbacivanje ponovljenog zahtjeva*
  - Ako je odgovor već poslan, moguć je gubitak paketa odgovora, tada su opcije:
    - Ponoviti obradu zahtjeva*
    - Koristiti privremenu memoriju za pohranu rezultata obrade (povijest), pokušati pronaći kopiju rezultata i ponovno je poslati klijentu unutar paketa odgovora – nema ponovne obrade zahtjeva. Sljedeći zahtjev od strane klijenta (s većim ID zahtjeva od prethodnog zahtjeva istog klijenta) implicitna je potvrda prethodno vraćenog odgovora, te se zapis iz privremene memorije rezultata obrade za tog klijenta može obrisati
  - *__idempotentne__ operacije – isti rezultat ma koliko puta bile izvršene
    - Primjeri idempotentnih operacija: i=2, set.insert(), GET /index.html
    - Primjeri ne-idempotentnih operacija: i=i+1, sequence.insert(), POST /form.cgi

Stilovi (varijacije) protokola zahtjev-odgovor (UDP)

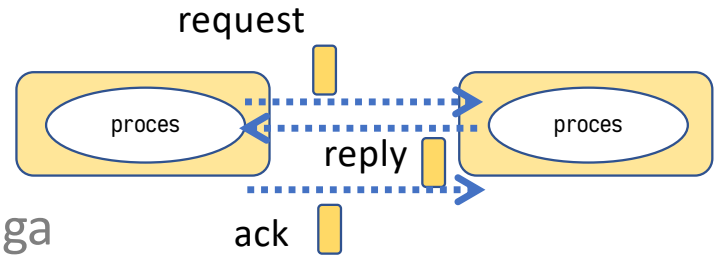
- R, RR, RRA
- Request
  - asinkrono slanje samo jedne poruke zahtjeva
  - ne čeka se odgovor, klijent nastavlja s izvođenjem



- Request-Response
  - Tipičan stil klijent-poslužitelj komunikacije
  - Odgovor poslužitelja je potvrda primitka zahtjeva klijenta
  - Sljedeći zahtjev klijenta (veći id zahtjeva) je potvrda primitka prethodnog odgovora poslužitelja



- Request-Response-Acknowledgement
  - Dodatna asinkrona potvrda klijenta poslužitelju o primitku njegova odgovora
  - Brisanje sadržaja povijesti rezultata obrade (bitno kod usluga s velikim brojem klijenata koji rijetko šalju zahtjeve)



Zahtjev-odgovor korištenjem protokola TCP

- UDP – efikasniji za poruke ograničene duljine (stanu unutar jednog paketa)
- TCP – neefikasniji, jednostavniji za korištenje kod većih količina prenošenih podataka ili kod podataka čija veličina nije unaprijed poznata
- Visoka efikasnost TCP zahtjev-odgovor protokola kroz izvođenje više RR transakcija koristeći dugotrajne kanale
  - HTTP 1.0 vs HTTP 1.1
- Model grešaka vrlo sličan
  - Umjesto gubitka paketa (UDP) mogući su prekidi veze u bilo kojem trenutku i posljedično gubitak prenošenih informacija
  - Ispravan redoslijed dostave poruka unutar TCP kanala je garantiran

Poziv udaljene procedure (I)

- Problem različitih razina apstrakcije kod programiranja raspodijeljenih sustava
  - Programski jezici nude više razine apstrakcije – procedure, razrede …
  - Zahtjev-odgovor i slične paradigme se oslanjaju na slanje i primanje poruka
  - CILJ: sakriti kompleksnost mrežne komunikacije i uklopiti ju u dominantnu paradigmu korištenu za razvoj programske podrške
- Poslužitelj nudi **sučelje** prema resursima u obliku **skupa procedura**
- Sučelje i implementacija usluge su neovisni, korisnik i implementacija usluge su neovisni**,** korisnik pristupa usluzi isključivo putem sučelja (slojevit model!)
- Jezik za opis sučelja (*engl. interface description language - IDL*)
  - naziv, argumenti, povratna vrijednost procedura, složeni tipovi podataka
  - prevodi se u programski kôd za stranu klijenta i poslužitelja

Poziv udaljene procedure (II)

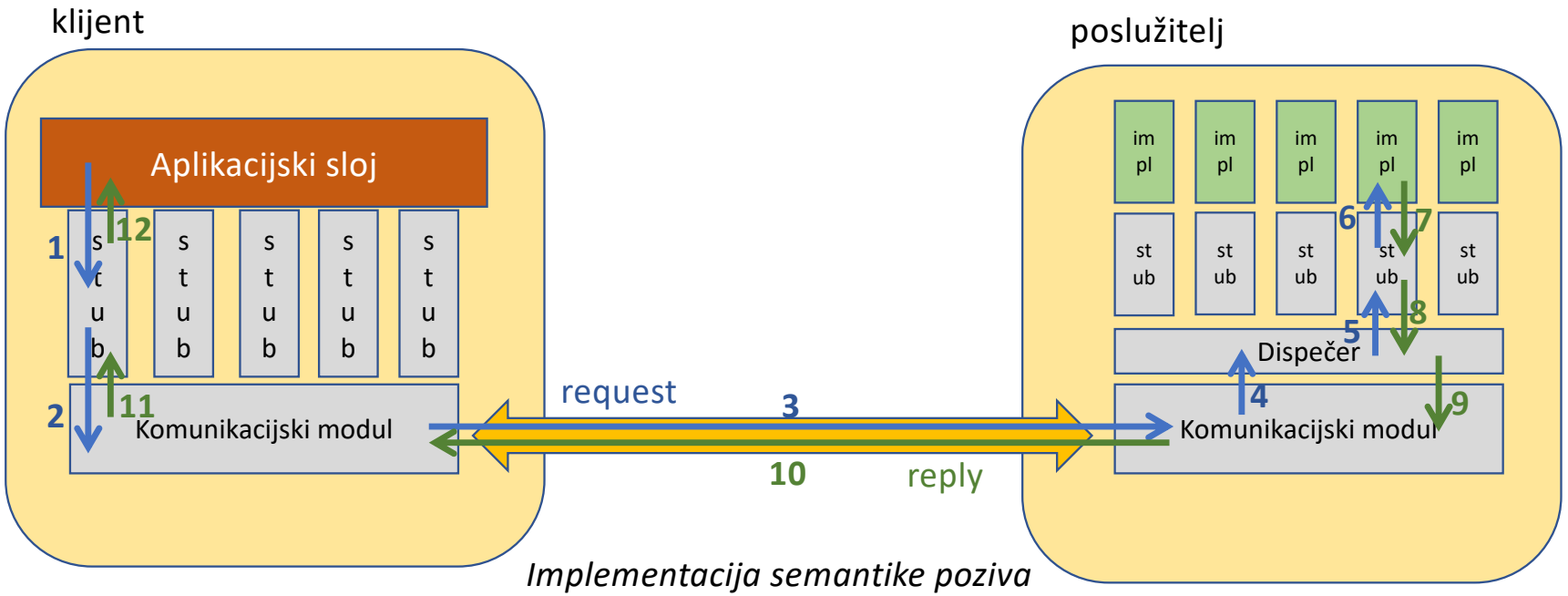
- Sustav **neovisan o platformi**
  - klijent i implementacija procedura mogu biti na različitim platformama i jezicima
  - kodiranje prenošenih argumenata – platformno i jezično neovisno: XDR, **XML, JSON, …**
- **Skrivena složenost** udaljene komunikacije i kodiranja podataka, suradnje između raznorodnih računalnih platformi
- Upravljanje promjenama:
  - implementacija usluge (procedura) se smije mijenjati – uz očuvanje semantike
  - sučelje usluge se ne smije mijenjati – u suprotnom klijenti više ne mogu koristiti uslugu

Semantika poziva udaljene procedure

Semantika	Retransmisija zahtjeva	Filtriranje dupliciranih zahtjeva	Ponovno izvođenje ili retransmisija odgovora
Možda (maybe)	ne	ne	-
Najmanje jednom (at-least-once)	da	ne	ponovno izvođenje
Najviše jednom (at-most-once)	da	da	retransmisija odgovora

- *Exactly-once:* idealno, ali kako to izvesti ?
- *Maybe:* greške su prihvatljive, npr. HTTP GET (u slučaju greške ponoviti upit)
- *At-least-once:* prihvatljivo u slučaju idempotentnih operacija
- *At-most-once:* nije prihvatljivo moguće višestruko izvođenje operacije (npr. novčana transakcija, doziranje lijeka …), radije prijava greške kao rezultat poziva, „ručni” postupak detekcije uzroka greške, određivanje trenutnog stanja sustava i pokušaja oporavka

Implementacija poziva udaljene procedure (I)



Implementacija semantike poziva

Implementacija poziva udaljene procedure (II)

1 - poziv lokalne reprezentacija udaljene procedure (*client stub* – generiran od IDL prevodioca)

2 – *stub* enkodira argumente (XDR), id procedure, prosljeđuje poziv komunikacijskom modulu

3 – komunikacijski modul (biblioteka) šalje poruku zahtjeva poslužitelju, implementira semantiku poziva (maybe …)

4 – komunikacijski modul poslužitelja zaprima zahtjev i prosljeđuje dispečeru (biblioteka)

5 – dispečer na osnovu id procedure odabire poslužiteljski stub i prosljeđuje zahtjev

6 – poslužiteljski stub (generiran od IDL prevodioca) dekodira argumente i poziva stvarnu implementaciju procedure (razvijena od programera)

7 – implementacija procedure završava s radom i vraća rezultat poslužiteljskom stubu

8 – poslužiteljski stub enkodira povratne vrijednosti i id procedure i vraća ih dispečeru

9 – dispečer prosljeđuje podatke komunikacijskom modulu

10 – komunikacijski modul formira poruku odgovora i vraća ju klijentu

11 – komunikacijski modul na klijentu prosljeđuje poruku stubu

12 – stub dekodira povratne vrijednosti i prosljeđuje ih pozivatelju lokalne reprezentacije udaljene procedure

Poziv udaljene metode

- RPC u **objektnom** svijetu
- Opis sučelja objekta koji implementira **niz metoda**
  - eksplicitni ili implicitni opisi (*CORBA IDL* / *JavaRMI*)
  - naziv metode, argumenti, povratna vrijednost
- Iz opisa se stvaraju programski entiteti
  - strana klijenta – objekti zastupnici (*proxy objects)*
    - transparentnost za korisnika
    - poziv se prosljeđuje metodi udaljenog objekta
    - argument i povratna vrijednost može biti referenca udaljenog objekta
  - strana poslužitelja – *skeleton*
    - "prazna" implementacija metoda u udaljenom sučelju
    - priprema parametara i prosljeđivanje pravoj lokalnoj metodi
    - vraćanje rezultata *proxy* objektu klijenta

Otvoreno računarstvo

7. Raspodijeljeni sustavi

Mehanizmi neizravne komunikacije

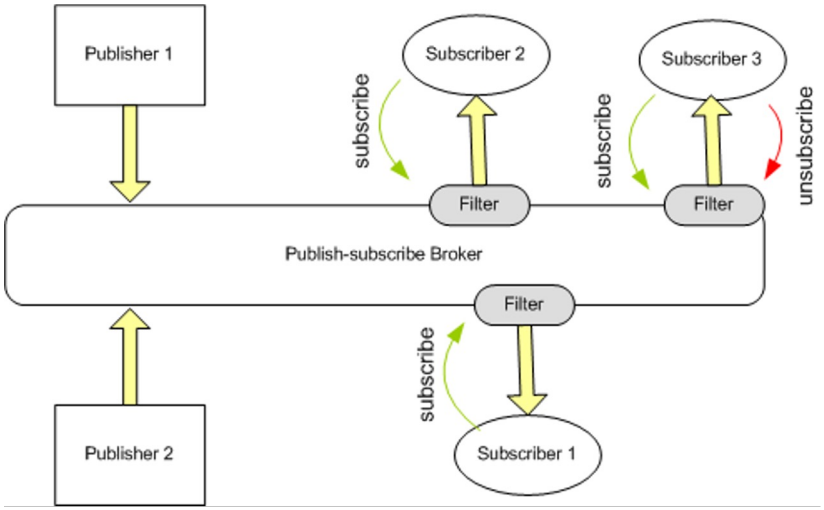
1) Grupna komunikacija
2) Objavi – pretplati
3) Redovi poruka
4) Raspodijeljena dijeljena memorija
5) Prostori podataka

Grupna komunikacija

- Apstrakcija grupne komunikacije
  - pošiljatelj šalje poruku grupi primatelja (1:n)
  - pošiljatelju **nisu poznati pojedini članovi** grupe
  - svi aktivni članovi grupe primaju odaslanu poruku
- Grupa primatelja:
  - apstrakcija grupe temelji se na **jedinstvenom identifikatoru grupe**
  - entitet se pridružuje grupi
  - član grupe može napustiti grupu
- Sprege između pošiljatelja i članova grupe
  - vremenska postoji
  - prostorna (najčešće) ne postoji

Objavi – pretplati (I)

- Veći broj pošiljatelja i primatelja (komunikacija n:m)
- Sustav temeljen na događajima/porukama
- Temelji se na **posredničkom entitetu** koji:
  - zaprima **pretplate** na događaje od zainteresiranih entiteta - pretplate mogu biti kompleksni filtri s obzirom na sadržaj
  - zaprima brisanja pretplata
  - prihvaća poruke objavljene od entiteta objavitelja i **prosljeđuje** ih svim **pretplaćenim entitetima** (ako je zadovoljen filtar pretplate)



Objavi – preplati (II)

- Sprega između pretplatnika i objavitelja
  - vremenska postoji
  - prostorna ne postoji
- Jedinstveni protokol komunikacije objavitelja i pretplatitelja
  - definiran protokolom prema posredničkom entitetu

Redovi poruka (I)

- Posrednički entitet koji sadrži **trajne redove poruka**
- Entitet može biti vlasnik jednog ili više redova poruka
- Entitet pošiljatelj:
  - postavlja poruku u red poruka ciljnog entiteta
    - komunikacija 1:1
  - posrednički entitet sigurno sprema poruku
- Entitet vlasnik:
  - provjerava postojanje poruka u redu poruka
  - zaprima poruku iz reda poruka (poruka se briše iz reda)

Redovi poruka (II)

- Entiteti koji komuniciraju ne moraju biti istovremeno aktivni
  - vremenska sprega ne postoji
  - prostorna sprega postoji
- Jedinstveni protokol komunikacije entiteta
  - definiran protokolom prema posredničkom entitetu

Raspodijeljena dijeljena memorija

- Temelji se na mehanizmu međuprocesne komunikacije – **dijeljena memorija**
  - između procesa istog računala
- **Lokalne kopije** dijeljene memorije na **svakom računalu**
- **Propagacija** promjena iz lokalne kopije u lokalne kopije svih ostalih računala
  - problem koherencije
- Sprega između računala
  - prostorna ne postoji
  - vremenska postoji

Prostori podataka

- Posrednički entitet
  - pruža prostor - trajnu memoriju (*tuple space*) koja služi čuvanju **strukturiranih** podataka (*tuples*)
- Entiteti proizvođači
  - postavljaju podatke u prostor
- Entiteti potrošači
  - čitaju podatke iz prostora (podatak ostaje u prostoru)
  - vade podatke iz prostora
    - čitanje ili vađenje na osnovu kriterija definiranih nad sadržajem podataka
- Sprega između proizvođača i potrošača
  - prostorne sprege nema
  - vremenske sprege nema

Pitanja?